QSAR Ready Biodegradability Prediction

Lei Lyu
UNI:ll3433
ll3433@columbia.edu

Yang Yu
UNI:yy3102
yy3102@columbia.edu

Wenxiang Zhou
UNI:wz2542
wz2542@columbia.edu

Yi Chen
UNI:yc4029
yc4029@columbia.edu

Abstract

With the development of modern technology, environmental problem is attracting more and more attention, especially for ready biodegradability of chemicals. In this work, we reproduced QSAR(Quantitative Structure-Activity Relationship) Models in Mansouri's Work: kNN(k Nearest Neighbors), PLSDA(Partial Least Squares Discriminant Analysis), and SVM(Support Vector Machines), as well as their consensus models. To better discriminate biodegradable and nonbiodegradable chemicals, we then proposed other models: LDA(Linear Discriminant Analysis), Naive Bayes, Decision Tree(without purning, pruning), Bagging, Random Forest, AdaBoost(with Freund coefficient, with Breiman coefficient), NN(Neural Network), DNN(Deep Neural Network) and their consensus models. All the method implementation were in R. The reproduced and newly proposed models demonstrated good classification performs with respect to the original paper work.

1. Introduction

With the development of science, technology and industry, environment pollution has always been a problem that is presented to the human being. To be specific, humans are manufacturing and using a vast amount of products every second, and in the meantime, discarding a lot of them and emitting exhaust gases and liquid waste. Many of these emissions and dropped products, which consist of various kinds of chemicals, are spreaded directly into the environment. Some of these chemicals are hard to degrade, which leads to accumulation of persistent chemicals, resulting in continuous exposure and the increasing chemical concentration in the surroundings [1].

From 2007, the REACH regulation has been in effect in Europe. As a screening test for the assessment of biodegradability, REACH requires that chemicals produced or imported in quantities of more than 1 ton per year need information on ready biodegradation [2]. This is very reasonable because products used or consumed by humans consist of these chemicals, and it is very likely that they are left in the environment in the end. The EINECS list comprises more than 100 000 chemicals registered as being on the European Community market between 1971 and 1981 [5]. And those that have been tested for their biodegradability make up only a small portion of the list.

To increase the biodegradability data so that the emission of those chemicals that are not ready biodegradable be better monitored and controlled, Mansouri and others use the approach called QSAR (Quantitative Structure-Activity Relationships) to predict the biodegradability of chemicals that have not yet been tested [3]. In this article, we reapply some of the methods taken by Mansouri and his colleagues, and try a few new methods that they did not use to attain the results they came to.

2. QSAR Biodegradation Data Set

We used the QSAR Biodegradation Data Set. This data set was built in the Milano Chemometrics and QSAR Research Group [3]. It has been used for developing QSAR (Quantitative Structure-Activity Relationships) models for the study of the relationships between chemical structure and biodegradation of molecules. We downloaded the data set from the UCI Machine Learning Repository [4].

2.1. Description

The QSAR Biodegradation Data Set collected experimental values of biodegradation of 1055 chemicals from the webpage of the National Institute of Technology and Evaluation of Japan (NITE).In QSAR Biodegradation Data Set, 356 molecules are experimentally ready biodegradable (RB), while the other 699 molecules are not ready biodegradable (NRB). The QSAR Biodegradation Data Set contains 41 molecular descriptors and 1 experimental class. Those attributes and corresponding explanations are listed in Table 1.

2.2. Exploration

Before we go straight into re-implementing various methods in the original paper, we did some data exploration analysis. This can ensure there isn't anything too anomalous and it's also very important for the interpretation of our results.

One thing we would like to explore are correlations between the descriptors in the data set. Correlations between descriptors can show redundant information and other potential structural information about the data set. Figure 1 shows a heatmap of descriptor correlations for the QSAR Biodegradation Data Set. In the heatmap, we use blue to show the positive relationship and red to show the negative relationship. The shade of a block's color also represents the magnitude of the correlation between the two descriptors. From the heatmap, there appears to be many strong correlations. These correlations should be kept in mind while continuing with the analysis.

2.3. Data Visualization

We visualized our data set using PCA (Principle Components Analysis) method, for which we

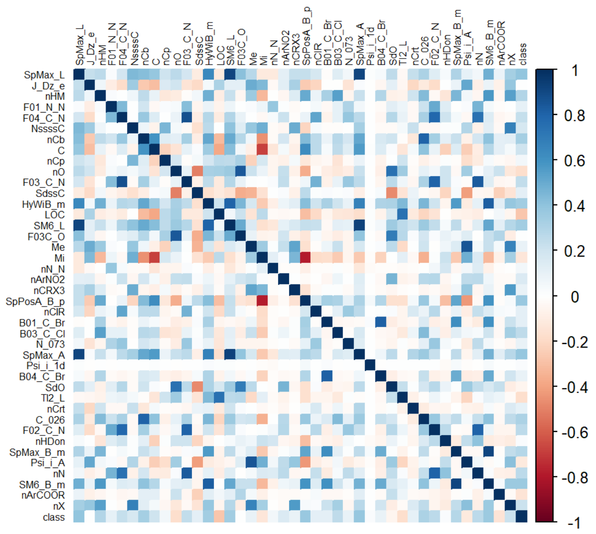


Figure 1. Heatmap of descriptor correlations

produced a score plot and loading plot on the first two principal components in Figure 2 and 3.

It shows that some variables are in high correlation in first two principal components. There is a big region of overlapping between two classes that indicates no easy splitting with PCA.

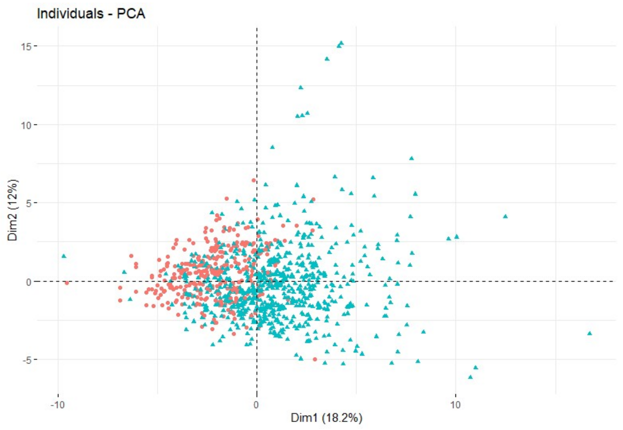


Figure 2. PCA score plot

3. Original Paper Details and Reproduction

3.1. Mansouri's Work (Quantitative Structure-Activity Relationship Models for Ready Biodegradability of Chemicals [3])

Classification models were produced in order to discriminate biodegradable and nonbiodegrad-

	Attribute	Explanation
1.	SpMax_L	Leading eigenvalue from Laplace matrix
2.	J_Dz(e)	Balaban-like index from Barysz matrix weighted by Sanderson electronegativity
3.	nHM	Number of heavy atoms
4.	F01[N-N]	Frequency of N-N at topological distance 1
5.	F04[C-N]	Frequency of C-N at topological distance 4
6.	NssssC	Number of atoms of type ssssC
7.	nCb-	Number of substituted benzene C(sp2)
8.	C%	Percentage of C atoms
9.	nCp	Number of terminal primary C(sp3)
10.	nO	Number of oxygen atoms
11.	F03[C-N]	Frequency of C-N at topological distance 3
12.	SdssC	Sum of dssC E-states
13.	HyWi_B(m)	Hyper-Wiener-like index (log function) from Burden matrix weighted by mass
14.	LOC	Lopping centric index
15.	SM6_L	Spectral moment of order 6 from Laplace matrix
16.	F03[C-O]	Frequency of C - O at topological distance 3
17.	Me	Mean atomic Sanderson electronegativity (scaled on Carbon atom)
18.	Mi	Mean first ionization potential (scaled on Carbon atom)
19.	nN-N	Number of N hydrazines
20.	nArNO2	Number of nitro groups (aromatic)
21.	nCRX3	Number of CRX3
22.	SpPosA_B(p)	Normalized spectral positive sum from Burden matrix weighted by polarizability
23.	nCIR	Number of circuits
24.	B01[C-Br]	Presence/absence of C - Br at topological distance 1
25.	B03[C-Cl]	Presence/absence of C - Cl at topological distance 3
26.	N-073	Ar2NH / Ar3N / Ar2N-Al / R..N..R
27.	SpMax_A	Leading eigenvalue from adjacency matrix (Lovasz-Pelikan index)
28.	Psi_i_1d	Intrinsic state pseudoconnectivity index - type 1d
29.	B04[C-Br]	Presence/absence of C - Br at topological distance 4
30.	SdO	Sum of dO E-states
31.	TI2_L	Second Mohar index from Laplace matrix
32.	nCrt	Number of ring tertiary C(sp3)
33.	C-026	R–CX–R
34.	F02[C-N]	Frequency of C - N at topological distance 2
35.	nHDon	Number of donor atoms for H-bonds (N and O)
36.	SpMax_B(m)	Leading eigenvalue from Burden matrix weighted by mass
37.	Psi_i_A	Intrinsic state pseudoconnectivity index - type S average
38.	nN	Number of Nitrogen atoms
39.	SM6_B(m)	Spectral moment of order 6 from Burden matrix weighted by mass
40.	nArCOOR	Number of esters (aromatic)
41.	nX	Number of halogen atoms
42.	experimental class	ready biodegradable (RB) and not ready biodegradable (NRB)

Table 1. Attributes description of the QSAR Biodegradation Data Set

able chemicals by means of different mathematical methods: k nearest neighbors (kNN), partial least squares discriminant analysis (PLSDA), and support vector machines (SVM), as well as their consensus models. The system of Mansouri's work can be seen in Figure 4.

Mansouri and his colleagues did a lot of work on screening the raw data collected from NITE to make sure that they arrived at accurate models based on correct experimental values and molecular structures. They also combined many classification modeling methods, such as linear, non-

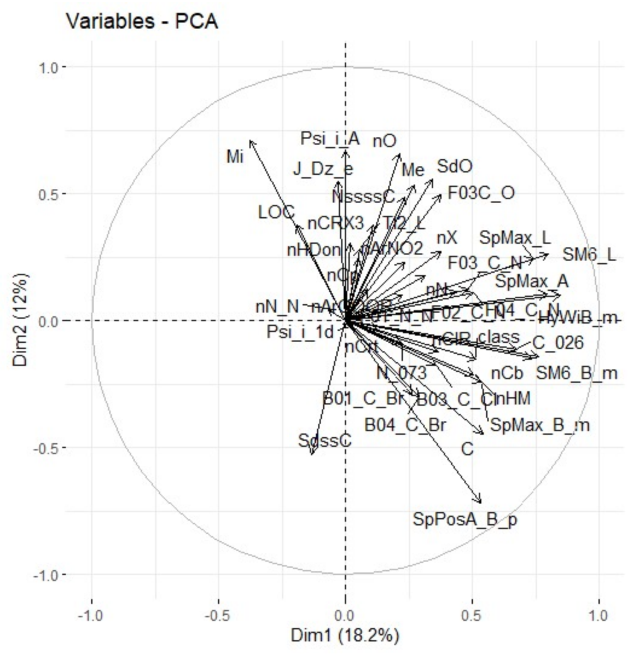


Figure 3. PCA loading plot

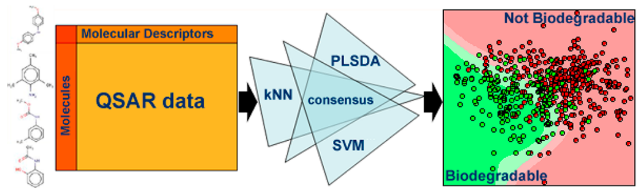


Figure 4. System of Mansouri's work

linear, local models and consensus models, with genetic algorithms in order to select the optimal subsets of molecular descriptors. Since we already have the final QSAR biodegradation data set, we are not going to discuss this part of their work thoroughly, but focus on the methods taken.

In Mansouri's work, 837 of the molecules were used for training, while the other 218 were used for testing. The number of RB and NRB molecules of training and test sets are summarized in Table 2. To ensure stability of the QSAR model and reliability of its predictions, Mansouri *et al.* reduced the descriptors into three subsets for kNN, PLSDA and SVM. The detailed information is listed in Table 3.

data	**ready biodegradable**	**not ready biodegradable**	**total**
training set	284	553	837
test set	72	146	218

Table 2. Number of molecules included in training and test set

3.2. Model Validation

One important thing needs to be mentioned is that the original paper used a cross-validation with five cancellation groups during model optimization and descriptor selection.

We have three evaluation metrics, specificity (Sp), sensitivity (Sn) and the classification error rate (ER). Specificity and sensitivity indicate the ability to correctly predict RB and NRB molecules, respectively. In addition, the classification error rate was calculated as the complement of the average of specificity and sensitivity. In particular, these metrics were calculated with the following equations:

$$S_p = \frac{TN}{TN + FP} \tag{1}$$

$$S_n = \frac{TP}{TP + FN} \tag{2}$$

$$ER = 1 - \frac{S_p + S_n}{2} \tag{3}$$

where, TN and TP are the number of true negatives and true positives, and FN and FP are the number of false negatives and false positives, respectively. One thing interesting is that considering this is a two-class model, the Sn of one class corresponds to the Sp of the other class.

These three metrics were used in order to better estimate classification performances in presence of a data set with unequal number of molecules in each class.

3.3. Reproduced Results

Three classification modeling methods were applied in the original paper: k nearest neighbors (kNN), partial least squares discriminant analysis (PLSDA), and support vector machines (SVM).

Symbol	Model
B01[C-Br]	PLSDA
B03[C-Cl]	PLSDA
B04[C-Br]	PLSDA
C%	kNN-PLSDA
C-026	SVM
F01[N-N]	kNN
F02[C-N]	SVM
F03[C-N]	kNN
F03[C-O]	PLSDA
F04[C-N]	kNN-PLSDA
HyWi_B(m)	PLSDA
J_Dz(e)	kNN
LOC	PLSDA
Me	PLSDA
Mi	PLSDA
N-073	PLSDA
nArCOOR	SVM
nArNO2	PLSDA
nCb-	kNN-SVM
nCIR	PLSDA
nCp	kNN
nCrt	SVM
nCRX3	PLSDA
nHDon	SVM
nHM	kNN
nN	SVM
nN-N	PLSDA-SVM
nO	kNN-PLSDA
NssssC	kNN-SVM
nX	SVM
Psi_i_1d	PLSDA
Psi_i_A	SVM
SdO	PLSDA
SdssC	kNN
SM6_B(m)	SVM
SM6_L	PLSDA
SpMax_A	PLSDA
SpMax_B(m)	SVM
SpMax_L	kNN-PLSDA-SVM
SpPosA_B(p)	PLSDA
TI2_L	PLSDA

Table 3. List of molecular descriptors selected in the QSAR models

We reproduced these three models in R and good results were obtained after comparison.

3.3.1 kNN (k Nearest Neighbors)

In accordance with the kNN classification rule, a molecule is classified according to the majority of its k nearest neighbors in the descriptors space. In this work, we used the Euclidean metric to measure distances between molecules as Equation 4.

$$\|a-b\|_2 = \sqrt{\sum_{i=1}^{d}(a_i-b_i)^2} \tag{4}$$

Note that the k value giving the lowest classification error in cross-validation was selected as the optimal one.

The confusion matrix of the reproduced kNN model can be seen as Figure 5 and the results compared with the original paper are listed in Table 4. In this work, kNN model selected 12 descriptors and from the table we known that our reproduced kNN model has lower classification error rate 0.12 with higher sensitivity 0.83.

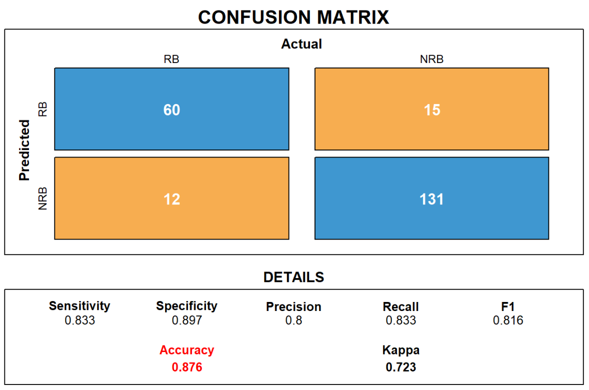


Figure 5. The confusion matrix of the reproduced kNN model

Model	desc	ER	Sp	Sn
kNN (paper)	12	0.15	0.90	0.81
kNN (reproduced)	12	0.12	0.90	0.83

Table 4. Classification results of the reproduced kNN model and kNN model in original paper

3.3.2 PLSDA (Partial Least Squares Discriminant Analysis)

PLSDA (Partial Least Squares Discriminant Analysis) is a Discriminant method in multivariate data analysis technology, which is often used to deal with classification and discrimination problems. Discriminant analysis is a commonly used statistical analysis method to judge how to classify research objects according to the values of several variables observed or measured. Its principle is to train the characteristics of different processing samples (such as observation samples and control samples) respectively, generate training sets, and test the credibility of training sets. Through proper rotation of principal components, PLSDA can effectively distinguish the observed values between groups and find the influencing variables that lead to differences between groups.

PLSDA adopts the classical partial least squares regression model, and its response variable is the classification information of the category relations among a group of response statistical units, which is a supervised discriminant analysis method. When the differences between groups of samples are not clear, but the differences within groups are large, unsupervised analysis is difficult to find and distinguish the differences between groups. In addition, if the difference between groups is small and the sample size of each group is large, the group with large sample size will dominate the model. Supervised analysis (PLSDA) can well solve these problems encountered in unsupervised analysis.

Note that PLSDA models were optimized in cross-validation to find a compromise between the classification performance and the number of selected latent variables.

The confusion matrix of the reproduced PLSDA model can be seen as Figure 6 and the results compared with the original paper are listed in Table 5. In this work, PLSDA model selected 23 descriptors and from the table we known that our reproduced PLSDA model has the same classification error rate 0.15 with higher specificity 0.92.

3.3.3 SVM (Support Vector Machines)

In machine learning, SVM (Support Vector Machines) are supervised learning models associated

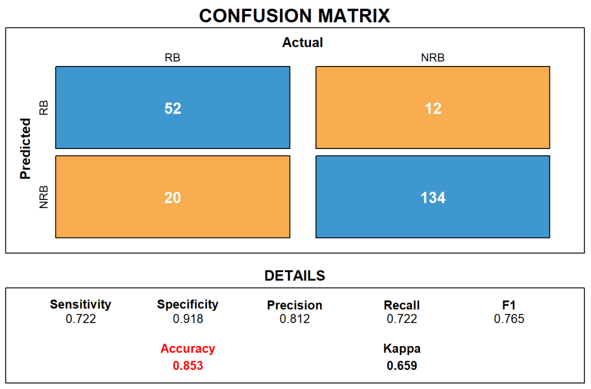


Figure 6. The confusion matrix of the reproduced PLSDA model

Model	desc	ER	Sp	Sn
PLSDA (paper)	23	0.15	0.87	0.83
PLSDA (reproduced)	23	0.15	0.92	0.72

Table 5. Classification results of the reproduced PLSDA model and PLSDA model in original paper

with related learning algorithms that can analyze data, identify patterns, and be used for classification and regression analysis. Given a set of training samples, each labeled as belonging to two classes, a SVM training algorithm builds a model that assigns new instances to one class or the other, making it an improbability binary linear classification. An example of a SVM model, such as point mapping in space, makes the representation of the different categories of examples divided by an obvious gap that is as wide as possible. The new embodiments are mapped into the same space and are predicted to belong to a class based on where they fall on the gap side.

In addition to linear classification, SVM can effectively perform nonlinear classification using the so-called kernel technique, where their inputs are implicitly mapped into higher-dimensional feature Spaces. More formally, a SVM constructs a hyperplane, or hyperplane in high or infinite dimensional space, which can be used to set the hyperplane in classification, regression, or other tasks. Intuitively, a good separation is achieved through the general implementation of a hyperplane with the maximum distance to the closest training data point of any class (the so-called functional margin) due to the generalization error of the classifier under the large margin.

The confusion matrix of the reproduced SVM model can be seen as Figure 7 and the results compared with the original paper are listed in Table 6. In this work, SVM model selected 14 descriptors and from the table we can see that our reproduced SVM model performed not as good as the original model in paperwork, with higher classification error rate 0.15.

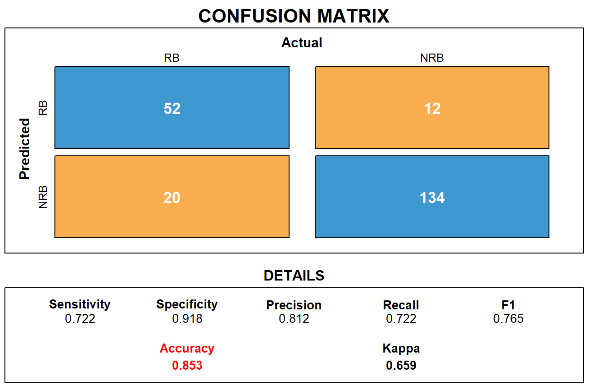


Figure 7. The confusion matrix of the reproduced SVM model

Model	desc	ER	Sp	Sn
SVM (paper)	14	0.14	0.91	0.82
SVM (reproduced)	14	0.15	0.92	0.72

Table 6. Classification results of the reproduced SVM model and SVM model in original paper

So, we improved this SVM model by change its selected descriptors and cross-validation folds. In this improved SVM model, we used all the descriptors instead of 14 selected descriptors, and set 7-fold cross-validation instead of 5-fold to reduce the classification error rate. The confusion matrix of the improved SVM model can be seen as Figure 8 and the results compared with the other two SVM models mentioned above are listed in Table 7.

From the table we can see that our improved SVM model performed much better than the original model in paperwork with classification error rate 0.12. Compared with our reproduced SVM model, the improved one has been optimized in all aspects (classification error rate, specificity and sensitivity).

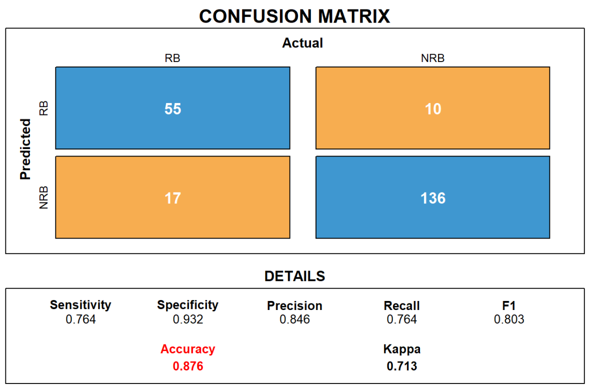


Figure 8. The confusion matrix of the improved SVM model

Model	desc	ER	Sp	Sn
SVM (paper)	14	0.14	0.91	0.82
SVM (reproduced)	14	0.15	0.92	0.72
SVM (improved)	41	0.12	0.93	0.76

Table 7. Classification results of the improved SVM model, the reproduced SVM model and SVM model in original paper

3.3.4 PCA (Principle Components Analysis)

We also reproduced the PCA visualization on the descriptors selected for kNN and SVM model in the paper. The results are shown in Figure 9, 10, 11 and 12.

4. Other Techniques Implementation

4.1. LDA (Linear Discriminant Analysis)

For Discriminant Analysis based on Bayes theorem to classify data to the class that it has the highest probability to be. Instead of directly estimating the posterior distribution, Discriminant Analysis makes assumptions on both likelihood $f_k(x)$ and prior π_k and then calculates the posterior according to Bayes theorem. Particularly in our classification tasks with two class RB and NRB, we can assume π_0 and π_1 proportional to the number of RB and NRB molecules in the whole data set. Then we assume that $f_0(x)$ and $f_1(x)$ follows normal distribution with different

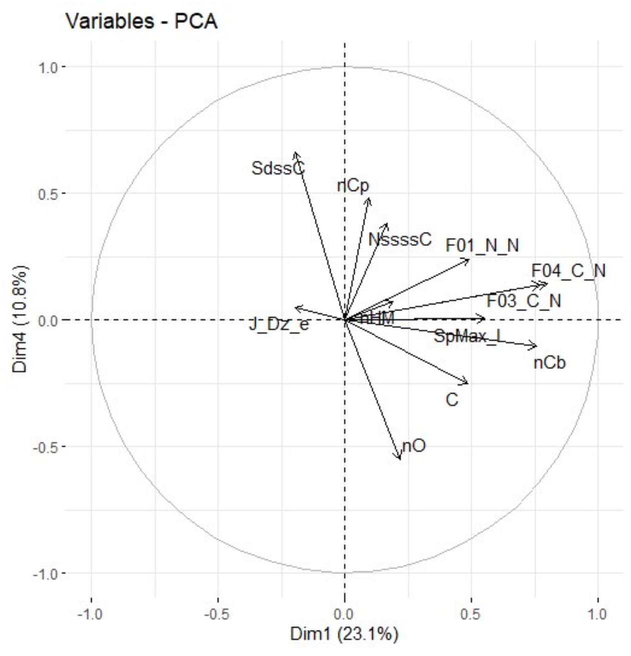


Figure 9. Knn descriptors PCA loading plot

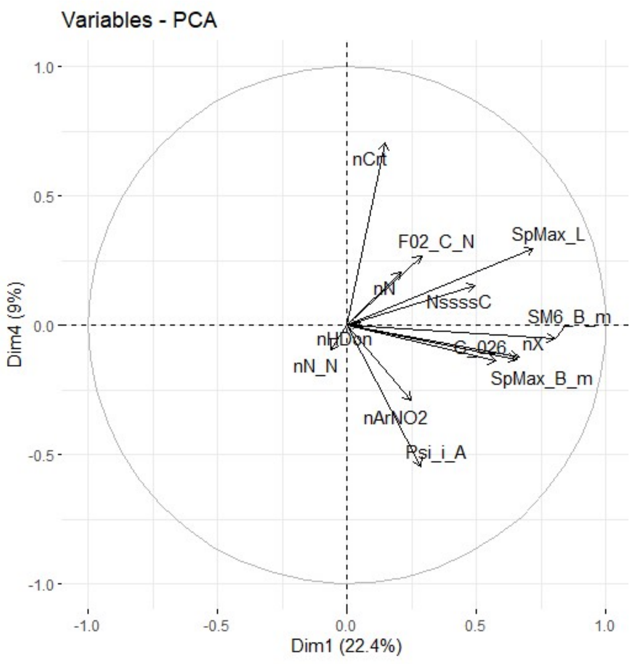


Figure 11. SVM descriptors PCA loading plot

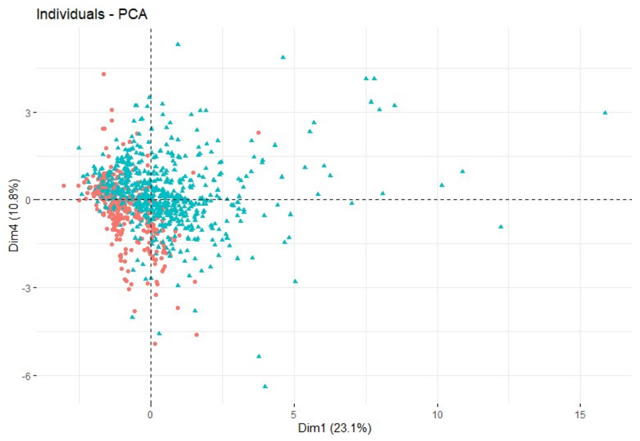


Figure 10. Knn descriptors PCA score plot

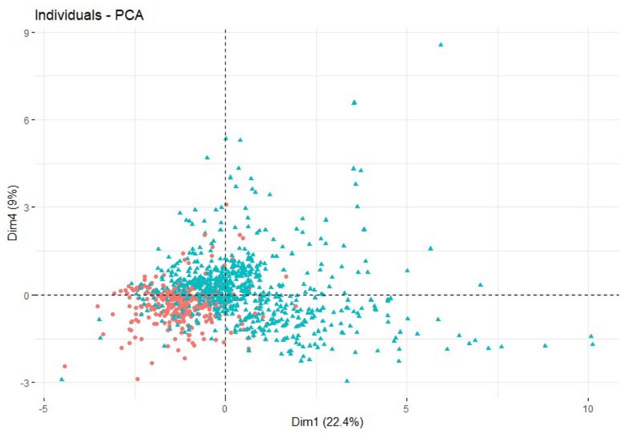


Figure 12. SVM descriptors PCA score plot

means $\mu_0 \neq \mu_1$ but the same variance $\sigma_0 = \sigma_1$. In this setting, it is just a Linear Discriminant model.

After building the LDA model and evaluating its performance on the testing set, we got the results shown in Figure 13. It shows that we got an accuracy of 85.3%, a sensitivity of 72.2% and a specificity of 91.8%.

4.2. Naive Bayes

Naive Bayes based on the idea that each variable in the data set is independent of each other, thus the likelihood function $f(x)$ can be expressed as products in terms all dimensions $f(x) = \prod_{i=1}^{p} f_i(x_i)$. Thus it is extremely useful to decrease the complexity of the model when dimension p of the variables is very large. Furthermore, it is not easy to define a joint probability function when there are some categorical variables. Naive Bayes algorithm addressed this problem by splitting those variables up, so it is much easier to define a probability function for

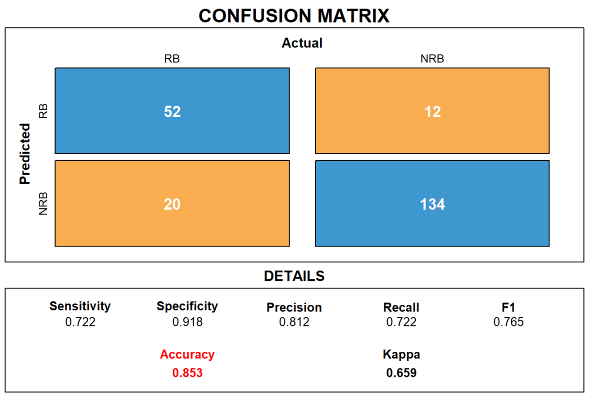


Figure 13. The confusion matrix of the LDA model

one categorical variable.

However, Naive Bayes seems to overly simplify the problem, as it is impossible to have variables independent of each other. Actually, from the PCA plot, we observed highly correlated relation between many pairs of variables. So it is expected that Naive Bayes can't do very well here.

The result in Figure 14 shows that we only got an accuracy of 72.9%, a sensitivity of 94.4% and a specificity of 62.3%. It is certainly that Naive Bayes is not a demanding model for this task.

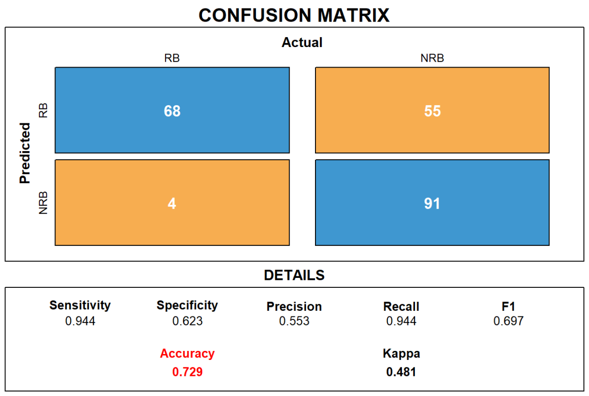


Figure 14. The confusion matrix of the Naive Bayes model

4.3. Decision Tree (Without Pruning)

Decision Tree algorithm is a wildly used method in classification tasks and it is also the fundamental methods to other algorithms like Bagging, Random Forest, AdaBoost, etc.. For each node in the decision tree that doesn't reach the leaf, it will be branched according to some criteria in some dimension. For example, if the data in one node of the tree is about to be split on dimension i, then finding the best splitting point a_i in dimension i, these data can be divided into two branches $x_i < a_i$ and $x_i \geq a_i$. This branching strategy also applies to categorical variable, depending whether $x_i = c_i$.

Thus Decision Tree method is very suitable for our classification task here. Then we first tried to build the tree in a forward direction, or in a a top to down direction, by greedily increasing the nodes in the Decision Tree model.

By doing so, we got the structure of the tree shown in Figure 15 and the results of prediction on testing set in Figure 16, which indicates an accuracy of 78.4%, a sensitivity of 80.3% and a specificity of 75.0%.

4.4. Decision Tree (Pruning)

Decision Tree can also be built in a backward manner. In this sense, we just grew a very huge Decision Tree first and then gradually pruned the tree to achieve the best performance under the trade-off between accuracy and size of the tree we set up for this problem.

Using cross-validation for the pruning process, we found the optimal size of leaves in the pruned tree to be 13, according to the plot in Figure 17. Then we grew such a Decision Tree with pruning shown in Figure 18. For the prediction performance of this model, we got the result in the confusion matrix in Figure 19, which achieved an accuracy of 81.7%, a sensitivity of 85.2% and a specificity of 75.0%.

From the results shown above, we can conclude that a single tree is not powerful enough to have a good prediction performance in this task. Thus we then considered using methods that ensemble various tree to make the prediction.

4.5. Bagging

Bagging is one of the method that can be built upon various tree. For the bagging algorithm, it will train several tree classifier $\hat{f}_t$ with $t = 1, \cdot, T$. For each f_t, there is a bootstrap procedure con-

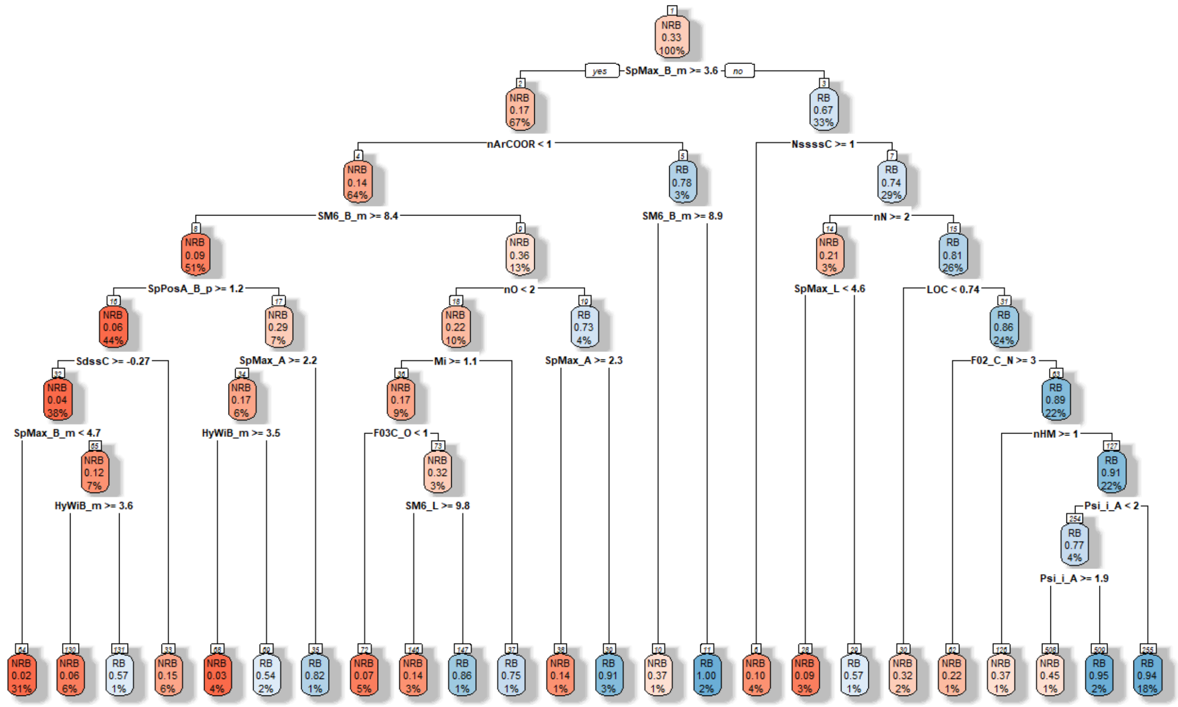
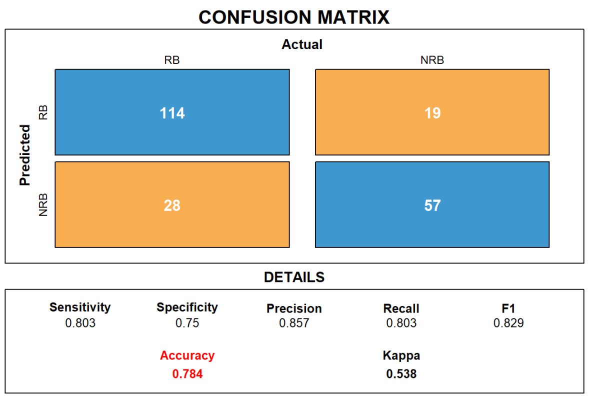


Figure 15. The structure of the Decision Tree (without pruning)



CONFUSION MATRIX

		Actual	
		RB	NRB
Predicted	RB	114	19
	NRB	28	57

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.803	0.75	0.857	0.803	0.829

Accuracy	Kappa
0.784	0.538

Figure 16. The confusion matrix of the Decision Tree (without pruning) model

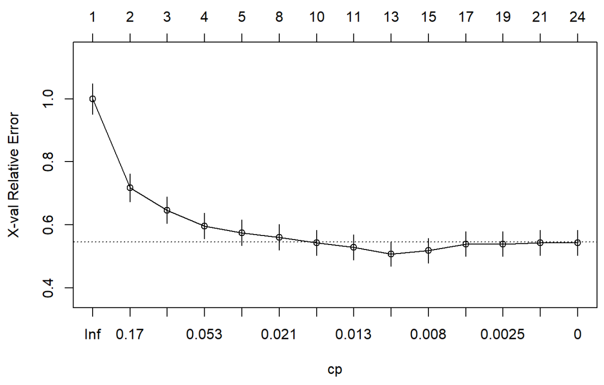


Figure 17. Cross validation for Decision Tree (pruning)

ducted before training to ensure that each of them is trained on different data sets. The bootstrap is unweighted sample from the original training set with replacement. Thus it can be assumed that each classifier can learn different knowledge from the original data set. Then after implementing all individual classifiers $\hat{f}_t$, their predictions are aggregated through a major vote strategy $\hat{f}_{major}(x_0) = round(\frac{1}{T}\sum_{i=1}^{T}\hat{f}_t(x_0))$, that is is the average prediction is closer to 0, then assigning the data to 0-class, otherwise, categorizing the data into 1-class.

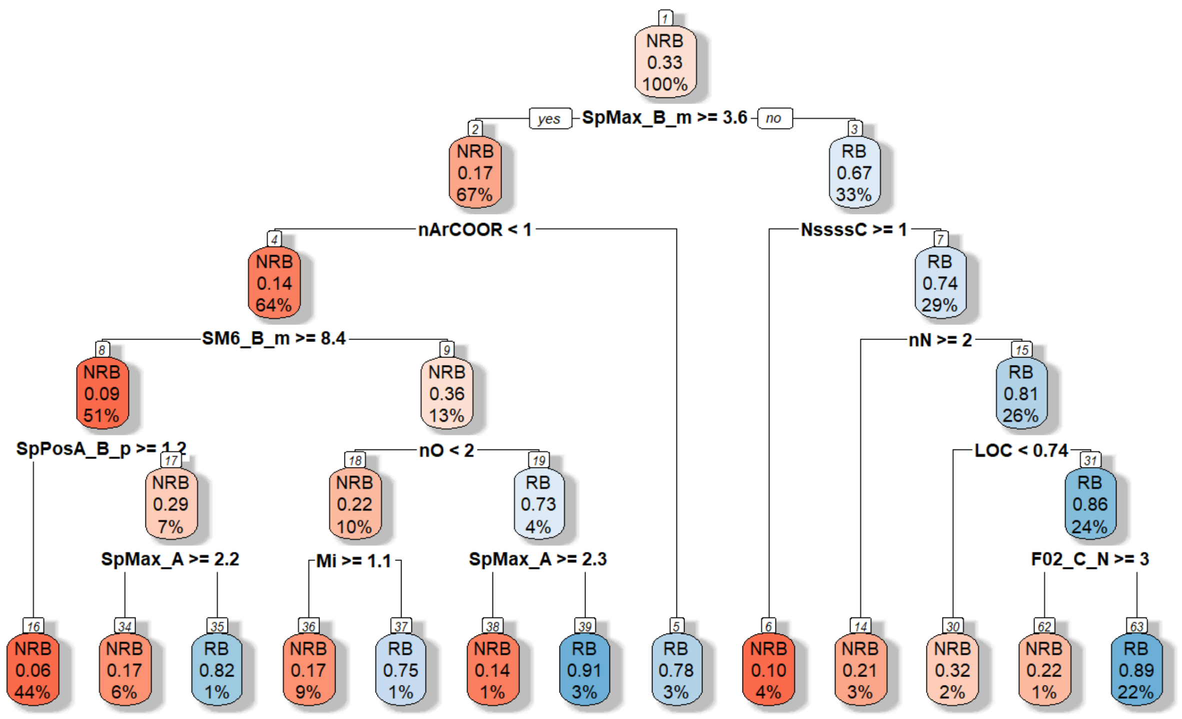
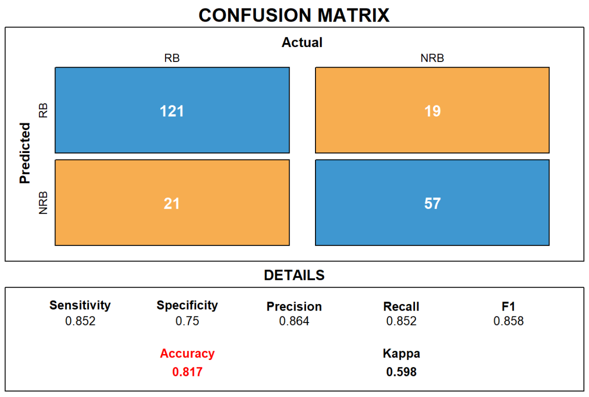


Figure 18. The structure of the Decision Tree (pruning)



CONFUSION MATRIX

Predicted \ Actual	RB	NRB
RB	121	19
NRB	21	57

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.852	0.75	0.864	0.852	0.858

Accuracy	Kappa
0.817	0.598

Figure 19. The confusion matrix of the Decision Tree (pruning) model

First, using a 5-fold cross-validation procedure accoding to Figure 20, we figured out the optimal number of classifiers $T^* = 150$.

Then we built the Bagging model with this size and gained the confusion matrix shown in Figure 21. This model reached an accuracy of 87.6%, a sensitivity of 75% and a specificity of 93.8%.

It can be seen that the performance of the model improved a lot upon the individual tree classifiers.

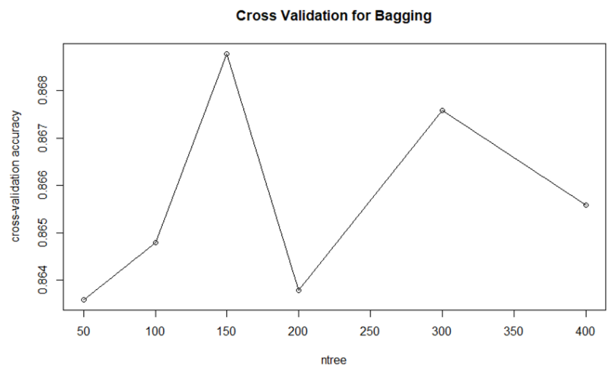


Figure 20. Cross validation for Bagging

4.6. Random Forest

Random Forest is another method to assembling these tree classifiers. It is very similar to

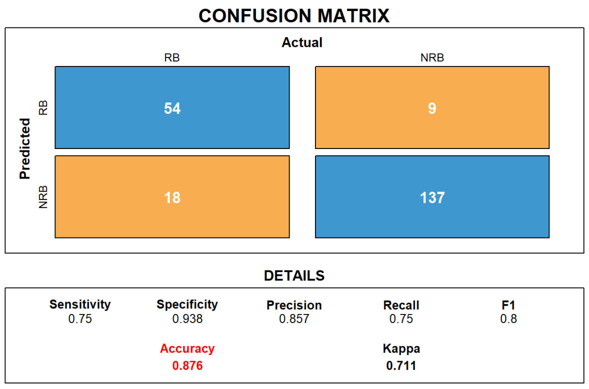


Figure 21. The confusion matrix of the Bagging model

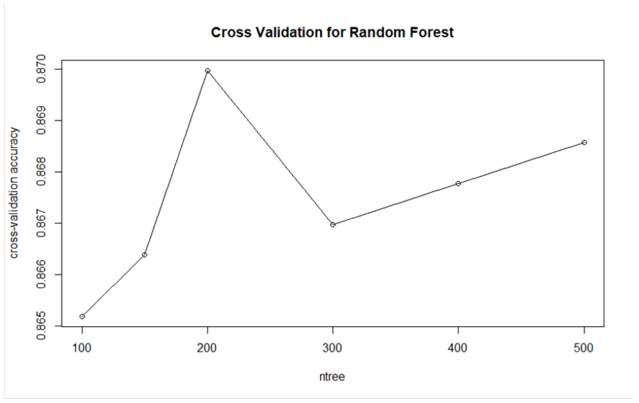


Figure 22. Cross validation for Random Forest

the Bagging algorithm, but with one key thing different. Instead of building the tree classifier in all dimensions, for each Random Forest classifier, they only expand the tree in several randomly selected dimensions. Usually the number of selected dimensions will be chosen as $\sqrt{p}$. Thus as we have 41 variable here in our data set, we only choose to branch each Random Forest classifier in 6 dimensions. This benefits of this strategy is to further decrease the correlations between each classifier, which has been alleviated through bootstrap. Thus it assumes that each classifier can learn more distinctive knowledge here, so that combining their results can be more reliable.

Again, by applying a 5-fold cross validation as shown in Figure 22, we chose the optimal size of the classifiers to be 200.

We then built the model and tested on the testing set, which gave us results in Figure 23. It shows that we gained an accuracy of 88.1%, a sensitivity of 73.6% and a specificity of 95.2%.

It seems to further improve on the performance of Bagging model.

4.7. AdaBoost (With Freund Coefficient)

AdaBoost algorithm, like Bagging and Random Forest, fits in various individual classifiers and generates the final classification based on all individual ones. One thing distinguishes it from the previous two methods is that AdaBoost applies much weaker individual classifiers that is just better than random guess but is much easier

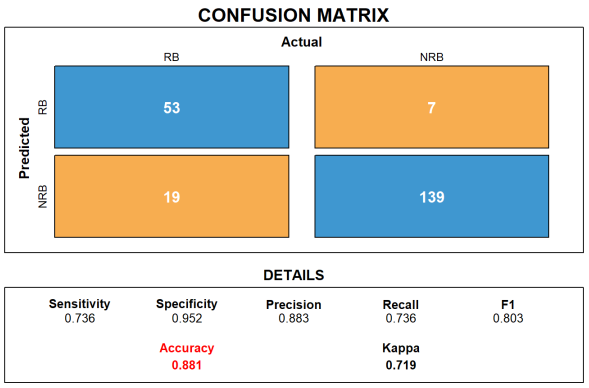


Figure 23. The confusion matrix of the Random Forest model

to train. Often, the weak classifier will be chosen as a tree classifier with only one branch. Then to train each individual classifier $\hat{f}_i$, there are two strategies to choose from.

One is no use of bootstrap. Under this scheme, all data will be used to fit in $\hat{f}_t$ but the final error is calculated with different weights on each data as $e_t = \frac{\sum_{i=1}^{n} w_i \mathbb{1}(y_i \neq \hat{f}_t(x_i))}{\sum_{i=1}^{n} w_i}$. Then the weight of this classifier $\hat{f}_t$ can be computed as the Freund coefficient $\alpha_t = \ln \frac{1-e_t}{e_t}$. Then the weights w_i are updated by $w_i \leftarrow w_i \exp\left(\alpha_t \mathbb{1}(y_i \neq \hat{f}_t(x_i))\right)$.

The other one is to use bootstrap before training each classifier. In this way, a new set of data with the same size is sampled from the original set of data with the weights of w_i on each original data.

The error is calculated as $e_t = \frac{\sum_{i=1}^{n} \mathbb{1}(y_i' \neq \hat{f}_t(x_i'))}{n}$. Then the following step of evaluating classifier weights α_t and updating data weights w_i is the same as above.

In our implementation, we used a 5-fold cross validation process to choose whether to use bootstrap and how many classifiers to use in total. After averaging the cross-validation error rates over 3 epochs, we got the plot as Figure 24, in which we can figure out the optimal choice to be no use of bootstrap and 50 classifiers to train.

The classification result of the mode under this choice in the testing data set is shown in Figure 25. We reached an accuracy of 85.8%, a sensitivity of 75% and a specificity of 91.1%.

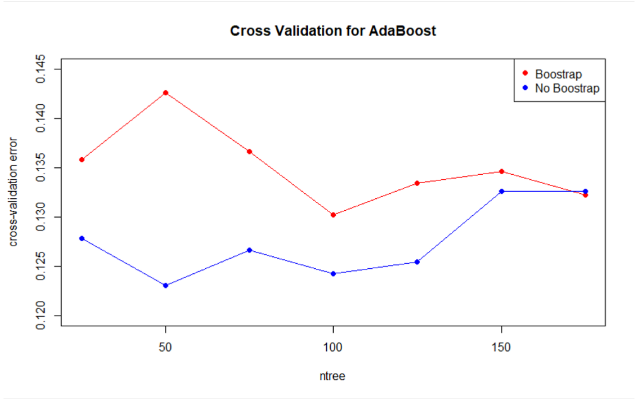
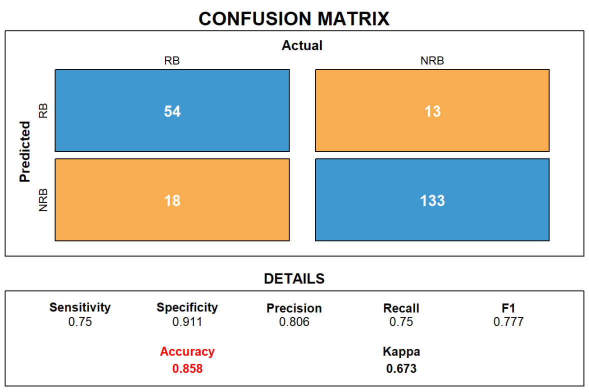


Figure 24. Cross validation for AdaBoost (with Freund coefficient)



CONFUSION MATRIX

Predicted \ Actual	RB	NRB
RB	54	13
NRB	18	133

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.75	0.911	0.806	0.75	0.777

Accuracy	Kappa
0.858	0.673

Figure 25. The confusion matrix of the AdaBoost (with Freund coefficient) model

4.8. AdaBoost (With Breiman Coefficient)

The same as AdaBoost algorithm describe above, in this section, we tried another version of AdaBoost with one coefficient differently defined. We set the weight of each classifier in the form of Breiman coefficient $\alpha_t = \frac{1}{2} \ln \frac{1-e_t}{e_t}$.

With this little modification, we again use a 5-fold cross validation process to choose whether to use bootstrap and how many classifiers to use in total. After averaging the cross-validation error rates over 3 epochs, we got the plot as Figure 26, in which we can figure out the optimal choice to be using bootstrap and 175 classifiers to train.

The classification result of the mode under this choice in the testing data set is shown in Figure 27. We reached an accuracy of 87.2%, a sensitivity of 76.4% and a specificity of 92.5%.

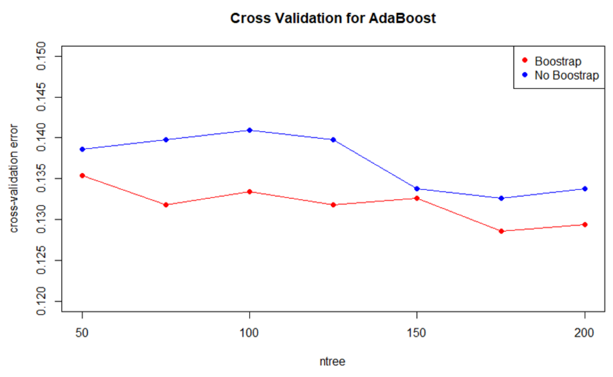
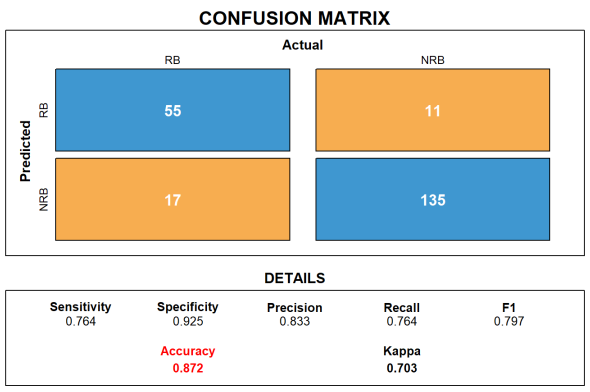


Figure 26. Cross validation for AdaBoost (with Breiman coefficient)



CONFUSION MATRIX

Predicted \ Actual	RB	NRB
RB	55	11
NRB	17	135

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.764	0.925	0.833	0.764	0.797

Accuracy	Kappa
0.872	0.703

Figure 27. The confusion matrix of the AdaBoost (with Breiman coefficient) model

4.9. Neural Network: 1 hidden layer

Neural Network consists of input layer, hidden layer and output layer. The depth of the hidden layer decide how complex the network is. For the simplest implementation, 1 fully connected layer added as the hidden layer is enough to do many tasks that seem to be challenging for many traditional machine learning model.

As shown in Figure 28, here we implemented such a neural network with 1 hidden layer activated by *ReLu* function, which is followed by a dropout layer with coefficient 0.4 and an output layer of 2 units activated by *Softmax* function. By using 0.2 of training data as validation set, we first figured out that the optimal number of units in that hidden layer should be 400 in our case.

Then we trained the neural network with our training set and the training process is shown in Figure 29. From these two plots of accuracy and loss changing during the training process, we concluded that our model converged well here.

Next, by using that trained model, we predicted on the testing set of data and the results are shown in Figure 30. We reached an accuracy of 88.5%, a sensitivity of 88.9% and a specificity of 88.4%.

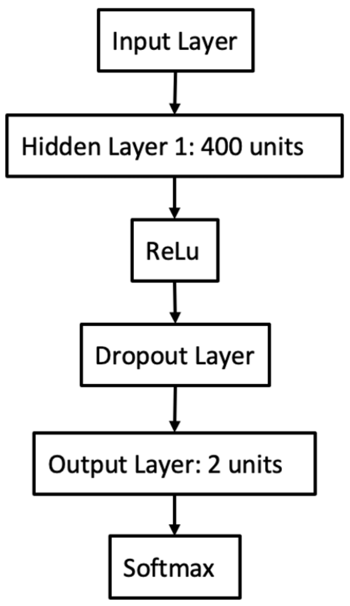


Figure 28. The structure of the Neural Network model

4.10. Deep Neural Network: 2 hidden layers

Furthermore, we want to see whether increase the complexity of the neural network will help here. Thus we implemented a Deep Neural Network Structure with 2 hidden layers between the input layer and output layer.

Again, with 0.2 out of all training data as the validation set, we chose the optimal units in each hidden layer as 400 units in the first fully connected layer and 50 units in the second fully connected layer.

Then training this model under the training set, we got the training process shown in Figure 31. The accuracy and loss plots behave not so well in terms of validation data, as they fluctuate in a big range on the plots.

The prediction results on the testing set also showed that adding to the complexity of neural network structure didn't help with the prediction here. According to Figure 32, the Deep Neural Network model have an accuracy of 87.6%, a sensitivity of 77.8% and a specificity of 92.5%. This performance is even not as good as the simpler neural network model.

5. Discussion and Conclusion

5.1. Consensus Model

After implementing all the algorithms and models listed above, we reached the best performance on predicting the testing data with the neural network, which achieved 88.5% accuracy, 88.4% specificity and 88.9% sensitivity. This is also the most balanced model, as it achieved almost the same performance in terms of distinguishing RB molecules and NRB molecules.

Furthermore, inspired by the paper, which raised up a consensus model after implementing KNN, SVM and PLSDA models with specific descriptors selection. In this consensus model, it applied a major vote strategy to ensemble the predictions from three individual models. For example, if kNN and SVM correctly predict a molecule as RB but PLSDA model fails to do so, the consensus model still can have a correct prediction of classifying the molecule as RB. In this sense,

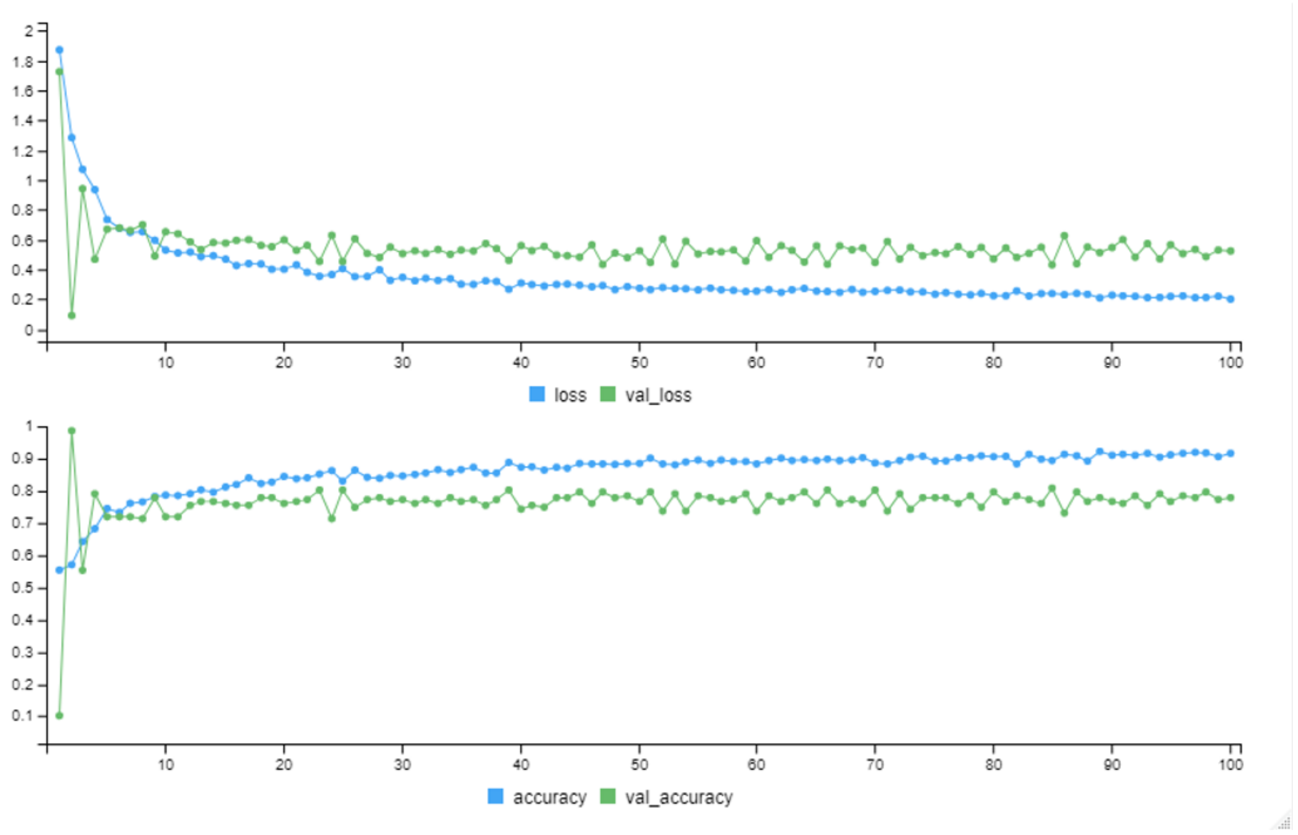
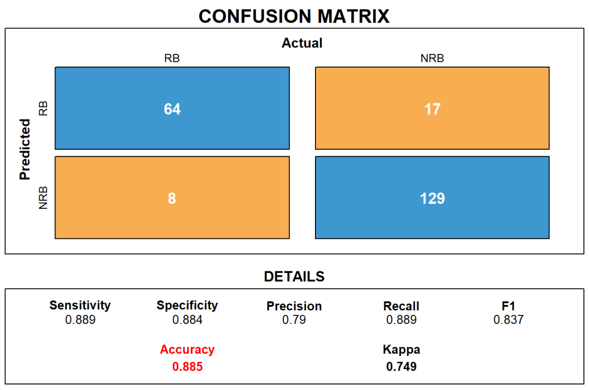


Figure 29. The training process of Neural Network with accuracy and loss



CONFUSION MATRIX
Actual
RB
NRB
Predicted
RB
NRB
64
17
8
129
DETAILS
Sensitivity 0.889
Specificity 0.884
Precision 0.79
Recall 0.889
F1 0.837
Accuracy 0.885
Kappa 0.749

Figure 30. The confusion matrix of the Neural Network model

some misclassifications can be avoided if that only happens in minority of models. However, this is not always the case, so we need to do many experiments to find the best combination of models.

However, some analysis on the choice of models can be done to avoid exhausting all possible combination. First, as we want a deterministic result, so we don't want to see a tie between models, then we can set the number of individual models used in our consensus models to be odd. Second, as can be observed in Table 8, for those models with over 85% accuracy, it is their performances on sensitivity that constrains their overall performance, thus we'd like models with better performance on sensitivity to be chosen in our consensus model. Therefore, from all models reaching over 85% accuracy, we selected kNN + D.S. as two fixed components in our consensus models and tried the rest of models one by one. This reduce the workload of all possible combinations quite a lot.

After implementing all 3-model consensus model, we listed several with the best results in Table 9. They both exceeded the performance of our best individual model-Neural Network. Among them, the combinations with two Ad-

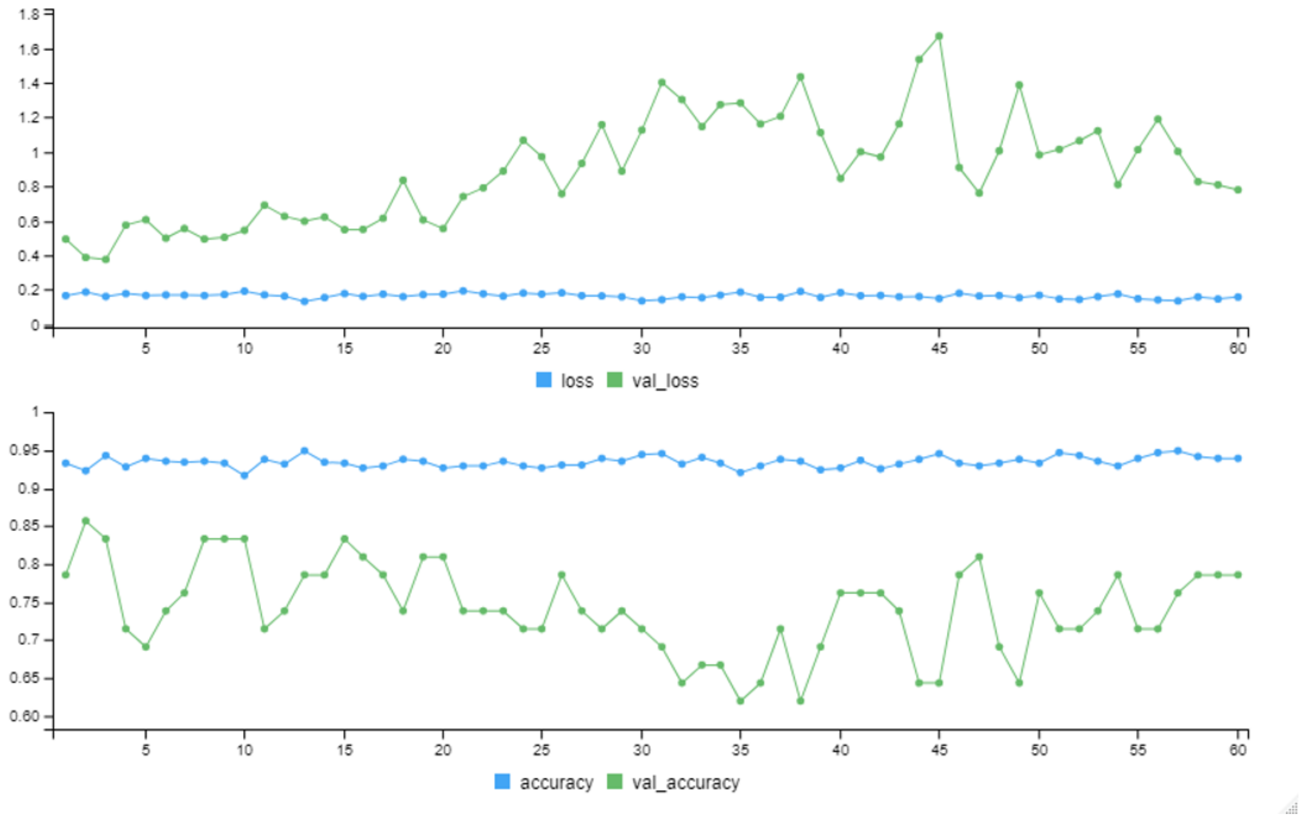
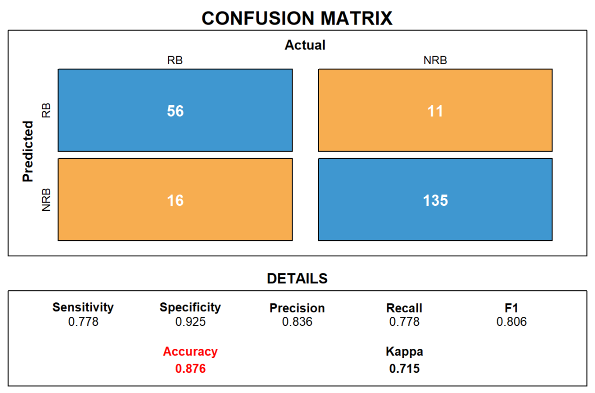


Figure 31. The training process of Deep Neural Network with accuracy and loss



CONFUSION MATRIX

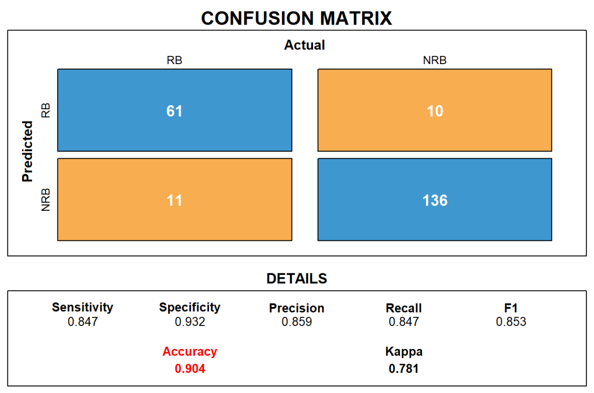
Predicted \ Actual	RB	NRB
RB	56	11
NRB	16	135

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.778	0.925	0.836	0.778	0.806

Accuracy	Kappa
0.876	0.715

Figure 32. The confusion matrix of the Deep Neural Network model



CONFUSION MATRIX

Predicted \ Actual	RB	NRB
RB	61	10
NRB	11	136

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.847	0.932	0.859	0.847	0.853

Accuracy	Kappa
0.904	0.781

Figure 33. The confusion matrix of consensus model C4

aBoost models received the best accuracy to 90.37 %. The confusion matrix for these two consensus models are shown in Figure 33 and 34.

The performance of our 3-model consensus models on sensitivity already achieved the almost the same as kNN + D.S. model. So what inspired us here is to give more votes to Neural Network model, as it is not only the best model in terms of accuracy but also the best model in terms of sensitivity. Therefore, we set that Neural Network model can have 2 votes in the major vote decision procedure. Corresponding to this change, we need to increase the size of our consensus model

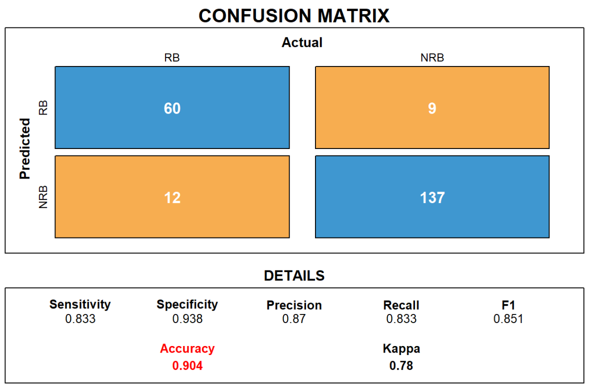


Figure 34. The confusion matrix of consensus model C7

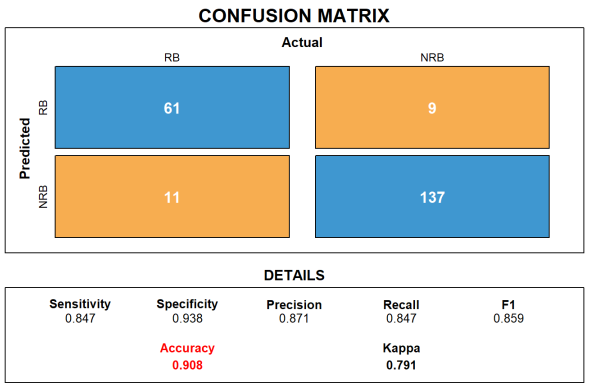


Figure 35. The confusion matrix of consensus model C8

to 5-model. In this sense, we need to choose another three models. The best choice of these three models seem to be obvious here, we just select kNN+D.S. and two AdaBoost model, which gave us the best performance on 3-model consensus model before.

After this combination of models, we successfully increased the accuracy of our prediction, as shown in Figure 35, to 90.83%, which is the best model among all.

5.2. Future Work

In this project, we have tried various machine learning models as well as consensus models which ensemble results from individual models. However, the best prediction results we gained is only about 91%. This accuracy has very large space to improve on.

First, in the original paper, a descriptor selection procedure is conducted to select specific variables used in each model. However, we are not able to reproduce this procedure here, as it requires a strong chemical knowledge to analyze those descriptors. Thus, this is a direction that can be further researched in the future.

Second, we noticed that in all of our well performed models, it is the performance on sensitivity that limited their overall performance. So this is also a direction we can think about to make our model stronger. We can try to find a model that gain very high sensitivity without dropping the specificity too much. Then we can combine it with models we already had to construct a consensus model, which is hopefully to achieve excellent performance on both sensitivity and specificity. However, it can't be guaranteed that it is easy to find such a model and the model can combine the existing models well to exhibit good enough overall performance.

Finally, a more realistic way, when we looked at the composition of the data is to increase the data size of RB molecules. Among all 1055 data points, 699 of them are NRB molecules, with RB molecules only half of NRB molecule. Thus it can account for why most of our models behave badly in term of sensitivity, or in other words, the ability of predicting RB molecules correct. Due to the limited number of RB molecules, our model can't grasp the property of RB molecules well. So we need a larger data set, or importing more RB-class data into our data set. This is hopefully to be feasible with more research in this area conducted.

Techniques	Accuracy	Specificity	Sensitivity
SVM + D.S.	85.3%	91.8%	72.2%
kNN + D.S.	87.6%	89.7%	83.3%
PLSDA + D.S.	85.3%	91.8%	72.2%
LDA	85.3%	91.8%	72.2%
Naive Bayes	72.9%	62.3%	94.4%
Tree	78.4%	75.0%	80.3%
Pruning Tree	81.7%	75.0%	85.2%
SVM	87.6%	93.2%	76.4%
Bagging	87.6%	93.8%	75.0%
Random Forest	88.1%	95.2%	73.6%
Adaboost (with Breiman coef.)	87.2%	92.5%	76.4%
Adaboost (with Freud coef.)	85.8%	91.1%	75.0%
Neural Network (NN)	88.5%	88.4%	88.9%
DNN	87.6%	92.5%	77.8%

Table 8. Individual model summary (D.S. = descriptors selection)

Consensus Model	Accuracy	Specificity	Sensitivity
C1: (SVM + D.S.) + (kNN + D.S.) + (PLSDA + D.S.)	87.61%	93.15%	76.39%
C2: NN + SVM + (kNN + D.S.)	88.99%	92.47%	81.94%
C3: NN + DNN + (kNN + D.S.)	89.91%	92.47%	84.72%
C4: NN + Adaboost (with Breiman coef.) + (kNN + D.S.)	90.37%	93.15%	84.72%
C5: NN + Random Forest + (kNN + D.S.)	89.91%	93.15%	83.33%
C6: NN + Bagging + (kNN + D.S.)	88.99%	91.78%	83.33%
C7: NN + Adaboost (with Freud coef.) + (kNN + D.S.)	90.37%	93.84%	83.33%
C8: NN * 2 + Adaboost (with Freud coef.) + Adaboost (with Breiman coef.) + (kNN + D.S.)	90.83%	93.84%	84.72%

Table 9. Consensus model summary (D.S. = descriptors selection)

References

[1] R. S. Boethling. Designing biodegradable chemicals. *ACS Symposium Series*, 640:156–171, 1996. 1

[2] European Communities. Regulation (ec) no 1907/2006 of the european parliament and of the council. 2006. 1

[3] K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni. Quantitative structure-activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information Modeling*, 53(4):867–878, 2013. 1, 2

[4] Milano Chemometrics and QSAR Research Grou. Qsar biodegradation data set, UCI Machine Learning Repository, 2013. `https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation`, donated on 2013-06-21. 2

[5] E. Rorije, H. Loonen, M Müller, G. Klopman, and Wjgm Peijnenburg. Evaluation and application of models for the prediction of ready biodegradability in the miti-i test. *Chemosphere*, 38(6):1409, 1999. 1