Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling

Vladimir Svetnik,*,† Andy Liaw,† Christopher Tong,† J. Christopher Culberson,‡ Robert P. Sheridan,§ and Bradley P. Feuston‡

Biometrics Research, Merck Research Laboratories, P.O. Box 2000, Rahway, New Jersey 07065, Molecular Systems, Merck Research Laboratories, P.O. Box 4, West Point, Pennsylvania 19486, and Molecular Systems, Merck Research Laboratories, P.O. Box 2000, Rahway, New Jersey 07065

Received July 30, 2003

A new classification and regression tool, Random Forest, is introduced and investigated for predicting a compound's quantitative or categorical biological activity based on a quantitative description of the compound's molecular structure. Random Forest is an ensemble of unpruned classification or regression trees created by using bootstrap samples of the training data and random feature selection in tree induction. Prediction is made by aggregating (majority vote or averaging) the predictions of the ensemble. We built predictive models for six cheminformatics data sets. Our analysis demonstrates that Random Forest is a powerful tool capable of delivering performance that is among the most accurate methods to date. We also present three additional features of Random Forest: built-in performance assessment, a measure of relative importance of descriptors, and a measure of compound similarity that is weighted by the relative importance of descriptors. It is the combination of relatively high prediction accuracy and its collection of desired features that makes Random Forest uniquely suited for modeling in cheminformatics.

1. INTRODUCTION

A QSAR modeling and compound classification tool has to meet a number of sometimes conflicting requirements to be effective in modern drug discovery and development processes. (In this paper we will use the term "QSAR" to refer to using molecular structure to predict both quantitative and categorical activities.) The complexity of these requirements has been rapidly increasing in recent years. Among them are high accuracy of prediction, ability to handle molecular diversity (e.g., multiple mechanisms of action), ability to handle data characterized by a very large number and diverse types of descriptors, ease of training, model interpretability, and computational efficiency. The demand for QSAR modeling tools has traditionally been satisfied by the "supply" coming from Statistics, Machine Learning, etc. A few well-known examples include Decision Tree (DT, or Recursive Partitioning),[1] Artificial Neural Networks (ANN),[2,3] Partial Least Squares (PLS),[4] *k*-Nearest Neighbors (*k*-NN),[3] Multiple Linear Regression (MLR),[3] Linear Discriminant Analysis (LDA),[5] and Support Vector Machines (SVM).[2]

All of these methods have a proven record of many successful applications in QSAR modeling. However, their use in industry still remains quite limited. One of the reasons for this comes from the fact that many QSAR tools do not possess a "right" combination of features required for their successful use. For example, *k*-NN, ANN, and nonlinear SVM have high prediction performance and are flexible enough to model multiple mechanisms of action. However, ANN and *k*-NN are not efficient in dealing with high-dimensional data without dimension reduction or preselection of descriptors (e.g., by using genetic algorithms[3]). Nonlinear SVM is capable of dealing with high-dimensional data but is not robust to the presence of a large number of irrelevant descriptors, thus requiring descriptor preselection as well. Linear methods such as PLS, MLR, and LDA may not be suitable for dealing with multiple mechanisms of action. Also, unlike PLS and linear SVM, both MLR and LDA can only handle data sets where the number of descriptors is smaller than the number of molecules, unless again a preselection of descriptors is carried out (e.g., by using genetic algorithms[3,5]).

Decision Tree is probably the closest to having the desired combination of features. It handles high-dimensional data well; has the ability to ignore irrelevant descriptors; handles multiple mechanisms of action; and is amenable to model interpretation. Very efficient Decision Tree building algorithms for QSAR are available too.[1] The major drawback, however, is that Decision Tree usually has relatively low prediction accuracy. This drawback may impede its use in applications such as virtual screening of compound libraries. In this application where the number of screened compounds can be on the order of 10^5 to 10^6, even a small fraction of a percent in accuracy makes a profound difference in the number of false positive and negative classifications.

Because of the great appeal of Decision Tree, there have been many efforts to improve its prediction accuracy. These attempts resulted in a large number of various tree-based algorithms.[6] It has recently been discovered that one of the best ways to improve the performance of Decision Tree-based algorithms is to use *ensembles* of trees.[7] In this paper we present one such ensemble method, *Random Forest*.[8]

* Corresponding author phone: (732)594-5544; fax: (732)594-1565; e-mail: vladimir_svetnik@merck.com.
† Biometrics Research.
‡ Molecular Systems, West Point, Pennsylvania.
§ Molecular Systems, Rahway, New Jersey.

Table 1. List of Major Abbreviations

ANN	artificial neural network
BBB	blood-brain barrier
CART	classification and regression trees
CNS	central nervous system
COX-2	cyclooxygenase-2
CV	cross-validation
DT	decision tree
ER	error rate
GA	genetic algorithm
k-NN	*k*-nearest neighbors
LDA	linear discriminant analysis
MDRR	multidrug resistance reversal
MLR	multiple linear regression
MOA	mechanism of action
MSE	mean square error
OOB	out-of-bag
P-gp	P-glycoprotein
PLS	partial least squares
QSAR	quantitative structure−activity relationship
RF	random forest
RMSE	root mean square error
SVM	support vector machine

Unlike the other ensemble methods, Random Forest offers some unique features that make it suitable for QSAR tasks. These include built-in estimation of prediction accuracy, measures of descriptor importance, and a measure of similarity between molecules.

In this paper we will present data supporting the viewpoint that Random Forest is a way to improve the performance of Decision Tree while retaining most of the latter's appealing properties. We keep in mind Wolpert's No Free Lunch theorem[9] that there is no one best algorithm for all problems. We only intend to show that in terms of prediction accuracy Random Forest is among the top performers, rather than "the top" performer, even without parameter tuning or descriptor reduction. It is only the combination of relatively high prediction accuracy and its collection of desired features that make Random Forest very attractive for QSAR.

This paper is organized as follows. In section 2 we describe in detail the Random Forest algorithm and its features. In section 3 we describe six data sets which we used to evaluate Random Forest prediction accuracy in classification and regression; the results are presented in section 4. Section 5 illustrates the optimization of the Random Forest algorithm by parameter optimization and removing irrelevant descriptors. Section 6 illustrates the use of Random Forest's additional features: out-of-bag performance assessment, descriptor importance, and intrinsic proximity. Section 7 is a concluding Discussion. For additional insights into how Random Forest improves upon the performance of a single tree, in the Appendix we use the bias-variance decomposition to show how Random Forest reduces the variance of prediction while retaining low bias. Since many acronyms are used in this paper, a list of the major abbreviations is presented in Table 1.

2. RANDOM FOREST

In this section we present the Random Forest, its training procedure, and its additional features, which were all first proposed by Breiman.[8]

Ensemble of Trees. Random Forest is an ensemble of B trees $\{T_1(X), ..., T_B(X)\}$, where $X = \{x_1, ..., x_p\}$ is a p-dimensional vector of molecular descriptors or properties associated with a molecule. The ensemble produces B outputs $\{\hat{Y}_1 = T_1(X), ..., \hat{Y}_B = T_B(X)\}$ where $\hat{Y}_b$, $b = 1, ..., B$, is the prediction for a molecule by the bth tree. Outputs of all trees are aggregated to produce one final prediction, $\hat{Y}$. For classification problems, $\hat{Y}$ is the class predicted by the majority of trees. In regression it is the average of the individual tree predictions.

Training Procedure. Given data on a set of n molecules for training, $D = \{(X_1, Y_1), ..., (X_n, Y_n)\}$, where X_i, $i = 1, ..., n$, is a vector of descriptors and Y_i is either the corresponding class label (e.g., active/inactive) or activity of interest (e.g., $-\log IC_{50}$), the training algorithm proceeds as follows.

(1) From the training data of n molecules, draw a bootstrap sample (i.e., randomly sample, *with replacement*, n molecules).

(2) For each bootstrap sample, grow a tree with the following modification: at each node, choose the best split among a randomly selected subset of m_{try} (rather than all) descriptors. Here m_{try} is essentially the only tuning parameter in the algorithm. The tree is grown to the maximum size (i.e., until no further splits are possible) and not pruned back.

(3) Repeat the above steps until (a sufficiently large number) B such trees are grown.

When $m_{try} = p$, i.e., the best split at each node is selected among all descriptors, the Random Forest algorithm is the same as Bagging.[10] The tree growing algorithm used in Random Forest is CART[11] although other alternatives could be considered as well.

One may suspect that the computational complexity of an ensemble of, say, 100 trees, is probably 100 times that of a single tree. This is actually not true. The Random Forest algorithm can be very efficient, especially when the number of descriptors is very large. The efficiency of the algorithm, compared to that of growing a single decision tree, comes from two differences between the two algorithms. First, in the usual tree growing algorithm, all descriptors are tested for their splitting performance at each node, while Random Forest only tests m_{try} of the descriptors. Since m_{try} is typically very small (the default in the software is the square root of the number of descriptors for classification), the search is very fast. (In other words, at each node, the Random Forest algorithm effectively only "sees" m_{try}, rather than p, descriptors.) Second, to get the right model complexity for optimal prediction strength, some pruning is usually needed for a single decision tree. This is typically done via cross-validation and can take up a significant portion of the computations. Random Forest, on the other hand, does not do any pruning at all. We have found that in cases where there are an excessively large number of descriptors, Random Forest can be trained in less time than a single decision tree.

The other prominent ensemble learning methods besides Random Forest are Bagging[10] and Boosting,[12,13] both of which have also been shown to significantly improve performance over that of a single tree. As discussed above, Bagging can be thought of as a Random Forest with $m_{try} = p$. Boosting is a sequence of trees, each trained on all the data, but with data points reweighted in each tree according to whether they were misclassified by the previous tree in the sequence, for classification. In regression, each tree is grown on the residuals of the previous trees. Prediction is done by weighted vote (in classification) or weighted average (in regression) of the ensemble outputs. It has been shown empirically that Boosting and Random Forest usually outperform Bagging and are quite similar to each other in performance.[14] We also mention two other ensemble methods because they have been used in QSAR modeling: Random FIRM[15] and Decision Forest.[6] In Random FIRM, there is an ensemble of trees, each built on all the training data, but at each split, a descriptor is randomly selected according to probabilities related to the descriptor's significance from an appropriate statistical test defining the split. Although Random FIRM could certainly be used for prediction, using one of the tree aggregation procedures, it is used primarily for model interpretation. Decision Forest is an ensemble built such that each tree uses descriptors not available to the other trees but such that the prediction accuracy of each tree does not fall below a specified threshold. The first known comparison of Decision Forest and Random Forest, presented in section 4, shows that they have similar performance on one data set. For a discussion of other ensemble learning methods, see ref 7.

Out-of-Bag Estimate of Performance. Ideally, an assessment of performance for a prediction algorithm should be done using a large independent test data set that was not used in the training. In practice when the data is limited, some type of cross-validation[16] is usually used, which, in some cases, could be computationally cumbersome. Random Forest performs a type of cross-validation *in parallel* with the training step by using the so-called *Out-Of-Bag* (OOB) samples.[17] Specifically, in the process of training, each tree is grown using a particular bootstrap sample. Since bootstrapping is sampling with replacement from the training data, some of the molecules will be "left out" of the sample, while some others will be repeated in the sample. The "left out" molecules, D^{OOB}, constitute the Out-of-Bag (OOB) sample. On the average, each tree is grown using about $1 - e^{-1} \approx 2/3$ of the training molecules,[18] leaving $e^{-1} \approx 1/3$ as OOB. Because OOB molecules have not been used in the tree construction, one can use them to estimate the ensemble prediction performance in the following way. Let $D_{\mathrm{b}}^{\mathrm{OOB}}$ be the OOB part of the data for the *b*th tree. We can then use the *b*th tree to predict $D_{\mathrm{b}}^{\mathrm{OOB}}$. Since each training molecule, X_{i}, is in an OOB sample, on the average, about 1/3 of the time, we can calculate an ensemble prediction $\hat{Y}^{\mathrm{OOB}}(X_{\mathrm{i}})$ by aggregating only its OOB predictions. Calculate an estimate of the error rate ER for classification or mean square error (MSE) for regression by

$$\mathrm{ER} \approx \mathrm{ER}^{\mathrm{OOB}} = n^{-1}\sum_{i=1}^{n} I(\hat{Y}^{\mathrm{OOB}}(X_{\mathrm{i}}) \neq Y_{\mathrm{i}})$$

$$\mathrm{MSE} \approx \mathrm{MSE}^{\mathrm{OOB}} = n^{-1}\sum_{i=1}^{n} \{\hat{Y}^{\mathrm{OOB}}(X_{\mathrm{i}}) - Y_{\mathrm{i}}\}^2$$

where $I(\cdot)$ is the indicator function. In section 6 we compare OOB performance estimation with *k*-fold cross-validation and show that they are in reasonably good agreement, suggesting that the assessment of Random Forest performance indeed does not require additional cross-validation.

Descriptor Importance. Decision Tree is known for its ability to select "important" descriptors among many and ignore (often irrelevant) others. In addition, Decision Tree gives an explicit model describing the relationship between these descriptors and predictions, thus easing model interpretation. Random Forest, as an ensemble of trees, inherits the ability to select "important" descriptors. However, it does not produce an explicit model. Instead, the relationship between descriptors and activity of interest is hidden inside a "black box".[19] Nonetheless, a measure of how each descriptor contributes to the prediction accuracy of Random Forest can be calculated in the course of training.

When a descriptor that contributes to prediction accuracy is "noised up" (e.g., replaced with random noise), the accuracy of prediction should noticeably degrade. On the other hand, if a descriptor is irrelevant, "noising" it up should have little effect on the performance. It turns out that in the classification case the change in the prediction accuracy is usually a less sensitive measure than the change in the *margin*. In words, the margin is the difference between the proportion of votes for the correct class and the maximum of the proportion of votes for the incorrect classes.[20] As an example, consider a three-class problem. Supppose a molecule belonging to class 1 has 60%, 30% and 10% votes for classes 1, 2, and 3, respectively, then the margin is equal to $0.6 - \max(0.3, 0.1) = 0.3$. When the margin is positive, the ensemble prediction is correct, while when the margin is negative, the ensemble prediction is incorrect. The tendency of the margin to become smaller (more negative or less positive) when a descriptor is "noised" up is what is used to assess the descriptor importance in classification.

In detail, the importance of descriptors is calculated according to the following procedure. As each tree is grown, make predictions on the OOB data for that tree. At the same time, each descriptor in the OOB data is randomly permuted, one at a time, and each such modified data set is also predicted by the tree. At the end of the model training process, the margins for each molecule are calculated based on the OOB prediction as well as the OOB predictions with each descriptor permuted. Let M be the average margin based on the OOB prediction and M_{j} the average margin based on the OOB prediction with the *j* th descriptor permuted. Then the measure of importance for the *j*th descriptor is simply $M - M_{\mathrm{j}}$. For regression problems, the margins are simply replaced by squared prediction errors.

Intrinsic Proximity Measure. Clustering of molecules in descriptor space as well as various analyses of the molecule neighborhoods are important problems in QSAR modeling.[21] Usually this is done in an unsupervised mode, i.e., using some measure of the similarity between two molecules in the descriptor space only. In many applications, however, researchers may be interested in the proximity measure determined by the subset of descriptors which are the most relevant to the activity of interest. Random Forest offers such a measure, called intrinsic proximity. This proximity could be calculated between any pair of molecules (they may both be in either the training or test sets or even one from each).

The proximity measure is calculated as follows. Predict a set of molecules whose pairwise proximity is of interest, using the model built by Random Forest. The proximity between a pair of molecules is the proportion of trees in the ensemble where the pair landed in the same terminal node. The proximity between a molecule and itself is thus always one. Intuitively, this measure of proximity has two advantages: It is "supervised" because the activity of interest dictates the structure of the trees in the forest. Also, because irrelevant descriptors contribute little to the ensemble, they have little influence on the proximity measure. (This proximity measure is potentially invalid for the case where both molecules are in the training set, unless only out-of-bag data are used to compute it. This should be a topic for future investigation.) In section 6 we compare results of the clustering using Random Forest proximity with one utilizing a commonly used similarity metric, the Tanimoto distance.

3. DATA SETS AND ASSESSMENT PROCEDURES

In this section, we describe the six publicly available QSAR data sets that we used to evaluate Random Forest and other QSAR methods. Four of these data sets presented classification problems (BBB, Estrogen, P-gp, and MDRR), one presented a regression problem (Dopamine), and the last one (COX-2) presented data that could be treated as either classification or regression (we did both). A summary of the data sets is shown in Table 2.

As discussed below, a large variety of different methods was used by the original authors to model these data sets, and the reported performances were quite impressive. Comparison of Random Forest with these methods will illustrate that Random Forest is either the top or among the top performers, which is the main goal of this paper. We will use two approaches of assessing the performance of Random Forest. The first and more systematic approach will be to do a simultaneous performance assessment using 50 replications of 5-fold cross-validation for three methods: Random Forest (RF), Decision Tree (DT), and Partial Least Squares (PLS). The second and more opportunistic approach will be, for each data set, to imitate the original authors' performance assessment procedures for the same three methods and compare them with the reported performances of the authors' QSAR models. Below, we describe the data sets and the authors' assessment procedures, case by case.

The selection of Decision Tree and PLS for systematic comparison is motivated by the following reasons. As we discussed in the Introduction, Decision Tree has many of the required features for QSAR modeling, except for its relatively low accuracy. We thus compare Random Forest with Decision Tree to illustrate the gain in performance due to aggregating the ensemble. PLS is a commonly used modeling tool with a proven track record of success, and it has built-in dimension reduction. In principle, we should have included other common methods in the comparison. However, this would have required us to overcome the problems with these methods discussed in the Introduction, especially dimension reduction or removal of irrelevant descriptors. There are many dimension reduction methods available, but none is clearly superior to others, further complicating the use of algorithms that require them. We thus decided to restrict the systematic part of our investigation to comparing Random Forest with Decision Tree and PLS.

BBB. Doniger et al.[2] studied a set of molecules' permeabilities across the blood-brain barrier (BBB) into the central nervous system (CNS). A set of 325 compounds was culled from the literature and from a number of databases, and each compound was rated CNS active (180 compounds) or inactive (145 compounds) based on their passive diffusion through the BBB. There were nine continuous descriptors used (all given in their paper[2]), chemical and physical properties such as molecular weight, octanol−water partitioning coefficient (logP), various hydrogen bonding characteristics, etc. Doniger et al.[2] employed two QSAR modeling techniques: neural networks and support vector machines. They found that the support vector machine is the best performer. The authors' assessment procedure was as follows. The data were randomly split into training and test sets, where the test set has 25 actives and 25 inactives. Thirty different such random splits of the data were used, and the average performance over these 30 training/test performances was taken as the overall measure of performance.

Estrogen. Tong et al.[6] studied a set of 232 compounds measured by the National Center for Toxicological Research (NCTR) for their estrogen receptor binding activity. There are 131 actives (binding) and 101 inactives (nonbinding) in the data set. Dr. Weida Tong supplied us with a set of 197 Cerius-2 descriptors (of 8 different types). The authors used their own Decision Forest to build a QSAR model of the data.[6] Their assessment procedure is to randomly split the data into a training set (2/3 of the data) and test set (1/3 of the data), performing 2000 different such random splits of the data.

P-gp. Penzotti et al.[22] compiled a set of 186 compounds from the literature, with their reported P-glycoprotein (P-gp) transport activity. In this data set, 108 compounds were reported as P-gp substrates and 78 were P-gp nonsubstrates. We worked with a set of 1522 binarized atom pair descriptors[23] generated in-house. (The original data set contained 195 compounds, but a number of them had proprietary structures that were not disclosed. We omitted the proprietary compounds from our study.) Penzotti et al.[22] built their own QSAR model based on sophisticated 3D pharmacophores. They assessed the performance of their procedure by randomly splitting the data into a training set and a test set, the latter consisting of 50 compounds that they identified in their paper.

MDRR. Bakken and Jurs[5] studied a set of compounds originally discussed by Klopman et al.,[24] who were interested in multidrug resistance reversal (MDRR) agents. The original response variable is a ratio measuring the ability of a compound to reverse a leukemia cell's resistance to adriamycin. However, the problem was treated as a classification problem, and compounds with the ratio >4.2 were considered active, and those with the ratio ≤ 2.0 were considered inactive. Compounds with the ratio between these two cutoffs were called moderate and removed from the data for two-class classification, leaving a set of 528 compounds (298 actives and 230 inactives). (Various other arrangements of these data were examined by Bakken and Jurs,[5] but we will focus on this particular one.) We did not have access to the original descriptors, but we generated a set of 342 descriptors of three different types that should be similar to the original descriptors, using the DRAGON[25] software. Bakken and Jurs[5] developed a QSAR model based on a genetic algorithm (GA) for descriptor selection and linear discriminant analysis for classification. (They also tried *k*-nearest neighbors and radial basis function neural networks; both underperformed the LDA.) For assessment, the data set was randomly split into a training set and a test set, the latter containing 77 inactives and 100 actives. The identities of the test set molecules were not reported.

Table 2. Summary of Data Sets Used for Performance Comparison

name	QSAR for	no. of samples	no. of descriptors	property[a] or quantity[b] to predict	no. of active/inactive[a] or range of activity[b]	descriptor type
BBB	CNS permeability[2] (blood-brain barrier)	325	9	active/inactive	180/145	chemical & physical properties
estrogen	estrogen receptor binding[6]	232	197	binding/nonbinding	131/101	Cerius-2
P-gp	P-glycoprotein transport activity[22]	186	1522	nonsubstrate/substrate	78/108	binarized atom pairs
MDRR	multidrug resistance reversal activity[5]	528	342	active/inactive	298/230	DRAGON
COX-2 (classification)	COX-2 inhibition[3]	314	135	active/inactive	153/161	topological
COX-2 (regression)	COX-2 inhibition[3]	272	135	$\log(IC_{50})$	0.23−5.00	topological
dopamine	dopamine receptor binding affinity[26]	116	374	$-\log(IC_{50})$	4.60−8.20	binarized atom pairs

[a] For classification. [b] For regression.

Dopamine. Gilligan et al.[26] synthesized a set of 116 disubstituted piperidines and tested them for their binding affinity for the dopamine D_2 receptor. These drugs are of interest for treating schizophrenia. The biological activity is the $-\log(IC_{50})$, which ranges from 4.60 to 8.20. The authors did not develop a QSAR model for their data, although Sheridan et al.[4] used a partial least squares approach to study the data set. We decided to redo the PLS analysis using 374 binarized atom pair descriptors[23] generated in-house.

COX-2. Kauffman and Jurs[3] studied a set of 314 compounds whose inhibitory activities for the cyclooxygenase-2 (COX-2) enzyme were assayed by a laboratory at G. D. Searle and reported in a series of five literature publications. These drugs are of interest for use as nonsteroidal antiinflammatory drugs (NSAIDs). The biological activity is the $\log(IC_{50})$, which ranges from 0.23 to 5.00 reported for 272 compounds; the remainder have only >5.00 recorded. For regression, only the 272 compounds with definite values were used; for classification, a cutoff of 2.5 was applied to the $\log(IC_{50})$, with values smaller than 2.5 considered active and values greater than 2.5 considered inactive. All 314 compounds were used in classification, so that there were 153 actives and 161 inactives. The original set of 135 continuous topological descriptors was kindly provided to us by Dr. Gregory Kauffman and Prof. Peter Jurs. The authors used genetic algorithms to reduce the number of descriptors; multiple linear regression (MLR) and artificial neural networks (ANN) were applied for regression (the MLR was also used to further reduce descriptors prior to inputting them to the ANN). They used linear discriminant and *k*-nearest neighbors for classification. In regression, MLR and ANN had about the same test set performance, but in classification *k*-NN performed better than LDA. For regression, 10% of the compounds (27 of them) were randomly selected as a test set; for classification, 30 actives and 30 inactives were randomly chosen to be in a test set. The identities of the test set compounds were not reported.

4. RESULTS OF COMPARISONS ON EXAMPLE DATA SETS

The results of the analyses described above are shown in Tables 3−6. In the tables, “accuracy” in classification is the proportion of correctly classified test samples to the total number of test samples. For regression, we report both the root-mean-square error (RMSE) and the correlation between the predicted and actual values of the response variable of the test data. We actually report only medians of 50 replications of 5-fold cross-validation in Tables 3 and 4.

Table 3. Median Accuracy from 50 5-Fold Cross-Validations for the Classification Data Sets

data	RF	DT	PLS
BBB	0.809	0.737	0.691
estrogen	0.828	0.759	0.813
P-gp	0.806	0.712	0.769
MDRR	0.830	0.780	0.821
COX-2	0.780	0.736	0.774

Table 4. Median RMSE and Correlation from 50 5-Fold Cross-Validations for the Regression Data Sets

	RMSE			corr		
data	RF	DT	PLS	RF	DT	PLS
dopamine	0.510	0.636	0.538	0.698	0.504	0.658
COX-2	0.828	0.984	0.921	0.658	0.525	0.573

Table 5. Comparison of Accuracy Assessed as In the Original Papers, for Classification

data	RF	DT	PLS	other
BBB	0.800	0.732	0.678	0.815 (SVM)[2] 0.757 (ANN)[2]
estrogen	0.818	0.753	0.805	0.800 (Dec. Forest)[6]
P-gp	0.700	0.700	0.680	0.627 (see ref 22)
MDRR	min: 0.780 med: 0.830 max: 0.870	min: 0.701 med: 0.780 max: 0.836	min: 0.757 med: 0.816 max: 0.870	0.831 (GA/LDA)[5]
COX-2	min: 0.700 med: 0.783 max: 0.933	min: 0.583 med: 0.742 max: 0.917	min: 0.650 med: 0.783 max: 0.850	0.833 (GA/*k*-NN)[3]

Table 6. Comparison of Regression Performance Assessed as In the Original Paper,[3] for COX-2 Regression Data

	RF	DT	PLS	GA/MLR[3]	GA-MLR/ANN[3]
RMSE	min: 0.613 med: 0.842 max: 1.010	min: 0.793 med: 1.000 max: 1.161	min: 0.645 med: 0.914 max: 1.101	0.655	0.625
corr	min: 0.455 med: 0.674 max: 0.766	min: 0.324 med: 0.500 max: 0.728	min: 0.384 med: 0.594 max: 0.792	0.666	0.719

Random Forest results were obtained using the R package randomForest.[27] The results reported here are based on using all descriptors, the default m_{try} (for classification, the square root of the number of descriptors; for regression, one-third of the number of descriptors) and 500 trees in the forest. See further discussion on the effect of descriptor selection and optimization of m_{try} on Random Forest performance in section 5. Decision Tree results were obtained using the R package rpart[28] which, by default, does cost-complexity pruning via 10-fold cross-validation. All parameters were kept at their default values. PLS results were obtained using the R package pls.pcr.[29] For each split of the data, we find the optimal number of factors via 10-fold cross-validation on the training set. For classification problems, the "optimal" number of factors is found with respect to error rate, rather than MSE.

In principle, PLS is a regression technique. However, it is also commonly used as a classification tool.[30,31] In this paper, we used the simplest classification modification: we train using a response variable taking only two values, 0 and 1, representing the two classes, and subsequently use a threshold of 0.5 on the predicted values, to classify the test data. More sophisticated modifications[30] could potentially improve PLS performance.

As we can see from Tables 3 and 4, Decision Tree is uniformly less accurate than Random Forest on all data sets. The performance of PLS is comparable to that of Random Forest except in the BBB and COX-2 regression cases, where Random Forest performed better than PLS. In Table 5, we see that in the BBB data, Random Forest performed comparably with Support Vector Machines, which, in turn, outperformed ANN. On the estrogen data, Random Forest and Decision Forest have comparable performance.

On the three data sets P-gp, COX-2, and MDRR, the original authors reported performance results of their QSAR methods on only one split of the data into training and test sets. In the P-gp case, the particular split used was reported by the authors; this was not the case for the COX-2 and MDRR data sets. Usually, the evaluation of performance on one particular split of the data could be quite questionable unless the number of molecules is large, due to the variability of the performance estimates on different test sets. We nevertheless proceeded to do so, and in the case of P-gp data, report our results on the same split used by the original authors. For the COX-2 and MDRR data sets, we report the range and median performance over 50 random training-test splits of the data using the same size of training and test sets as the authors did. We consider two methods equivalent if the results reported by the authors are within the range of our results and do not attempt to make any conclusion about which one is better.

For the P-gp data, Random Forest outperformed the authors' pharmacophore model.[22] However, on the authors' single test data set, their model has a true positive rate (proportion of correctly classified nonsubstrates) of 79%, compared to 72% for random forest. On the other hand, their model has a false positive rate (proportion of incorrectly classified substrates) of 47%, compared to 31% for Random Forest. Random Forest's reduction in false positive rate of about 16% compared to the pharmacophore model is paid for by a relatively modest reduction in true positive rate of about 7%. Such a low false positive rate combined with a reasonably high true positive rate scores Random Forest favorably with respect to other models in this application.

The performance of k-NN and LDA (both with an optimal set of descriptors selected by a genetic algorithm) is within the range of Random Forest's performance on the COX-2 and MDRR data sets. Thus we consider them comparable. The same can be said about the performance of the Kauffman–Jurs procedures (GA for descriptor selection, followed by MLR, as well as using both GA and MLR to select descriptors for ANN) in the COX-2 regression data (Table 6). For the dopamine regression data, Gilligan et al.[26] did not report developing a QSAR model, so no comparison with the authors' method is presented.

In summary, where comparisons are meaningful, Random Forest typically ranks among the best algorithms in prediction performance. It is very important to note that no parameter tuning and/or descriptor selection was used with Random Forest in the results reported here, illustrating that Random Forest is a powerful tool even when used "off-the-shelf".

5. OPTIMAL PARAMETER AND DESCRIPTOR SELECTION

Most QSAR modeling tools require at least a moderate amount of parameter tuning to optimize performance. Performance of some tools can be significantly impacted if irrelevant descriptors are not removed prior to training. In this section, we demonstrate that Random Forest performs relatively well "off the shelf", although we advise to always investigate parameter and descriptor selection optimization, simultaneously if possible.

Parameter Tuning. Random Forest effectively has only one tuning parameter, m_{try}. This parameter is the number of descriptors randomly sampled as candidates for splitting at each node during tree induction. It can range from 1 to p, the total number of descriptors available. The latter case is equivalent to bagging. Choosing $m_{\mathrm{try}} < p$ is expected to improve performance over bagging, as discussed in section 2. On the other hand, if $m_{\mathrm{try}} = 1$, then the trees are essentially making random splits, and the only optimization is the selection of splitting point for the chosen descriptor. One would expect the performance of the ensemble to be suboptimal, unless all descriptors are of equal importance. (The latter case is illustrated by the twonorm simulated data discussed in the Appendix.) In general, m_{try} can be chosen to be some function of p. The default values of m_{try} ($p^{1/2}$ for classification and $p/3$ for regression) were chosen based on empirical experiments, and the performance of Random Forest seems to change very little over a wide range of values, except near the extremes, $m_{\mathrm{try}} = 1$ or p.

We show evidence supporting the above claim with the P-gp data. We ran 50 replications of 5-fold cross-validation to assess the error rate for a range of m_{try} values, including the default. Figure 1 shows boxplots of these error rates. The default value of m_{try} is $\sqrt{1522} \approx 39$ in this case. The plot shows that up to 5% improvement from the bagging case ($m_{\mathrm{try}} = 1522$) by reducing m_{try}. However, the error rate increased to above 30% at $m_{\mathrm{try}} = 1$ (not shown in the figure).

Descriptor Selection. Decision Tree's performance is generally not sensitive to the presence of irrelevant descriptors, since descriptor selection is intrinsic to the tree growing process ("embedded variable selection"[32]). Ensembles of trees should be even more capable of avoiding the influence of irrelevant descriptors. Therefore, we do not expect Random Forest to gain much in accuracy if descriptor reduction is implemented. As an illustration, again consider the P-gp data. We implemented the following descriptor reduction algorithm:

(1) Partition the data for k-fold cross-validation (CV). (We used $k = 5$.)

(2) On each CV training set, train a model on all descriptors and use the descriptor importance measure to rank them. Record the CV test set predictions.

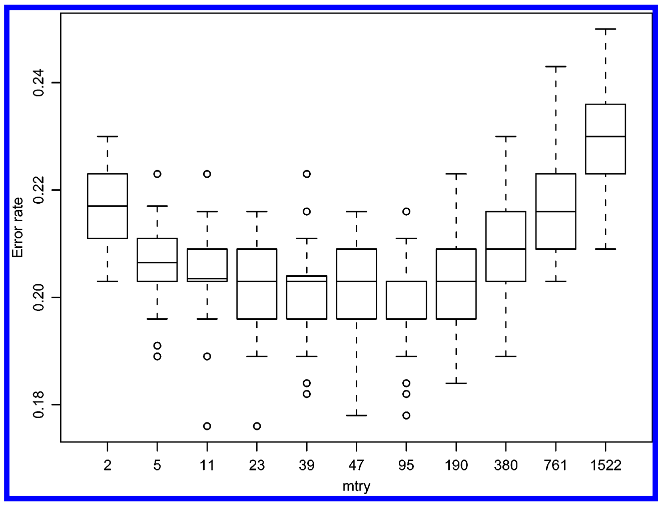

Figure 1. Boxplots of 50 5-fold cross-validation test error rates at various values of m_{try} for the P-gp data set. Horizontal lines inside the boxes are the median error rates. The plot suggests that m_{try} is optimal near 39, the default value, and that the performance is similar for values ranging from 11 to 190.

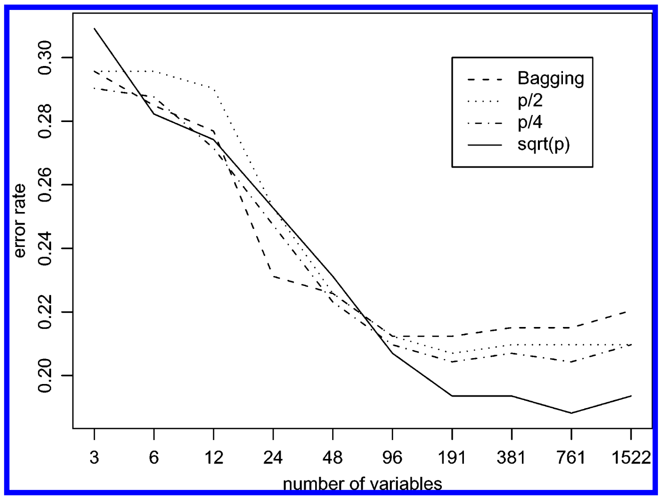

Figure 2. Median cross-validation test error rates at each step of halving the number of important descriptors, using different m_{try} functions, for the P-gp data set. Line segments connect the medians of 20 5-fold CV error rates. The plot suggests that the default m_{try}, $p^{1/2}$, performs best, and that one can reduce to about 191 important descriptors without degrading prediction performance.

(3) Use the descriptor ranking to remove the least important half of the descriptors and retrain the model, predicting the CV test set. Repeat removal of half of the descriptors until there are two left.

(4) Aggregate results from all k CV partitions and compute the error rate (or MSE) at each step of descriptor reduction.

(5) Replicate steps 1−4 10−50 times to "smooth out" the variability. The median error rates for the 50 replications, with medians connected by line segments, are shown in Figure 2, for various choices of m_{try}. The four choices of m_{try} that we examined here are $m_{\text{try}} = p$ (1522), $p/2$ (761), $p/4$ (380), and $p^{1/2}$ (39).

It can be seen that the median error rate remains roughly constant until the number of descriptors is reduced to about 191, after which the performance begins to degrade. There is no evidence that performance actually improves with descriptor selection. Also, the default m_{try} has the best performance. We observe similar results with the other data sets used in this study: when there is a difference between

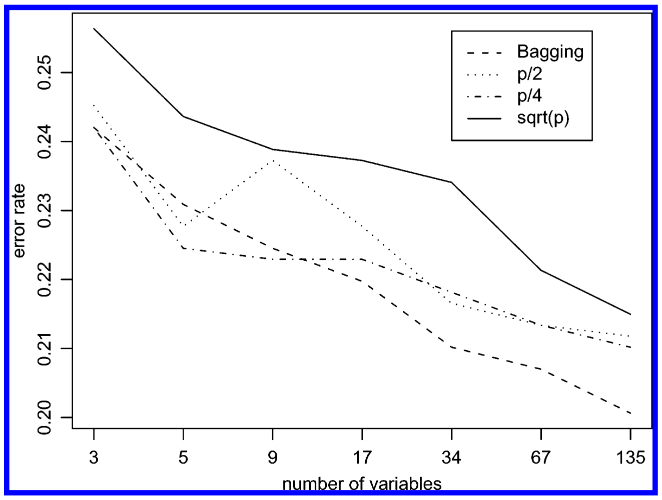

Figure 3. Median cross-validation test error rates at each step of halving the number of important descriptors, using different m_{try} functions, for the COX-2 classification data set. Line segments connect the medians of 20 5-fold CV error rates. The plot suggests that bagging ($m_{\text{try}} = p$) performs best, but that no descriptor reduction should be attempted.

the performance of various m_{try}s, the default one is usually among the best, and bagging is the worst. In some cases, there is no performance difference between any of the choices. Moreover, usually the performance stays flat or begins to degrade when descriptors are removed, but we usually do not see an actual improvement in performance with descriptor selection. The exceptional case is the COX-2 classification data. In this case, $m_{\text{try}} = p$ (bagging) is the best performer, as can be seen in Figure 3, although descriptor reduction only degrades performance.

Other Parameters. There are two other parameters in Random Forest: number of trees and minimum node size. The number of trees should only be chosen to be sufficiently large so that the OOB error has stabilized. In many cases, 500 trees are sufficient (more are needed if descriptor importance or molecular proximity is desired). There is no penalty for having "too many" trees, other than waste in computational resources, in contrast to other algorithms which require a stopping rule. To illustrate this point, we made a random split of the estrogen data into training (2/3) and test (1/3) sets and compared the OOB error rate with the independent test set error rate, for Random Forest as the number of trees increases; see Figure 4. The plot shows that the OOB error rate tracks the test set error rate fairly closely, once there are a sufficient number of trees (around 100). The use of the OOB error rate to approximate the test error rate will be discussed further in the next section. Figure 4 also shows an interesting phenomenon which is characteristic of Random Forest: the test and OOB error rates do *not* increase after the training error reaches zero; instead they converge to their "asymptotic" values, which is close to their minimum. The situation of the test error *increasing* after the training error reaches zero is often called "overfitting". In this sense, one can say that Random Forest "does not overfit".

Another parameter, minimum node size, determines the minimum size of nodes below which no split will be attempted. This parameter has some effect on the size of the trees grown. For classification, the default is 1, ensuring that trees are grown to their maximum size. (There is no need to change this except to speed up the computation for

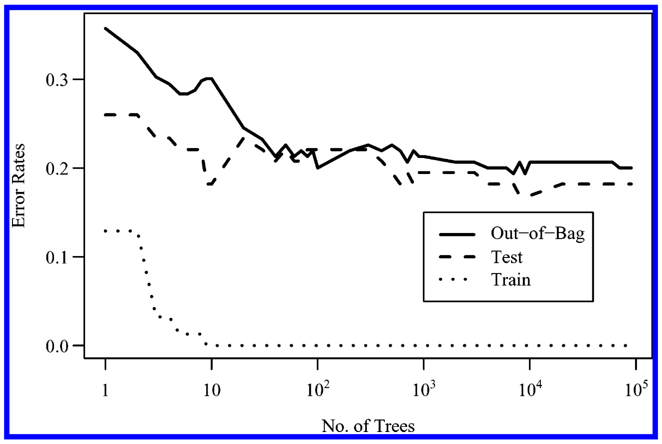

Figure 4. Comparison of training, out-of-bag, and independent test set error rates for Random Forest on Estrogen data, as the number of trees increases. The plot illustrates that the OOB error rate tracks the test error rate fairly well once the number of trees is sufficiently large. The plot also illustrates the lack of overfitting once the training error reaches zero.

large data sets.) For regression, the default is 5. The latter is arguably another tuning parameter, but we have found that the performance of Random Forest is relatively insensitive to changes in it (as long as it is relatively small). If the data are severely unbalanced, or there are differential costs of misclassification, etc., there may be a need to add additional features requiring parameter tuning to the basic Random Forest algorithm.

In conclusion, Random Forest can often be used "off the shelf", without expending much effort on parameter tuning or descriptor selection. However, it is still important for users to investigate the sensitivity of Random Forest to changes in m_{try} or descriptor selection, as illustrated in the COX-2 classification data, where $m_{\mathrm{try}} = p$ turned out to be the best choice. For further discussion of descriptor selection with Random Forest, see ref 33.

6. USING ADDITIONAL FEATURES OF RANDOM FOREST

In this section we illustrate the use of the Random Forest features provided along with prediction: out-of-bag predictions, descriptor importance, and a measure of molecular proximity. These features were all introduced in section 2.

Out-of-Bag Performance Assessment. The out-of-bag (OOB) estimate of prediction accuracy can be obtained with insignificant additional computation, yielding results comparable to those of the more computationally burdensome *k*-fold cross-validation. To demonstrate this claim, Table 7 compares OOB and 5-fold CV accuracy rates for classification and root-mean-squared errors for regression. All of the OOB estimates are calculated as was explained in section 2, using the entire training set. The OOB accuracy rate is 1 − $\mathrm{ER}^{\mathrm{OOB}}$. We did not use OOB estimates of performance in section 4 because we want to make a fair comparison with other algorithms that lack this feature. Also, the OOB performance estimate should *not* be used in any descriptor reduction procedure, such as that described in section 5, because *severe* overfitting will occur, as discussed in ref 33.

Descriptor Importance. We demonstrate the use of Random Forest's descriptor importance on the COX-2 regression data set. Since the original authors performed

Table 7. Comparison of Random Forest Performance Estimates Based on Out-of-Bag (OOB) Predictions and 5-Fold Cross-Validation (CV) for All Data Sets

data	OOB	CV
Classification Accuracy		
BBB	0.834	0.801
estrogen	0.836	0.828
P-gp	0.817	0.812
MDRR	0.828	0.822
COX-2	0.803	0.777
Regression RMSE		
dopamine	0.50	0.51
COX-2	0.81	0.84

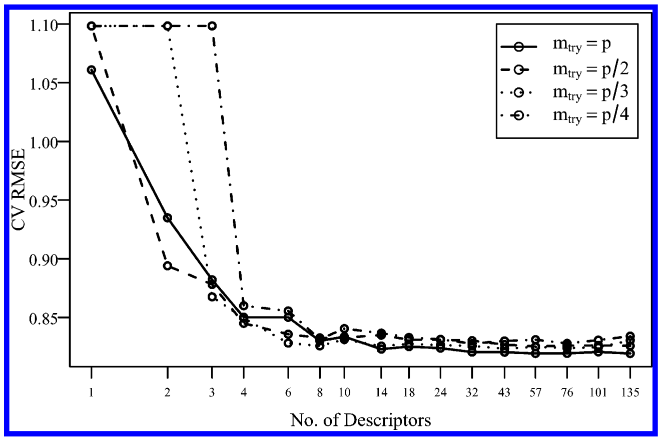

Figure 5. Median cross-validation root-mean-squared error (RMSE) at each step of reducing the number of important descriptors, using different m_{try} functions, for the COX-2 regression data set. Line segments connect the medians of 10 5-fold CV error rates. The plot suggests that there is not much difference between m_{try}s and that reducing to 8 important descriptors will not significantly degrade prediction performance.

descriptor selection using a genetic algorithm and multiple linear regression (GA-MLR), we can compare the results of their descriptor selection and ours. In the regression case, Random Forest, performing descriptor selection and m_{try} optimization as shown in section 5 results in a choice of about 8 descriptors in the final model (Figure 5). Kauffman and Jurs[3] also used 8 descriptors in their model; only one of these (PND-5) is in common with the Random Forest list. It is interesting to note that Kauffman and Jurs' procedure deliberately avoided selecting descriptors that are correlated with each other. However, Random Forest has no trouble dealing with correlated descriptors, and such descriptors are expected to have roughly similar importance measures associated with them. In the list of Random Forest descriptors, MDE−23 and MDE−24 have a pairwise correlation of 0.97, and their importances (percent increase in MSE) are 1.77 and 1.22, respectively. Comparing the correlation of Random Forest descriptors and the descriptors unique to the Kauffman−Jurs model, 2SP3−1 in their list is correlated with both MDE−23 and MDE−24 from Random Forest's list, with correlations of 0.90 and 0.93, respectively. ESUM in Random Forest's list has a correlation of 1 with EAVE−2 in the Kauffman−Jurs list. However, the other descriptors in the two lists are not strongly correlated with each other. This suggests that there may not be one unique set of optimal descriptors.

In summary, Random Forest and the Kauffman−Jurs procedures can perform roughly equally well but using

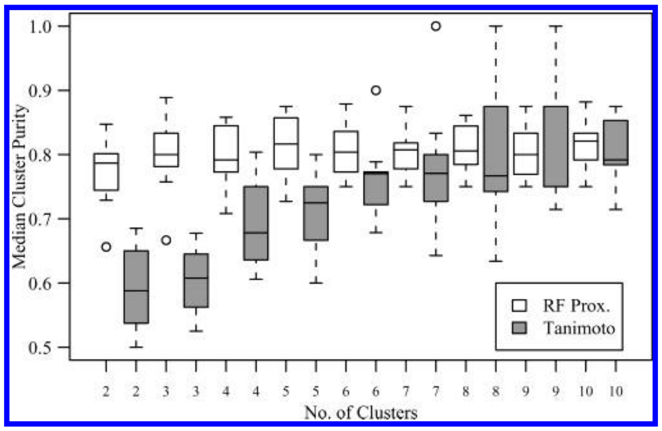

Figure 6. Cluster purity vs number of clusters for clusterings based on Random Forest proximity and Tanimoto distance for the P-gp data set. Results are boxplots for 10 random splits of the data into training and test sets, where clustering is done on test data only. Cluster purity is defined in the text. Horizontal lines inside the boxes are the median purities. The plot suggests that Random Forest proximity yields purer clusters than Tanimoto distance does when the number of clusters is small (and hence least likely to be homogeneous)

mostly different sets of descriptors. These results underscore potential problems with the interpretation of the molecular mechanism of action (MOA). It is quite possible that different sets of descriptors are pointing to the same MOA, or it could be a purely mathematical fact that in the multivariate optimization problem of descriptor selection, there may be a number of equivalent solutions and selection of the one “right” solution requires subject matter expertise.

Molecular Proximity. We use the P-gp data set as an example to demonstrate clustering molecules based on molecular proximity from Random Forest. We randomly split the data into two subsets, one for training and the other for testing (containing 30 substrates and 30 nonsubstrates). We train Random Forest on the training subset, predict the test molecules, and compute the proximities among the test molecules. We then cluster the test molecules using these proximities by the Ward hierarchical clustering algorithm with different numbers of clusters, *k*. In parallel, we also cluster the test molecules using Tanimoto similarity[34] and the same clustering algorithm. We compute the purities of each cluster in these two clusterings. The purity of each cluster is the proportion of majority class molecules in the cluster, either substrate or nonsubstrate. The purity of a clustering consisting of *k* clusters is defined as the median over the *k* cluster purities. The whole procedure is repeated 10 times; the resulting boxplots are shown in Figure 6. For small numbers of clusters, the Random Forest proximity clusters are more pure than those based on Tanimoto similarity. This result may serve as evidence that Random Forest’s proximity indeed tends to take into account descriptors that are the most important for discriminating between classes. The Tanimoto similarity, on the other hand, does not take into account the different discriminating power of descriptors and treats all descriptors equally, which results in lower purity when the number of clusters is small. However, when the number of clusters grows larger, the difference between the two procedures diminishes. This could be explained by the tendency of smaller clusters to be more homogeneous in descriptor space, where the “paradigm” that “very similar” molecules tend to have similar activities holds.

7. DISCUSSION

Random Forest is a powerful tool for QSAR modeling, since it offers prediction performance among the best as well as additional features (an out-of-bag performance estimate that can be used in place of cross-validation, a measure of descriptor importance that can aid in descriptor selection, and an intrinsic measure of molecular proximity). Random Forest seems to perform well even without parameter tuning and descriptor selection. Random Forest preserves most of the appealing features of Decision Trees, such as the ability to handle a large number of different types of descriptors simultaneously, handling of redundant/irrelevant descriptors, incorporating interactions and multiple mechanisms of action, and ability to handle both classification and regression.

On the other hand, unlike Decision Tree, Random Forest does not produce an explicit model; instead, the “important” descriptors and their relation with biological activity are hidden inside a “black box”.[19] The interpretation of Decision Tree may actually be spurious because (1) the tree could be unstable, i.e., small changes in the training data could result in completely different tree structures; and (2) Decision Tree is usually a relatively low accuracy algorithm. This “makes interpretation somewhat precarious” (p 274 in ref. 18). Nonetheless, in those cases where the tree is stable and accurate, the interpretability of the model is an obvious advantage. In Random Forest, the measure of descriptor importance may, in principle, aid in the interpretation of the data, especially in combination with the partial dependence plots[18] and a significance test for descriptor importance.[35] However, as we showed in the case of the COX-2 regression example, there may be more than one set of descriptors that generate equally good performance. This suggests that a valid interpretation of descriptor importance requires subject matter knowledge. The artistry here is in choosing a set of descriptors that provides good performance as well as ease of interpretation.

The Random Forest intrinsic proximity measure may provide additional utility, for example, in uncovering false negatives in high throughput screening.[35] In this example, one builds a QSAR model on *confirmation* data for molecules selected from the primary screen. This model is then used to predict all molecules that were not selected from the primary screen. Molecules with high predicted probabilities of being active are thought to be false negatives and are recommended for retest. To reduce the burden of retest, these molecules can be clustered using Random Forest proximity, and only representatives from each cluster are considered. Also, sometimes researchers are interested only in those false negatives that are novel, i.e., have low similarities with compounds already known to be active.

As further evidence of Random Forest’s utility, we have had several successful in-house applications of Random Forest for QSAR modeling of high throughput screening data and in vitro ADME assay properties. We hope that this paper stimulates other researchers to consider applying Random Forest to challenging classification and regression problems.

Random Forest Software. Software for Random Forest is available in Fortran from Breiman and Cutler.[36] The R language software interfacing with this Fortran code[27] is available on CRAN.[37] (The R packages rpart for decision tree and pls.pcr for PLS are also available there.) We are in the process of developing Matlab software interfacing the same Random Forest Fortran code; this version was used to obtain the results discussed in the Appendix. The code will be made available from the authors.

ACKNOWLEDGMENT

We are grateful, above all, to Leo Breiman for discussions and consultations regarding ensemble learning and Random Forest over the past few years. We would also like to thank the following individuals for providing data and comments on their work: Peter Grootenhuis, Michelle Lamb, Peter Jurs, Gregory Kauffman, Weida Tong, Joanne Yeh, and David Meyer. We are in debt to Vladimir Maiorov for assisting in software development and Ting Wang for the development of Matlab Random Forest software. We would also like to thank Raymond Evers, Dylan Hartley, and Leslie Romanyshyn for discussions on the context of QSAR modeling in ADME. We would also like to thank Tom Dietterich for valuable discussions on ensemble learning and bias-variance decomposition.

APPENDIX: RANDOM FOREST INSIGHTS

In this appendix we offer some explanation of how Random Forest works, considering the classification problem. Suppose that a feature vector $X \in |\mathrm{R}^p$ and associated class label $Y \in \{1, 2, ..., n_{\mathrm{class}}\}$ are two random variables with joint probability distribution $p(x, y)$. The goal is to find a classifier such that given a vector X, unseen at the training, the classifier predicts its class label $\hat{Y}(X)$. The quality of this prediction is measured by the expectation of some loss function. The expectation is taken over all X and Y with respect to $p(x, y)$. For simplicity, we will consider the 0−1 loss for label misclassification as the loss function and its expectation, the probability of misclassification:

$$\mathrm{ER} = E_{X,Y} I(\hat{Y} \neq Y) = \Pr\{\hat{Y} \neq Y\}$$

Here $E_{X,Y}$ is expectation with respect to $p(x, y)$ and Pr is probability.

When the distribution $p(x, y)$ is known, one can build the optimal, Bayes classifier, i.e., the classifier with the minimum error rate.[38] However, in most practical applications the distribution $p(x, y)$ is unknown; instead only the training data $D = \{(X_1, Y_1), ...,(X_n, Y_n)\}$, randomly drawn from this distribution is available. The goal then is to create a classifier that makes a prediction $\hat{Y}(X)$ by minimizing some estimate of the error rate

$$\mathrm{ER} \approx E_{X,Y,D}(I(\hat{Y}(X, D) \neq Y)) = \Pr\{\hat{Y}(X, D) \neq Y\}$$

where the expectation is taken over all X, Y and training data D.

Breiman[39] argues that some classifiers and regression models that are created as a result of minimization of estimated expected losses are unstable. The instability means that small changes in the training data and/or in the optimization procedure may lead to large changes in the predictor structure and its output. Classification and regression trees are unstable predictors, which is the main factor contributing to their relatively low accuracy. The performance of unstable predictors can be improved, in some cases quite dramatically, by combining predictions from multiple instances obtained as a result of deliberate perturbation of the training data and/or the optimization procedure. Random Forest is one such ensemble predictor where each bootstrap training set is a "perturbed" version of the training data, and the optimization procedure is perturbed by the random split selection. Aggregation is done by majority vote (in classification) or averaging over individual tree predictions (in regression).

To explain why the procedure works, we will use the so-called bias-variance decomposition of the error rate, ER. This decomposition in regression is a standard and widely used procedure for characterizing various aspects of a regression model.[18] In classification, the bias-variance decomposition is not that straightforward nor unique. Nonetheless its importance in the understanding of classification methods is crucial, and a number of decompositions have been proposed in the literature.[39−42] In the example below, we will use the decomposition of Domingos.[41]

In addition to the classifier prediction $\hat{Y}(X, D)$ whose error rate decomposition we are interested in, Domingos considers two other predictions. The first is $Y^*(X)$ which is the prediction of the optimal, Bayes classifier, whose error rate is minimum. The second is called the main prediction in Domingos, but we will call it the aggregated, or average prediction

$$\hat{Y}_A(X) = \mathrm{argmax}\{V_1(X), V_2(X), ..., V_{n_{\mathrm{class}}}(X)\}$$

where $V_i(X) = \Pr_D\{\hat{Y}(X, D) = i\}$ is the probability that given a feature vector X, a classifier trained on the randomly selected training data D, casts a vote $\hat{Y}(X, D)$ for class i. Hence, $\hat{Y}_A(X)$ is the class label most frequently voted for by the replicas of the classifier trained on different data sets. With these definitions, Domingos showed the following

$$\mathrm{ER} = E_X[h_1(X)\cdot\mathrm{Noise}(X) + \mathrm{Bias}(X) + h_2(X)\cdot\mathrm{Var}(X)]$$

where

$$Noise(X) = I(Y^*(X) \neq Y)$$

$$Bias(X) = I(\hat{Y}_A(X) \neq Y^*(X))$$

$$Var(X) = \Pr_D\{\hat{Y}(X, D) \neq \hat{Y}_A(X)\}$$

$$h_1(X) = 2\cdot\Pr_D\{\hat{Y}(X, D) = Y^*(X)\} - 1;\ \text{and}$$

$$h_2(X) = 1 \text{ if } \mathrm{Bias}(X) = 0, -1 \text{ otherwise}$$

The meaning of the elements in this decomposition is the following. Noise(X) is an "irreducible" component of the error rate, which is independent of the classifier. Bias(X) reflects how the average prediction of the feature vector X is different from the optimal prediction. Var(X) shows the classifier's "variability", i.e., how much the prediction by each replica of the classifier would disagree with the average of such predictions. Coefficients h_1 and h_2 reflect interactions between all three components. The presence of these interactions is fundamental for classification as opposed to regression. Specifically, an increase in the variance in the presence of the bias, Bias(X) = 1, decreases the error rate ($h_1(x) = -1$). The opposite, an increase in the error rate with the increase in the variance, happens when the bias is zero.

Similarly, the error rate decreases when Noise(X) = 1 if simultaneously $\Pr_D\{\hat{Y}(X, D) = Y^*(X)\} < 0.5$, i.e., the prediction of the majority of classifiers disagrees with the optimal prediction. Taking expectation (averaging over all X) in the decomposition yields

$$\mathrm{ER} = \mathrm{Noise} + \mathrm{Bias} + \mathrm{Var}$$

where Noise = $E_X(h_1(x)\cdot\mathrm{Noise}(x))$; Bias = $E_X(\mathrm{Bias}(X))$; Var = $E_X(h_2(X)\cdot\mathrm{Var}(x))$. In addition to the variance, Domingos considers its two components, "biased", $\mathrm{Var_B}$, and "unbiased", $\mathrm{Var_U}$, such that $\mathrm{Var} = \mathrm{Var_U} - \mathrm{Var_B}$ with $\mathrm{Var_U} = E_X\{I(\mathrm{Bias}(x) = 0)\cdot\mathrm{Var}(x)\}$ and $\mathrm{Var_B} = E_X\{(1 - I(\mathrm{Bias}(x) = 0))\cdot\mathrm{Var}(x)\}$.

Breiman, using a different decomposition but with qualitatively similar components, demonstrates that "procedures like CART have high variance, but they are 'on average, right', that is, they are largely unbiased".[39] An ensemble of trees reduces the variance leaving the already low bias practically unaffected. The net effect is a reduction in the error rate.

As an example, we consider an example of a two-class classification problem with simulated data from refs. 39 and 8, called "twonorm". In the example, the data is from a 20-dimensional multivariate Gaussian with identity covariance matrix. Class 1 has mean A and class 2 has mean $-A$, where $A = (a_1, a_2, ..., a_{20})$, $a_1 = a_2 = ... = a_{20} = 2/20^{1/2}$.

Both classes have the same number of samples. For the training we generated 100 data sets $D_1, D_2, ..., D_{100}$ with $n = 300$ samples in each. Using these 100 data sets we created groups of 100 classifiers with each group being a particular classifier type, e.g. Random Forest with a particular value of m_{try}. All classifiers were then used to predict a large, fixed test set of 18 000 compounds; error rates and components of the bias-variance decomposition were calculated. Breiman[39] notes that this problem is difficult for classification trees since "the optimal separating surface is an oblique plane [which] is hard to approximate by the multidimensional rectangles [whose sides are parallel to the axes] used in CART".

Figure 7 shows the error rates for a single pruned tree and for Random Forest with two different values of m_{try}. The first value, $m_{\mathrm{try}} = 20$, corresponds to the case of bagging, when the tree growing algorithm selects the best split among all 20 variables. The second value, $m_{\mathrm{try}} = 1$, is a completely opposite case, when the splitting is done on just one randomly selected feature. Each boxplot represents the distribution of 100 error rates. The effect of tree aggregation is evident. When the number of trees in Random Forest increases, the median error rate significantly decreases from about 23% (bagging) and 24% ($m_{\mathrm{try}} = 1$) to 6.2% and 3.2%, respectively. Note also that Random Forest with random feature selection ($m_{\mathrm{try}} = 1$) scores favorably to bagging indicating that "perturbation" of the optimization procedure plays a role in the error rate reduction. Finally, the error rate 3.2% is very close to the Bayes error rate equal to 2.4% in this example. Figure 8 shows on a log−log scale the bias and variance components of the error rate as functions of the number of trees in Random Forest. To avoid overloading the figure, we omitted the Noise component whose magnitude is close to that of the bias or smaller. Also, $\mathrm{Var_U}$ is much larger than $\mathrm{Var_B}$ and we do not show them separately. Also shown are

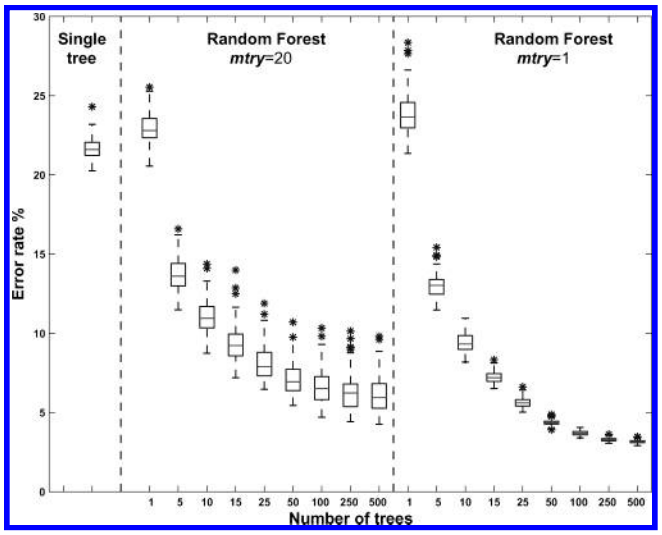

Figure 7. Boxplots of error rates for 100 replications of classification of twonorm data (simulation). Results are shown for a single (pruned) tree, Random Forest with $m_{\mathrm{try}} = 20$ (equivalent to bagging) and $m_{\mathrm{try}} = 1$. Boxplots of Random Forest results are shown for different numbers of trees. Horizontal lines inside the boxes are the median error rates. The plot shows the effect of an ensemble of trees on the reduction of the error rate. Moreover, the error rate of Random Forest with $m_{\mathrm{try}} = 1$ is progressively lower than that for bagging, when the number of trees exceeds about 50.

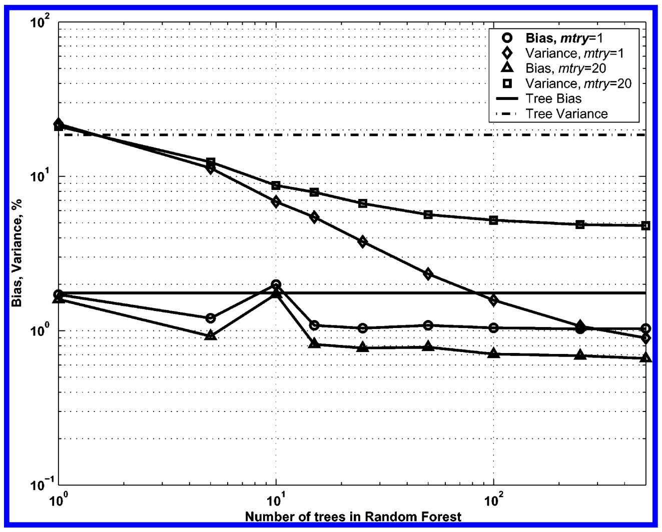

Figure 8. Bias-variance decomposition for Random Forest with $m_{\mathrm{try}} = 20$ (equivalent to bagging) and $m_{\mathrm{try}} = 1$, and a single (pruned) tree, for the twonorm simulated data. The bias and variance components (in percent) are shown for different numbers of trees in the case of Random Forest; the plot is on the log−log scale (base 10 log). The bias curves indicate that ensemble has a relatively small effect on the bias reduction. On the other hand, the variance curves show a significant effect on the reduction of variance as the number of trees increases. The reduction in variance is more significant for $m_{\mathrm{try}} = 1$ than for bagging ($m_{\mathrm{try}} = 20$).

the bias and variance for a single pruned tree. As one can see, an increase in the number of trees has a very small effect on the bias. Its value is also quite small for a single tree. On the contrary, the variance, which is the largest component of the error rate, is reduced quite dramatically, and more so when $m_{\mathrm{try}} = 1$ versus bagging. The bias and variance behaviors in this example support the commonly held view that Random Forest is primarily a variance reduction technique and as such will be effective in applications where variance is a dominant component of the error rate. This explains why Random Forest outperforms a single tree, who clearly suffers from high variance (Figure 7).

This also explains why Random Forest uses trees grown to the maximum size. Since Random Forest is a variance reduction procedure, with a small effect on the bias, it is advantageous to aggregate classifiers that start out with low bias, even at the expense of having large variance. Although not illustrated in this example, unpruned trees tend to have lower bias and higher variance than pruned trees.

Finally, we note that we experimented with several values of m_{try} and found that $m_{\mathrm{try}} = 1$ gives the best performance of Random Forest in this example (results with other values of m_{try} are not presented here). However, this will not be the case in general, as discussed in section 5.

Supporting Information Available: The R code used to generate most of the results in this paper is available as a text file, as are data files for the data sets for which we generated our own descriptors (P-gp, MDRR, and Dopamine). This material is available free of charge via the Internet at http://pubs.acs.org.

REFERENCES AND NOTES

(1) Rusinko, A., III; Farmen, M. W.; Lambert, C. G.; Brown, P. L.; Young, S. S. Analysis of a large structure/biological activity data set using recursive partitioning. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 1017−1026.

(2) Doniger, S.; Hofmann, T.; Yeh, J. Predicting CNS permeability of drug molecules: comparison of neural network and support vector machine algorithms. *J. Comput. Biol.* **2002**, *9*, 849−864.

(3) Kauffman, G. W.; Jurs, P. C. QSAR and *k*-nearest neighbor classification analysis of selective cyclooxygenase-2 inhibitors using topologically based numerical descriptors. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1553−1560.

(4) Sheridan, R. P.; Nachbar, R. B.; Bush, B. L. Extending the trend vector: the trend matrix and sample-based partial least squares. *J. Comput.-Aided Mol. Des.* **1994**, *8*, 323−340.

(5) Bakken, G. A.; Jurs, P. C. Classification of multidrug-resistance reversal agents using structure-based descriptors and linear discriminant analysis. *J. Med. Chem.* **2000**, *43*, 4534−4541.

(6) Tong, W.; Hong, H.; Fang, H.; Xie, Q.; Perkins, R. Decision forest: combining the predictions of multiple independent decision tree models. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 525−531.

(7) Dietterich, T. G. Ensemble learning. In *The Handbook of Brain Theory and Neural Networks*, 2nd ed.; Arbib, M. A., Ed.; The MIT Press: Cambridge, 2002.

(8) Breiman, L. Random forests. *Machine Learning* **2001**, *45*, 5−32.

(9) *The Mathematics of Generalization*; Wolpert, D. H., Ed.; Addison-Wesley: Reading, 1995.

(10) Breiman, L. Bagging predictors. *Machine Learning* **1996**, *24*, 123−140.

(11) Breiman, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J. *Classification and Regression Trees*; Chapman & Hall/CRC: Boca Raton, 1984.

(12) Freund, Y.; Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.* **1997**, *55*, 119−139.

(13) Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189−1202.

(14) Meyer, D.; Leisch, F.; Hornik, K. The support vector machine under test. *Neurocomputing* **2003**, submitted.

(15) Hawkins, D. M.; Musser, B. J. One tree or a forest? Alternative dendrographic models. *Comput. Sci. Stat.* **1999**, *30*, 534−542.

(16) Hawkins, D. M.; Basak, S. C.; Mills, D. Assessing model fit by cross-validation. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 579−586.

(17) Breiman, L. *Out-of-bag estimation*; Technical Report; Department of Statistics: UC Berkeley, 1996.

(18) Hastie, T.; Tibshirani, R.; Friedman, J. H. *The Elements of Statistical Learning*; Springer-Verlag: New York, 2001.

(19) Breiman, L. Using models to infer mechanisms, IMS Wald Lecture 2, 2002, http://oz.berkeley.edu/users/breiman/wald2002-2.pdf.

(20) Schapire, R. E.; Freund, Y.; Bartlett, P.; Lee, W. S. Boosting the margin: A new explanation for the effectiveness of voting methods. *Ann. Stat.* **1998**, *26*, 1651−1686.

(21) Brown, R. D.; Martin, Y. C. Use of structure−activity data to compare structure-based clustering methods and descriptors for use in compound selection. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 572−584.

(22) Penzotti, J. E.; Lamb, M. L.; Evensen, E.; Grootenhuis, P. D. J. A computational ensemble pharmacophore model for identifying substrates of P-glycoprotein. *J. Med. Chem.* **2002**, *45*, 1737−1740.

(23) Carhart, R. E.; Smith, D. H.; Venkataraghavan, R. Atom pairs as molecular features in structure−activity studies: definition and applications. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 64−73.

(24) Klopman, G.; Shi, L. M.; Ramu, A. Quantitative structure−activity relationship of multidrug resistance reversal agents. *Mol. Pharmacol.* **1997**, *52*, 323−334.

(25) *DRAGON*; Milano Chemometrics, http://www.disat.unimib.it/chm/Dragon.htm.

(26) Gilligan, P. J.; Cain, G. A.; Christos, T. E.; Cook, L.; Drummond, S.; Johnson, A. L.; Kergaye, A. A.; McElroy, J. F.; Rohrbach, K. W.; Schmidt, W. K.; Tam, S. W. Novel piperidine σ receptor ligands as potential antipsychotic drugs. *J. Med. Chem.* **1992**, *35*, 4344−4361.

(27) Liaw, A.; Wiener, M. Classification and regression by randomForest. *R News* **2002**, *2/3*, 18−22.

(28) Therneau, T. M.; Atkinson, E. J. *An introduction to recursive partitioning using the RPART routines*; Technical Report; Department of Health Sciences Research: Mayo Clinic, 1997.

(29) *pls.pcr;* http://cran.r-project.org/src/contrib/PACKAGES.html#pls.pcr.

(30) Nguyen, D. V.; Rocke, D. M. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics* **2002**, *18*, 39−50.

(31) Barker, M.; Rayens, W. Partial least squares for discrimination. *J. Chemometrics* **2003**, *17*, 166−173.

(32) Guyon, I.; Eliseeff, A. An introduction to variable and feature selection. *J. Machine Learning Res.* **2003**, *3*, 1157−1182.

(33) Svetnik, V.; Liaw, A.; Tong, C. Variable selection in random forest with application to quantitative structure−activity relationship. In *Proceedings of the 7th Course on Ensemble Methods for Learning Machines*; Intrator, N., Masulli, F., Eds.; Springer-Verlag: Berlin, 2004; submitted.

(34) Willett, P. Chemical similarity searching. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 983−996.

(35) Liaw, A.; Svetnik, V. *Application of random forest to QSAR modeling of HTS data, Presented at the 26th Annual Midwest Biopharmaceutical Statistics Workshop*; Muncie, IN, 2003.

(36) Breiman, L.; Cutler, A. In *Manual-Setting up, using, and understanding Random Forests*; v4.0, 2003, http://www.stat.berkeley.edu/users/breiman/rf.html.

(37) The Comprehensive R Archive Network, http://cran.r-project.org.

(38) Ripley, B. D. *Pattern Recognition and Neural Networks*; Cambridge University Press: Cambridge, 1996.

(39) Breiman, L. Arcing classifiers. *Ann. Stat.* **1998**, *26*, 801−849.

(40) Friedman, J. H. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining Knowledge Discovery* **1997**, *1*, 55−77.

(41) Domingos, P. A unified bias-variance decomposition for zero-one and squared loss. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*; AAAI Press: Austin, 2000.

(42) Valentini, G.; Dietterich, T. G. Bias-variance analysis and ensembles of SVM. In *Third International Workshop on Multiple Classifier Systems*; Kittler, J., Roli, F., Eds.; Springer: New York, 2002.

CI034160G