Flexible Web Service Infrastructure for the Development and Deployment of Predictive Models

Rajarshi Guha*

School of Informatics, Indiana University, Bloomington, Indiana 47406

Received June 1, 2007

The development of predictive statistical models is a common task in the field of drug design. The process of developing such models involves two main steps: building the model and then deploying the model. Traditionally such models have been deployed using Web page interfaces. This approach restricts the user to using the specified Web page, and using the model in other ways can be cumbersome. In this paper we present a flexible and generalizable approach to the deployment of predictive models, based on a Web service infrastructure using R. The infrastructure described allows one to access the functionality of these models using a variety of approaches ranging from Web pages to workflow tools. We highlight the advantages of this infrastructure by developing and subsequently deploying random forest models for two data sets.

1. INTRODUCTION

The field of Quantitative Structure−Activity Relationship (QSAR) modeling is intimately linked with the development of statistical methods. Initially QSAR models focused on the use of multiple linear regression, though in recent times a wide variety of more complex and sometimes more accurate methods such as neural networks[1] and support vector machines[2] have been employed. If one considers the literature, then it is evident that many thousands of QSAR models have been published, and an even larger number have been developed but not publicly disclosed.

It is clear that for QSAR models to be effective, two important steps are involved. First, an accurate and validated model must be developed. The process of actually developing the model should utilize the proper procedures to ensure generaliability and prevent overfitting. Second, the model must be deployed. This step is based on the assumption that the other people would like to utilize the model to predict properties for new molecules. The procedure to deploy a model can vary significantly. However, the traditional approaches that allow nonexperts to utilize the models have generally focused on some sort of Web based interface. Thus, a user would visit a predefined Web page, upload data, and obtain predictions. Though this interface is simple and easy to use it has one downside. That is, it constrains the user to use the specified Web page. In such an interface it is difficult for the user to perform tasks that were not considered by the designer. For example, bulk predictions may or may not be possible. Furthermore, if the user is working in a certain environment, the use of a Web page implies that the workflow must be altered to include a visit to the Web page. Finally, programmatic access to such Web pages can become cumbersome.

Web services provide a general solution to the problem of remote access of a variety of functionality. Essentially a Web service is a remote function that is accessed (in most cases) over the HTTP protocol. A specific protocol is used by programs that communicate with Web services and is termed the Simple Object Access Protocol (SOAP). This protocol allows clients to determine what types of data a service will require as input, what types of data it may receive from a service, and what errors may be thrown by the service. One of the main reasons that Web services are increasing in popularity[3] is their ability to expose arbitrary functionality in a distributed manner. This implies that a user of a Web service does not need to install any specific software (beyond being required to communicate with a Web service). Thus, for example, a Web service can be provided to perform database searches, perform docking calculations, or generate molecular fingerprints. In all cases, the user does not need to install or manage any software specific to database, docking, or cheminformatics.

Clearly, Web services provide an easy approach to exposing a variety of functionality. More important however, is that this approach does not constrain a user in any fashion. That is, given the link to a Web service, the user is free to access it using any language or program that supports SOAP based Web services. Thus one can design a Web page that obtains results by accessing Web services. Alternatively one can use a workflow tool such as Taverna[4] or Scitegic Pipeline Pilot and access Web service functionality directly from within a workflow. One could also write a custom command line or GUI clients. Clearly, there is no restriction in the manner in which Web services can be used.

Web services thus provide an interesting and potentially general solution to the deployment of predictive statistical models. Clearly, predictive models can be developed in a wide variety of packages such as R,[5] SPSS, SAS, and so on. Using a Web service infrastructure allows one to deploy models developed in different packages. As noted above, the user does not need to be familiar with any of the actual packages themselves, as long as the Web service is well defined. Furthermore, in addition to simply obtaining new predictions, the model provider can provide additional information such as measures of model applicability.

* Corresponding author e-mail: rguha@indiana.edu.

In the following sections we describe a Web service infrastructure for the deployment of models developed in R. We focus on the use of R due to the fact that it is open-source and thus freely available. In addition to providing access to prebuilt models, the service provides access to a wide variety of R functionality for building models ranging from linear regression to random forests. We provide a brief description of the design of the infrastructure and then show its usage by developing random forest classification models for two data sets and then deploying the models in the infrastructure. The source code for the project is freely available and can obtained from the ChembioGrid Sourceforge page (http://sourceforge.net/projects/cicc-grid).

2. DATA SETS

For the purposes of exercising the computational infrastructure we considered a number of data sets. In the interests of conciseness we only discuss two of them in this paper. The first data set consisted of 4337 molecules that had been tested for mutagenicity using the Ames test.[6] This data set had been studied by Kazius et al.[7] for the purpose of identifying toxicophores. The dependent variable for this data set was a label of *mutagen* or *nonmutagen* thus representing a classification problem with 2401 molecules classified as mutagens and 1936 molecules as nonmutagens.

This data set has been used in a variety of modeling studies that have aimed to improve mutagenicity predictions. Thus Liao et al.[8] employed these compounds as part of a training set for a support vector machine (SVM) and recursive partitioning (RP) classification models. Their results indicate that the RP model exhibited 85.2% and 83.0% correct classification for the mutagenic and nonmutagenic classes in the training set (83.4% overall). The SVM model showed similar results on the individual classes and 83.8% correct classification overall. They tested their models on a variety of toxicity and drug databases and observed percent correct classification results ranging from 80% to 84%. The authors employed a set of 66 topological descriptors[9] but did not employ a form of feature selection. This is surprising since neither the RP or SVM modeling techniques provide implicit feature selection, and thus overfitting is a possibility. Serafimova et al.[10] employed data from the National Toxicology Program (NTP) with the aim of predicting mutagenicity and deriving structural alerts.[7] They obtained a smaller set than described in Kazius et al., but, unlike the previous work, they were able to assign mechanisms of action to all the identified toxicophores.

The second data set we considered was obtained from the NCI Developmental Therapeutics Program (DTP). This data set consisted of approximately 43 000 molecules which had been tested against 60 cancer cell lines for possible anticancer activity. Not all the molecules were tested in all the cell lines, and so we considered only those molecules that had measured assay values for a given cell line. This resulted in 60 individual data sets whose sizes ranged from 25 767 to 42 723 molecules. Since the reported assay values were real-valued numbers (ranging between −4.0 and 11.3 log units), we selected a cutoff of 5.0, and molecules whose measured assay value was greater than this value were classified as *active* and those which were below the cutoff as *inactive*. This resulted in a range of class distributions for the 60 data sets we considered. The ratio of the actives to the inactives ranged from 0.15 to 0.40 indicating that in some of the data sets, there was a significant imbalance in the class distributions.

The NCI DTP assay data has been employed in a number of studies. Many of these have focused on the anticancer activity of specific classes of compounds such as quinones,[11] platinum compounds,[12] and artemisinins.[13] A number of studies have focused on general tools and models to characterize anticancer activity. One of the earliest such tools was COMPARE[14] developed by Zaharevitz et al. Huang et al.[15] combined physicochemical descriptors, binary fingerprints, and GI_{50} data to develop a global predictive model for GI_{50} values. Their model exhibited an R^2 of 0.77 and 0.67 for the training and test sets, respectively. However, their model was not cell line-specific and rather used the mean of GI_{50} over all the cell lines as the dependent variable. Ren et al.[16] employed a binary QSAR model to predict the GI_{50} values for a set of 166 anticancer compounds from the DTP. It should be noted that none of the approaches described here (with the exception of the COMPARE tool) take into account individual cell lines but rather merge the data. Our goal was to develop individual models for each cell line, thus allowing a user to obtain a *prediction panel*.

For both data sets considered here, we evaluated 166 bit MACCS keys. We did not investigate other real-valued descriptors since we were interested in identifying structural fragments that were correlated to the property in question.

3. MODEL DEVELOPMENT

Since both data sets considered here represented classification problems we chose to develop random forest[17] models. These types of models have been used in a variety of QSAR modeling studies[2,18−20] and have been shown to provide good predictive ability. More importantly, random forest models have been shown to be resistant to overfitting and also perform implicit feature selection. This is especially useful for the mutagenicity data set as it means that we do not need to perform any a priori feature selection to obtain a set reduced set of relevant features. We employed the implementation of the random forest method available in the R software package.[5] This implementation provides a number of parameters including the number of trees to use and the number of descriptors to consider at each split point. We investigated a variety of values of these two parameters using a grid search. In the case of the mutagenicity data set, we selected 700 trees and 12 descriptors to be considered at each split point. For the case of the NCI DTP data set we considered we did not observe any significant improvement over the default values.

As noted above some of the individual data sets used to build the models for the NCI DTP data were significantly imbalanced, in favor of the inactive class. Since our goal was to identify active compounds, we adopted a sampling approach that would favor the active class. Thus for each tree in a random forest model, we instructed the algorithm to use all the active compounds but randomly select a set of inactive compounds whose size was 60% of actives. We had investigated alternative numbers of inactive compounds but settled on 60% as it led to a relatively good performance over the 60 cell lines.

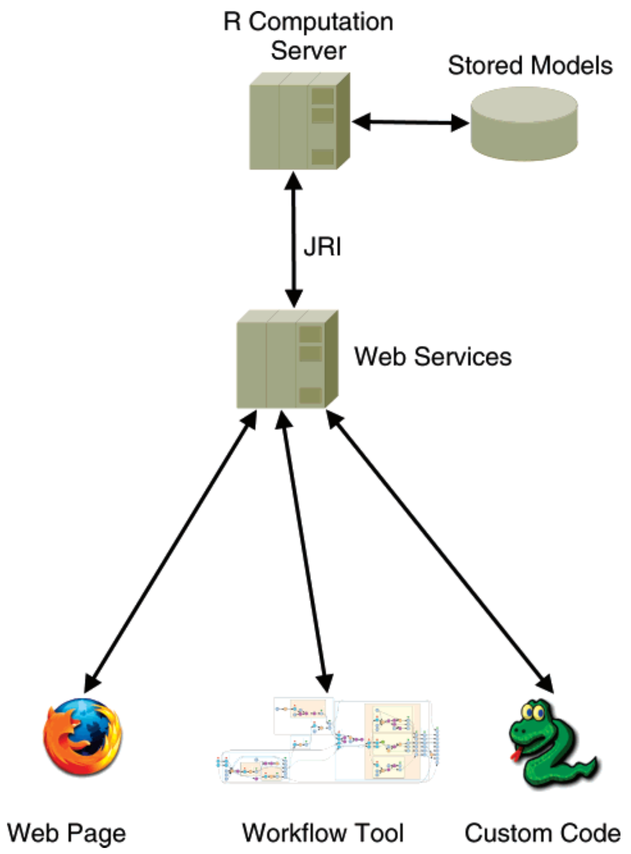


Figure 1. An overview of the R Web service infrastructure. JRI is a Java library that allows Java code to communicate with a remote R server. Currently models are stored in the filesystem rather than a database.

In addition to providing a prediction of the class membership, a random forest model also allows us to investigate the importance of the descriptors provided to it. That is, we can extract a ranking of the input descriptors based on their importance. In this context, the term "importance'' refers to the importance of the descriptor to the models predictive ability. Since the descriptors we employ are fragment descriptors, this approach allows us to identify the structural features of a molecule that are important for predicting their mutagenicity or anticancer activity class.

4. INFRASTRUCTURE

Once we have built models that exhibit suitable predictive performance, they must be deployed. The manner in which predictive models are deployed varies depending on the nature of the users of the models as well as the resources available. Traditionally, predictive models have been deployed as a Web page. As noted previously, this approach is simple but can be restrictive.

Our approach to the problem of model deployment is based on Web services, which can be considered to be a form of remote function calls. Now, given a model that has been built in R, we would like to be able to use this model directly in our infrastructure. To allow this we have developed a set of Web services that allows one to use the computational functionality of R in a distributed manner. The general infrastructure is described graphically in Figure 1. The basic design of the system involves the use of the Rserve package,[21] which allows one to run an R server on an arbitrary machine and then communicate with it via the TCP/IP protocol. We set up such a remote R server and prepared a number of scripts which allow us to store and load arbitrary R models from binary files. That is, once a model has been built, it is saved to the native R binary format and then placed on the remote R server.

Table 1. Summary of the R Functionality Available as Web Services

service	description
linear regression	build multiple linear regression models and use previously built models for prediction; provides methods to retrieve model statistics
neural network regression	build neural network regression models and use previously stored models for prediction
random forest	build random forest models for regression and classification; also use previously stored models for prediction
LDA	build linear discriminant analysis models for classification and use previously built models for prediction
k-means	performs *k*-means clustering
feature selection	performs variable selection; currently restricted to exhaustive search and stepwise methods for linear regression models
model generation	automatically generate linear and nonlinear models given a set of descriptors and a dependent variable
sampling distributions	sample from a variety of sampling distributions
Plots	generate 2D scatter plots and histogram plots

Once the remote server has been started, one can programmatically communicate with the server using a variety of languages. Since our focus was on Java, we used the JRI library to develop a set of Web services (hosted in an Apache Tomcat environment) that would interact with the remote R server. These classes essentially provide a simplified application programming interface to the R server—abstracting many of the details and presenting a task oriented interface to the user. Table 1 summarizes the various R methods that are available via the Web service infrastructure. It is clear that these methods focus on the analysis of new data but do not provide direct access to previously built models. Essentially, R functions for various model types are made visible to a remote caller in the form of a Web service function. Thus the lm() function in R takes a number of arguments representing x and y data, weights, and so on and returns a linear regression model. The Web service infrastructure provides a class called LinearRregressionModel that allows one to access the lm() function on the remote R server. It should be noted that the Web service interface does not completely imitate the lm() function. That is, we provide a Web service that allows one to specify x and y data along with weights, but, in addition, we also provide a Web service function that allows one to specify just x and y data and uses defaults for the other parameters to lm(). Once a model has been built the caller can query various aspects of the model such as fitted values, *F*-statistic, *t*-values, and so on. The functionality for other model types such as neural networks and random forests is similar. A list of the statistical Web services that we provide can be viewed at http://www.chembiogrid.org/projects/proj_statistics.html

The characteristic feature of these Web service interfaces is to let one develop statistical models on the fly. That is, one uploads the data, develops a model, and then uses it for prediction. However, one may not always have new data for prediction purposes at the same time as model building. That is, the development of the model and the use of the model may not necessarily be sequential operations. As a result the infrastructure must be able to store models for later use. However, the underlying principle of Web service technologies is that they are generally stateless. This presents us with a contradiction since we must maintain state (i.e., the model we just developed), yet this cannot be done with the Web service. This problem is quite easily solved by the use of R as a back end to the Web service rather than it being tightly integrated with the Web service itself. Thus when a user calls a Web service method to build a linear regression model, the request along with the associated data is passed to the R back end. The R server builds the model, then writes it to a binary file on the local disk, and then returns a unique model ID back to the caller. Thus the caller does not interact with the model directly. Now, after the initial call to the Web service to build the model, the caller is not required to immediately use the model for prediction as it has been saved on the server. When the need arises for a prediction (or some other information from the model), the caller once again uses the Web service methods provided but will now specify the model ID that was returned initially. Since this is a unique identification string, the server can easily reload the saved model and perform the required calculation.

We have thus developed a set of R functions which allow us to load stored model files into the R server and then access them via a Web service. This implies that for each model, we must develop a front end class that handles the details of connecting to the R server and running the actual model. Currently the development of the front end class is done manually, though we will provide an automatic mechanism by which such a class can be generated.

The model loading classes are generic so that it is possible to save models of different types (linear regression, random forest, and neural networks) and use the same underlying code to obtain predictions from them. In addition, we have provided a class that is capable of handling ensemble models, where the components may be of different types.

Once the models have been stored, it is relatively easy to access the model functionality in a variety of ways. Probably the easiest and most familiar method would be via a Web page. We have generated such Web page clients for a variety of models including the two mentioned in this paper. In addition, one can easily create command line clients that access the models via the Web services, thus allowing one to use them in batch mode. Finally, since the models are simply R binary files, it is possible to download them and load them into a local R session. We provide a Web page (http://www.chembiogrid.org/cheminfo/rws/mlist) that allows one to browse the list of stored models and download them if desired.

In addition to model deployment, the fact that one can incorporate arbitrary R code into the back end and provide a Web service front end makes the infrastructure very flexible. As an example, we were able to convert a pharmacokinetic parameter calculator[22] into R and then deploy it via a Web service. We then provided a Web page client

Table 2. Confusion Matrices for the Mutagenicity Data Set Obtained Using the Random Forest Model.

		predicted		
		mutagen	nonmutagen	class error (%)
		Training Set		
observed	mutagen	1633	272	14.2
	nonmutagen	262	1302	16.8
		Prediction Set		
observed	mutagen	435	61	12.3
	nonmutagen	72	300	19.4

for this service (http://www.chembiogrid.org/cheminfo/pk-cell). More relevant to QSAR models, we have also developed services that allow one to perform feature selection on a set of uploaded descriptors as well as upload a small set of descriptors and automatically generate a set of models (both linear and nonlinear). In the latter case, we also provide some basic validation statistics (RMSE from y-scrambling and LOO q^2). Both services can be accessed via SOAP, and as an example we have provided Web page clients for both the services (http://www.chembiogrid.org//cheminfo/rws/fs and http://www.chembiogrid.org/cheminfo/rws/mmg). We should note that these are examples of applications that can be built on top of the current infrastructure. Obviously, one would like to have more extensive measures of model validity as well as a measure of model domain applicability for the automatically generated models. We are currently researching such methods, and once working algorithms have been developed they can be easily deployed in the infrastructure.

5. RESULTS

We first report on the results obtained for the mutagenicity data set. Since the populations of the mutagenic and nonmutagenic classes were not significantly imbalanced, we did not employ any special procedures to generate balanced models. We split the data set into a training set and a prediction set such that 80% of the data set was placed in the training set and the remainder in the prediction set. The confusion matrices for each set are summarized in Table 2. For the training and prediction sets the overall percent classification error was 15.4% and 15.3% (i.e., the ratio false positives and false negatives to the total number of compounds), respectively, which compared favorably with previous work on this data set.[8] Given the similarity of predictive performance between the training and prediction sets we believe that the model does not exhibit overfitting and can be usefully deployed. The good performance of the model is further justified by an analysis of the true positives versus the false positives. This analysis can be summarized by a graphical representation termed the Receiver Operating Characteristic (ROC) curve. The ROC curve for this model is shown in Figure 2. The red dashed line indicates the performance of a perfectly random classifier. A perfect classifier would have a curve such that at a false positive rate of 0% it would have a true positive rate of 100%. However, in real classifiers this is not the case, and thus, for any given true positive rate, there will be a number of false positives. A good classifier is one that minimizes the false positive rate, and thus the area under the curve (AUC) will be close to 1. For the ROC curve shown in Figure 2 the

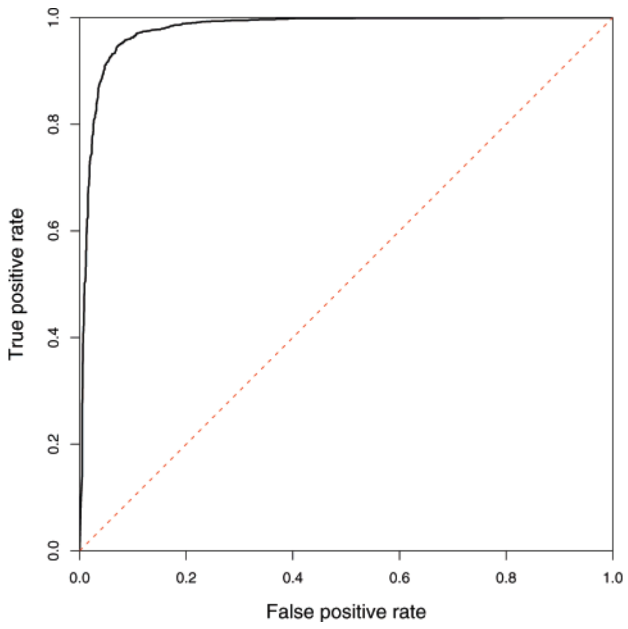


Figure 2. Receiver Operating Characteristic (ROC) curve for the random forest classification model developed for the mutagenicity data set. The dashed red line corresponds to a perfectly random classifier.

AUC is equal to 0.97, indicating good classification performance.

We then considered the structural features as identified by the random forest model that were important for predictive ability. The random forest uses a randomization scheme to evaluate the importance of individual descriptors. Essentially, it considers a single feature, which it then shuffles. The model is then rebuilt using the shuffled feature, and the decrease in prediction accuracy (or increase in root-mean-square error for regression modes) is noted. It is assumed that the larger the decrease in prediction accuracy arising from the scrambling of a feature, the more important that feature is to the models predictive ability. Using this approach one can rank the input descriptors in order of importance. Figure 3 is a descriptor importance plot for the five most important bits in the MACCS fingerprint. From the figure it is clear that scrambling the descriptor represented by bit position 105 (corresponding to spiro ring systems) in the MACCS fingerprint leads to a significant decrease in accuracy. Table 3 lists the SMARTS patterns for the bit positions noted in Figure 3. Table 4 displays some structures from the data set that exhibit the substructures identified in Table 3. It is interesting to note that none of the SMARTS are significantly specific. Rather, the second most important feature appears to be the presence of halogens. Furthermore, bit position 125 (more than one aromatic ring) appears to be a subset of the features characterized by bit position 145 (one or more 6-membered rings). However, when compared to the results reported by Bursi et al.,[7] we see that 4 of the important SMARTS patterns do correspond to reported toxicophores. It was also interesting to note that the ring compounds exhibiting a spiro structure were not noted as a toxicophore in the original work. One possible reason for this is that the random forest variable importance measure focuses on identifying descriptors that are important for the models predictive ability. That is, they are important because they allow the model to predict the positive (mutagenic) and negative (nonmutagenic) classes properly and do not necessarily focus on one specific class.

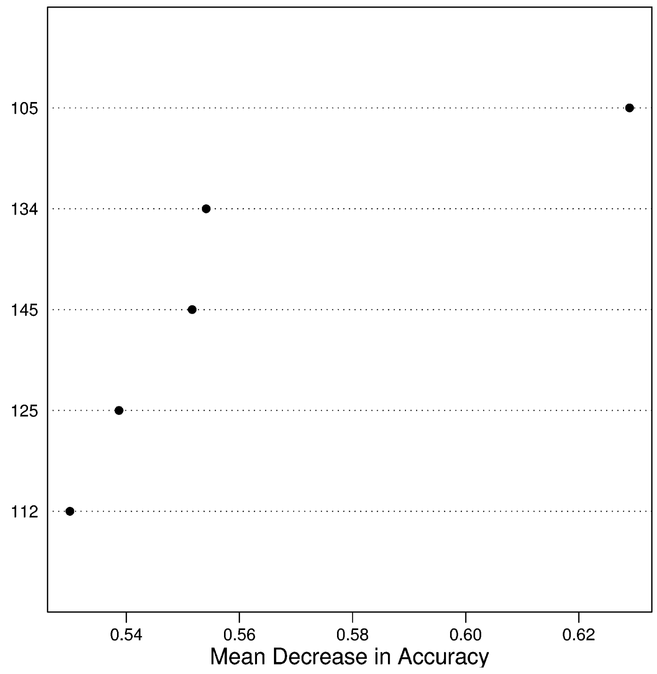


Figure 3. A variable importance plot of the random forest model for the mutagenicity data set. Only the five most important bit positions are shown.

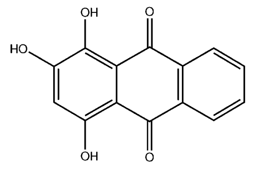
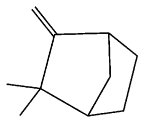
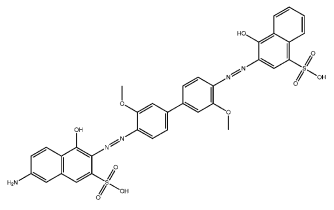
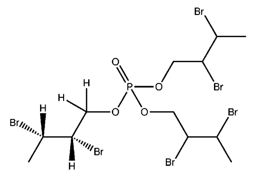
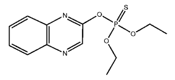
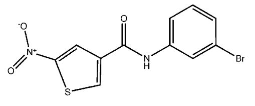
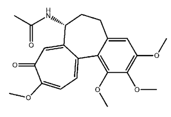
Table 3. SMARTS Patterns for the Five Most Important MACCS Keys for the Ames Mutagenicity Data Set

bit position	SMARTS	explanation
105	*@*(@)*@	spiro ring systems
134	[F,Cl,Br,I]	halogens
145	*~1~*~*~*~*~*1.*~1~*~*~*~*~*1	one or more 6-membered rings
125	more than one aromatic ring	
112	*~*(~*)(~*)~*	an atom in a quarternary position (i.e., bonded to 4 other atoms)

Next we consider the results for the models built on the NCI data set. As noted above, we developed models for the 60 cell lines. The performance of these models are summarized in Figure 4. The topmost figure shows the percent correct classification on the prediction set for each of the 60 models developed, with the red bar indicating the average percent correct over all 60 models. It is clear that for a number of the models the classification accuracy is greater than 80%. The middle figure shows the percent correct for the active class in the prediction set for the 60 models, and the lower figure shows the percent correct for the inactive class in the prediction set. In general a number of the models exhibit accuracy rates of 80%, though when averaged over all the models, this decreases to around 76%.

The relatively consistent behavior of the models in terms of classification accuracy is a little surprising given the variety of cell lines used. There are a number of possible reasons for this behavior. First, the modeling protocol employed presents an equal distribution of classes to the random forest algorithm. Thus, even though some cell lines have a 2.5:1 ratio of inactives to actives, the model, during training, will see a 1:1 ratio. This implies that the imbalanced

Table 4. Examples of Molecules Exhibiting the Most Important Substructures Identified in Table 3

Bit position		
105		
145		
112		
125		

nature of the classes for different cell lines is no longer visible to the modeling routine. This does not entirely explain the results since Rabow et al.[23] have used self-organizing maps to cluster the DTP compounds into several distinct response categories. The second reason for the predictive consistency of our models is the descriptors used. We have used 1052 bit structural fingerprints, and, though discriminatory, such descriptors do not take into account a variety of physicochemical effects such as permeability, absorption, metabolism, and so on, though they may be considered indirectly by virtue of specific substructures. As a result it is possible that within the chemical space of the BCI fingerprints, the distribution of features within the compounds is sufficiently general (as opposed to the distribution being skewed toward a small set of features) that the random forest algorithm is able to correlate the different dependent variables with the descriptors to a similar degree. This also implies that though we have presented results for a prediction set, this set is part of the DTP and thus will have much in common with the training set. Even though random forests do not overfit, good results on the prediction set do not translate to similar predictive performance for molecules that the model has not been trained on or validated with. This is the problem of domain applicability. Though this aspect must be taken into account, we are still in the process of developing methods to define the domain of applicability, and we expect to add such methods to the model described here in the future.

We next considered the importance descriptors (i.e., structural features) for the NCI DTP models. Clearly it is difficult to consider a descriptor's importance for all 60 models, and, rather than considering a descriptor's importance for the individual models, we examined the unique set of descriptors obtained from the top three descriptors for each model. Given that the cell lines are not identical one would expect that the over 60 cell lines identifying the top three features would exhibit some degree of variation. In the worst case, we would get 180 distinct features. However, analysis of the 60 models indicates that the total number of unique features obtained from the top three for each model is just 18. One might conclude that in terms of chemical structure the cell lines do not differ significantly in their response. If we next consider the distribution of features in the first, second, and third position we see that over the 60 models, only ten distinct features are ranked as the most important feature for predictive ability. For the second and third most important features, we observe 15 and 13 unique features, respectively. It turns out that when we consider the most important feature across the 60 models, bit position 106 (corresponding to [!#6]~*(~[!#6])~[!#6]) occurs 23 times and bit position 132 (corresponding to [#8]~*~[CH2]~*) occurs 12 times.

6. MODEL DEPLOYMENT

Both of the models described in this paper have been deployed in the R Web service infrastructure. After installing each model as a binary file on the remote R server, a single Java class was implemented that would act as the interface to the model. Since both models used fingerprints as features, the only input to these classes was the SMILES string for a new prediction. The Java class handled the evaluation of the fingerprints for the supplied SMILES, conversion to the appropriate data structure, and communication with the R server. In addition, the models can be downloaded as binary R files from http://www.chembiogrid.org/cheminfo/rws/mlist.

Since the Ames model was a single random forest model, it lead to a reasonable small model in terms of disk space as well as memory consumption. In the case of the DTP models, though each individual model was reasonably small, the fact that we had to take into account 60 models leads to having to split the models into 3 separate files, each approximately 230 MB in size. Clearly loading models from the disk will consume a significant amount of time. Once loaded all 60 models consume nearly 1.5 GB of memory. To get around the model loading times, we preload the model files when starting the remote R server. Thus the models are always available in memory. However, due to the high memory consumption we cannot utilize all 60 models. As a result, we only provide predictions from the first 40 models.

Clearly, simply deploying the models in the infrastructure does not necessarily imply that the models are usable. That is, clients must be written to make use of the models via their Web service interfaces. As noted above, Web service clients can take on many forms ranging from Web pages and workflow tools to standalone command line or GUI clients. For the case of the models described in this paper we have developed Web page clients written in PHP and Python that obtain predictions from the models via Web services. The client for the Ames model can be viewed at http://www.chembiogrid.org/cheminfo/rws/ames, and the DTP model can be viewed at http://www.chembiogrid.org/cheminfo/ncidtp/dtp. As can be seen the complexity of the clients is up to the designer. For the Ames model we simply provide an entry box to supply SMILES string and return the predicted class. On the other hand, the pages for the DTP models are more involved and try to represent the predicted profiles for the supplied SMILES in a variety of ways. We have also developed other models which have been successfully deployed within Pipeline Pilot workflows.

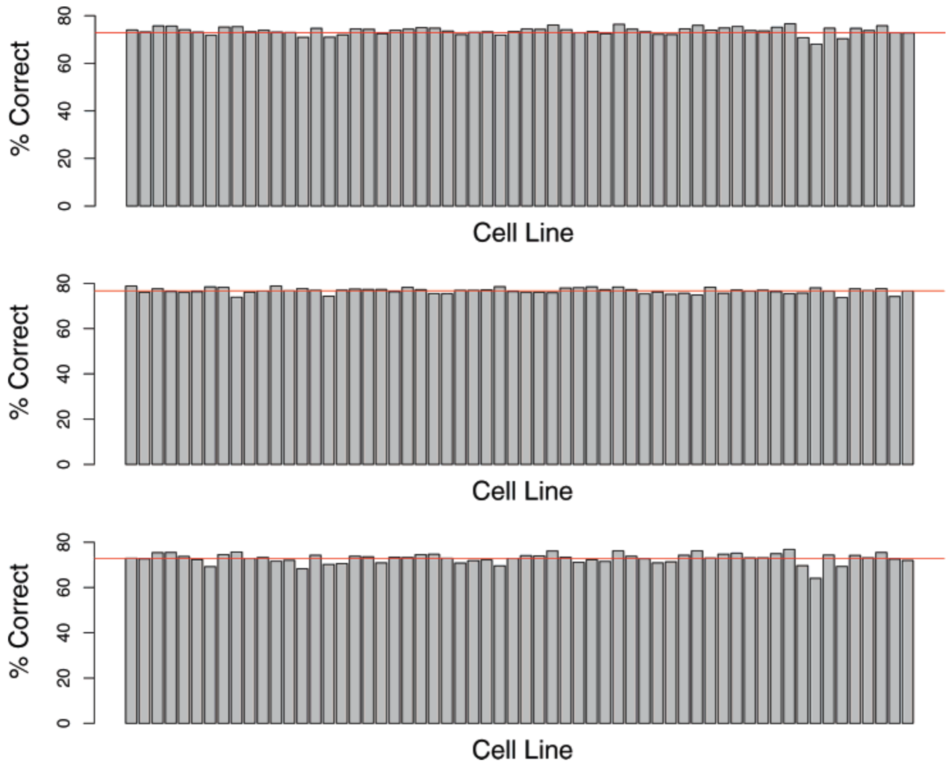


Figure 4. A summary of the overall percentage correct classification of the prediction set (top), percentage correct for the active class (middle), and the percent correct for the inactive class (bottom) for the 60 random forest models built for the 60 cell lines of the NCI DTP data set. The red line is the mean percent correct over all the cell lines.

At this point it is useful to compare the infrastructure described here to other approaches that are currently available. One of the recent approaches to model deployment is workflow tools such as Pipeline Pilot (Scitegic Inc.) or Knime (Knime GmBH). In the case of Pipeline Pilot, one is able to develop a workflow and create a Web portal. The Web portal does not require a user to have any knowledge of building workflows and simply requires the user to upload the relevant input data. The downside to this approach is that one must have a Pipeline Pilot installation to create such a Web portal. The cost of such a system may be prohibitive for a user. Second, the system is designed to provide Web interfaces to Pipeline Pilot workflows only. Granted, one can perform a wide variety of tasks in this environment, yet there are a wide variety of tools and platforms and even custom code that may be required to be deployed. Similar arguments can be provided for the case of Knime, though we are not aware of a “Web portal’’ facility for this tool. Compared to these environments, our infrastructure is much more low-level. But at the same time, our infrastructure is more general. Though we focus on R, this is more due to our usage of this platform than any restrictions to a specific platform. One could easily integrate models developed in SAS or SPSS into this system, but we are not restricted to predictive models. It is possible to deploy an arbitrary R code in the infrastructure, and we have performed precisely this task by making available a pharmacokinetic parameter calculator designed by Zhang et al.[22] This is not a predictive model but is a set of differential equations. However, after being deployed it was trivial to generate the Web service class and a simple Web page interface. Granted this step does require programming expertise. However, this is a well-defined procedure, and we are currently researching approaches to XML-based model specifications which would allow this procedure to be automated. Finally, another aspect of the generality of our approach is that by virtue of it being based on Web services, one is not restricted to any specific client platform. In many cases, such a platform will be a Web browser, but one could imagine custom applications that use Web services to create RSS feeds (possibly annotating new additions to a databases with predictions). Essentially, our infrastructure is client agnostic.

7. FUTURE WORK

Though the R Web service framework is quite general, a number of enhancements are planned. As noted in Table 1 we provide Web service interfaces to a number of R modeling techniques. We plan to extend these to a larger set of methods. Though the current process involves the manual implementation of such a Web service class, we plan to a provide code generator which will be able to automatically generate such a class.

We plan to significantly improve the performance of the back end R server by making use of cluster computing. The snow package[24] allows R to utilize MPI to distribute tasks across multiple CPUs. We plan to improve model loading for situations such as the DTP models (ensembles of models in general) so that individual models can be loaded on different CPUs which will ideally be on different machines. Furthermore, predictions can be sped up by distributing the calculations across multiple CPUs. We expect that these improvements will lead to significant speed ups, especially for large models.

Finally, a longer term task is to provide a better specification of models. Currently R itself provides capabilities to query a model regarding a number of characteristics. Thus, we currently simply store the model object in a binary file, and when we need to identify, for example, the type of the model being stored (neural network, linear regression, and so on), the number of input descriptors, or summary statistics, we must load the model into the R session and extract the required information. Though this is usually only done when the R server is started, it is quite inefficient in addition to being limited to what R stores along with the model. Thus we are not able to extract information regarding the authorship of the model, what the model is supposed to be predicting, and so on. More importantly, the current design does not allow us to identify what types of descriptors should be supplied to the model when making a new prediction. Thus a significant improvement would be to associate a specification document along with any model to be stored in the R server. We expect that the document would be based on XML and would include various details of the model such as model type, training statistics, and so on. In addition, we would be able to include descriptions of the model's purpose and authorship. As noted, it is equally important to be able to identify the nature of the descriptors that the model requires. We plan to include the ability to provide detailed specifications of descriptors covering information such as descriptor name, references, sources, and so on and will be based on the CDK[25] descriptor ontology. By including such extra, semantic information we believe that models will be able to be processed in an automatic fashion. Furthermore, with the development of semantic discovery tools, we expect that models can be discovered and analyzed automatically and subsequently used for their relevant purposes.

8. CONCLUSIONS

The development of predictive models is a common task in the cheminformatics community. Though models can be deployed in the form of Web pages supported by a variety of mechanisms, this approach traditionally binds the user to the interface developed by the modeler. The use of a Web service approach to model deployment allows a significant level of flexibility. Our approach has been to design an architecture based on R which loosely couples the computation engine and the actual Web service interface. The current infrastructure provides a simplified, task oriented interface to the R computation engine, allowing access to a wide variety of tools. For the purposes of model deployment, a number of classes have been designed to allow the loading of arbitrary models. These classes deal with the problem of maintaining state within the Web service by using the back end computation engine to actually store the models in disk-based binary files.

It should be noted that the infrastructure is quite low level and essentially defines a set of protocols for a Web service front end to interact with a computational back end. One of the stated aims of the infrastructure is to allow easy and flexible deployment of predictive models. The current infrastructure allows this task. However, one is not limited to model deployment. As we have noted, one can deploy arbitrary R code and provide a Web service front end. We have developed such applications (comprised of the computational and Web service code and exemplar client code) which highlight this aspect. Given that they are exemplars, they do not provide all the functionality that may be desired. However, it is clear that any improvements or advances (say, a new model applicability algorithm) can easily be deployed in the infrastructure to enhance any given application.

We believe that the infrastructure provides a useful and generalizable approach to the deployment of arbitrary predictive statistical models.

ACKNOWLEDGMENT

This work was supported by NIH grant No. HG 003894-01.

REFERENCES AND NOTES

(1) Manallack, D. T.; Pitt, W. R.; Gancia, E.; Montana, J. G.; Livingstone, D. J.; Ford, M. G.; Whitley, D. C. Selecting Screening Candidates for Kinase and G Protein-Coupled Receptor Targets Using Neural Networks. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 1256−1262.

(2) Bruce, C. L.; Melville, J. L.; Pickett, S. D.; Hirst, J. D. Contemporary QSAR Classifiers Compared. *J. Chem. Inf. Model.* **2007**, *47*, 219−227.

(3) Curcin, V.; Ghaem, M.; Guo, Y. Web Services in the Life Sciences. *Drug Discovery Today* **2005**, *10*, 865−871.

(4) Oinn, T.; Addis, M.; Ferris, J.; Marvin, D.; Senger, M.; Greenwood, M.; Carver, T.; Glover, K.; Pocock, M.; Wipat, A.; Li, P. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics J.* **2004**, *20*, 3045−3054.

(5) Development Core Team, R. *R: A Language and Environment for Statistical Computing*'; R Foundation for Statistical Computing: Vienna, Austria, 2005; ISBN 3-900051-07-0.

(6) Ames, B. N.; McCann, H.; Yamasaki, E. Methods for Detecting Carcinogens and Mutagens with the Salmonella/Mammalian-Microsome Mutagenicity Test. *Mutat. Res.* **1975**, *31*, 347−364.

(7) Kazius, J.; McGuire, R.; Bursi, R. Derivation and Validation of Toxicophores for Mutagenicity Prediction. *J. Med. Chem.* **2005**, *48*, 312−320.

(8) Liao, Q.; Yao, J.; Yuan, S. Prediction of Mutagenic Toxicity by Combination of Recursive Partitioning and Support Vector Machines. *Mol. Diversity* **2007**, *11*, 59−72.

(9) Liao, Q.; Yao, J.; Li, F.; Yuan, S.; Doucet, J.; Panaye, A.; Fant, B. CISOC-PSCT: A Predictive System for Carcinogenic Toxicity. *SAR QSAR Environ. Res.* **2004**, *15*, 217−235.

(10) Serafimova, R.; Todorov, M.; Pavlov, T.; Kotov, S.; Jacob, E.; Aptula, A.; Mekenyan, O. Identification of the Structural Requirements for Mutagenicity, by Incorporating Molecular Flexibility and Metabolic Activation of Chemicals. II. General Ames Mutagenicity Model. *Chem. Res. Toxicol.* **2007**, *20*, 662−676.

(11) Tudor, G.; Gutierrez, P.; Aguilera-Gutierrez, A.; Sausville, E. Cytotoxicity and Apoptosis of Benzoquinones: Redox Cycling, Cytochrome C Release, and BAD Protein Expression. *Biochem. Pharamcol.* **2003**, *65*, 1061−1075.

(12) Vekris, A.; Meynard, D.; Haaz, M.; Bayssas, M.; Bonnet, J.; Robert, J. Molecular Determinants of the Cytotoxicity of Platinum Compounds: The Contribution of In Silico Research. *Can. Res.* **2004**, *64*, 356−362.

(13) Efferth, T.; Oesch, F. Oxidative Stress Response of Tumor Cells: Microarray-Based Comparison Between Artemisinins and Anthracyclines. *Biochem. Pharamcol.* **2004**, *68*, 3−10.

(14) Zaharevitz, D.; Holbeck, S.; Bowerman, C.; Svetlik, P. COMPARE: A Web Accessible Tool for Investigating Mechanisms of Cell Growth Inhibition. *J. Mol. Graphics Modell.* **2002**, *20*, 297−303.

(15) Huang, R.; Wallqvist, A.; Covell, D. Assessment of In Vitro and In Vivo Activities in the National Cancer Institute's anticancer Screen with Respect to Chemical Structure, Target Specificity, and Mechanism of Action. *J. Med. Chem.* **2006**, *49*, 1964−1979.

(16) Ren, S.; Lien, E. Anticancer Agents: Tumor Cell Growth Inhibitory Activity and Binary QSAR Analysis. *Curr. Pharm. Des.* **2004**, *10*, 1399−1415.

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

(18) Svetnik, V.; Liaw, A.; Tong, C.; Culberson, C.; Sheridan, R.; Feuston, B. Random Forest: A Classification and Regression Tool for

Compound Classification and QSAR Modeling. *J. Chem. Inf. Comput. Sci.* **2003**, *42*, 1947−1958.
(19) Cannon, E. O.; Bender, A.; Palmer, D. S.; Mitchell, J. B. O. Chemoinformatics-Based Classification of Prohibited Substances Employed for Doping in Sport. *J. Chem. Inf. Model.* **2006**, *46*, 2369−2380.
(20) Guha, R.; Jurs, P. C. Development of Linear, Ensemble, and Nonlinear Models for the Prediction and Interpretation of the Biological Activity of a Set of PDGFR Inhibitors. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 2179−2189.
(21) Urbanek, S. *Rserve: Binary R server*. http://cran.r-project.org/src/contrib/Descriptions/Rserve.html (accessed September 2007).
(22) Zhang, X.; Shedden, K.; Rosania, G. A Cell-Based Molecular Transport Simulator for Pharmacokinetic Prediction and Cheminformatic Exploration. *Mol. Pharm.* **2006**, *3*, 704−716.
(23) Rabow, A.; Shoemaker, R.; Sausville, E.; Covell, D. Mining the National Cancer Institute's Tumor-Screening Database: Identification of Compounds with Similar Cellular Activities. *J. Med. Chem.* **2002**, *45*, 818−840.
(24) Tierney, L.; Rossini, A.; Li, N.; Sevcikova, H. *snow: Simple Network of Workstations*. http://cran.r-project.org/src/contrib/Descriptions/snow.html (accessed September 2007).
(25) Steinbeck, C.; Hoppe, C.; Kuhn, S.; Floris, M.; Guha, R.; Willighagen, E. Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics. *Curr. Pharm. Des.* **2006**, *12*, 2110−2120.

CI700188U