Classification models for protein ligandibility prediction

Jakub Bahyl[1]
MFF, NPFL054

(Dated: 31 January 2018)

In this work, I'm introducing various automatic predictors, trained on open development dataset of proteins, obtained from PDB database. The aim is to create the best predictor for protein ligandibility based on its physical and chemical properties. I took advantage of *Caret* package an applied these methods: *decision tree, random forest, AdaBoost*, and *Support Vector Machine*. All of them were tuned at their best, cross-validated and evaluated with accuracies, F1 scores and t-tests.

"In ML, where algorithms get published quickly and state-of-the-art frameworks are open-source, there isn't any first-mover advantage."
— François Chollet

INTRO AND MOTIVATION

Biological functions of proteins are performed by their interaction with other molecules (another proteins, ligands, etc.). Specifically, ligand is an ion or molecule that binds to a central atom to form a coordination complex. Usually there exist several number of points on protein surface with high probability of *ligandibility* - ability to bind a ligand.

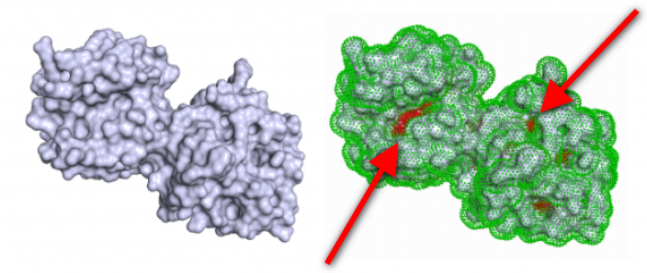


FIG. 1. Left: Generic protein. Right: Highlighted points with high ligandibility property

I have been provided with development dataset of selected proteins from public PDB database, overall containing 35 different attributes (such as *hydrophobicity, acidity, atom density*,..). Each protein (data aggregate) is labeled with an unique id and ligandibility classes ("Yes", "No") for each point of the protein surface.

The main task here is to develop the best possible classifiers and obtain the highest accuracy in forecasting ligandibility of new, yet unseen proteins. Quality of each model is represented in terms of cross-validated tests (data divided into *train* and *test* sets 90:10) and ROC curves.

I. BASIC DATA ANALYSIS

Development dataset contains overall 30 proteins with averagely more than 1500 surface points of 35 attributes. 27 proteins were used as a *training* set (44730 points) and the rest 3 as a *testing* set (5270 points). All confusion matrices, accuracies and F1 scores in this work were calculated with respect to the cross-validated *testing* set.

A. Table of proteins

Happily, none row contained any NULL data, so no cleaning was needed. Here I provide a table of all protein ids, number of surface points and ligandibility percentages.

Protein id.	Points	"No"%	"Yes"%
5	753	92.83	7.17
13	1965	96.64	3.36
22	864	91.90	8.10
25	1587	93.89	6.11
46	706	99.43	0.57
51	974	94.87	5.13
62	3404	99.68	0.32
84	1753	96.12	3.88
92	3432	98.14	1.86
95	1194	93.63	6.37
103	1584	99.37	0.63
104	1329	92.40	7.60
106	1100	97.82	2.18
112	1174	97.53	2.47
123	1091	96.06	3.94
131	1624	95.81	4.19
137	1654	98.07	1.93
139	1370	98.98	1.02
148	1761	97.27	2.73
151	1547	98.13	1.87
156	1661	96.81	3.19
163	1646	98.91	1.09
165	1645	96.41	3.59
171	2648	97.36	2.64
173	3340	96.56	3.44
180	1602	97.32	2.68
211	989	97.47	2.53
219	1056	98.48	1.52
222	1855	97.90	2.10
250	2692	98.03	1.97

No protein contained more than 10% of its are as *active*. Therefore the final testing of classifiers has to be performed with regard to determination of such areas. Accuracy parameter is no longer plausible and I'm reporting also F1 scores. Furthermore, the models should have better performance than the **Naive classifier** (always assuming areas as *inactive*):

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	0	119
	"No"	0	5270

Accuracy: **97.74%**, F1 score: **0%**.

B. Model 1: Decision tree

A set of simple decision trees was trained with wide variety of complexity parameters $cp = 2^n, n \in \texttt{Int}\langle -15, -4\rangle$. Here I provide the graph of *cvacc* (cross-validated accuracy) against various *cps*:

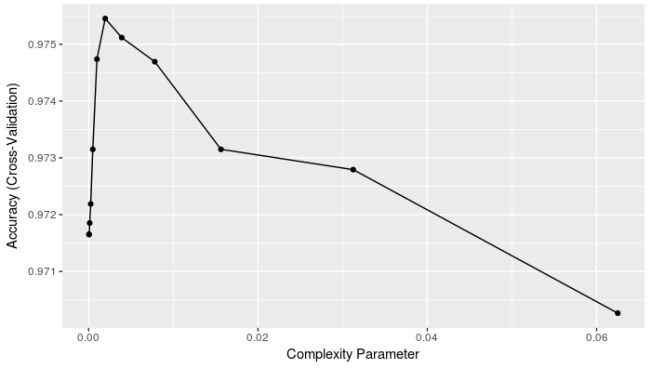


FIG. 2. Evolution of accuracy metric with increasing cp

The best *cp* is the one which minimises error: $bestcp = 0.0039$. After pruning, there emerged a simple tree graph:

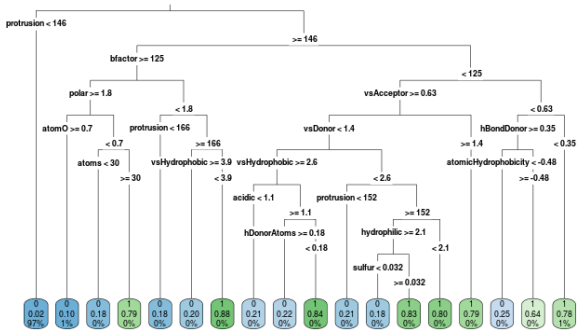


FIG. 3. Decision tree graph. Green areas denote decisions with high confidence.

Such tree model returned a confusion matrix (contingency table) for testing data as follows:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	51	68
	"No"	28	5123

In purpose of calculating accuracy and F1 with high confidence, I cross-validated training and testing set during each step and also adapted *bestcp*. The mean accuracy resulted as **97.18%** and F1 score **33.7%**.

C. T-tests of DT model

Cross-validation returned 10 results of accuracies and F1 scores following (at least we believe) normal distribution. After performing Student's t-tests, the obtained error intervals for the true mean accuracy of confidence 90%, 95%, 99% can be found in this table:

mean=**97.18%**	**t-tests for accuracy**		
Confidences	90%	95%	99%
Errors from mean	±0.48%	±0.60%	±0.86%

mean=**33.7%**	**t-tests for F1 score**		
Confidences	90%	95%	99%
Errors from mean	±10.6%	±13.1%	±18.8%

D. Analysis of DT graph

The DT graph on the left is not much readable, but provides at least some information about feature importance. The first big decision comes from the magnitude of *protrusion*. If $protrusion < 146$, there is only 3% error of classifying protein point as inactive. In other case there comes second decision about *bfactor*, and so on...

According to the tree graph, one can identify several attributes as promising model features: protrusion, bfactor, polar, vsAcceptor,...

Ensemble model will definitely tell more.

II. MODEL 2: RANDOM FOREST

Ensemble model such as Random Forest (RF) is operating on a multitude of decision trees, outputting mode of the classes. The first shot here wa training RF with default settings: *ntree*=500 (number of trees) and *mtry*=6 (number of features used for splitting at each tree node). The OOB (out-of-bag) error is falling with increasing number of used trees, as expected:

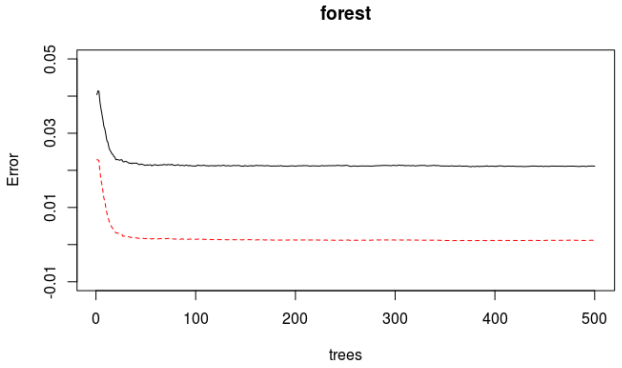


FIG. 4. Black line: OOB error. Red line: one of class' error.

The default *ntree*=500 is enough for the OOB error convergence and I used this value in the rest of RF simulations. Since RF automatically sets the parameter *mtry* as $mtry = \sqrt{35} \approx 6$ (square-root of number of features), I had to tune this a bit. This was performed (due to time complexity) with a smaller forest *ntree*=100 sample:

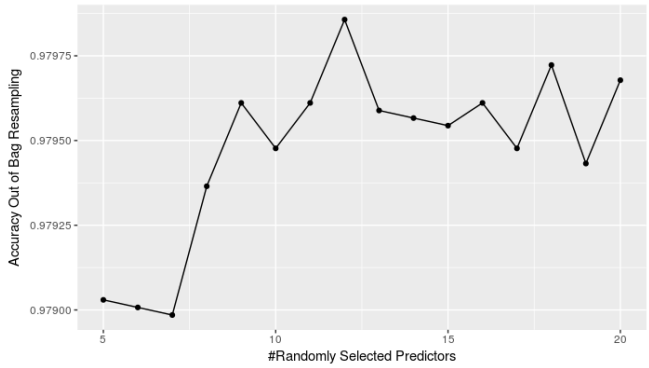


FIG. 5. OOB accuracy with increasing integer *mtry*

OOB accuracy reaches maximum around $mtry = 12$. This also provides a bottom estimate of the number of important features. Additionally, RF model (after retraining with proper parameters) also generated an importance map, where the scaled *importance values* are clearly visualised and ordered.

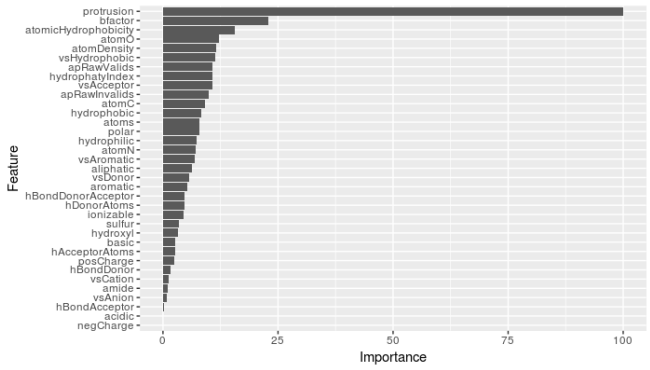


FIG. 6. Full feature importance map

Attributes *protrusion* and *bfactor* seem again as the most important features - RF is more or less consistent with DT. The same table with the top 10 most important features:

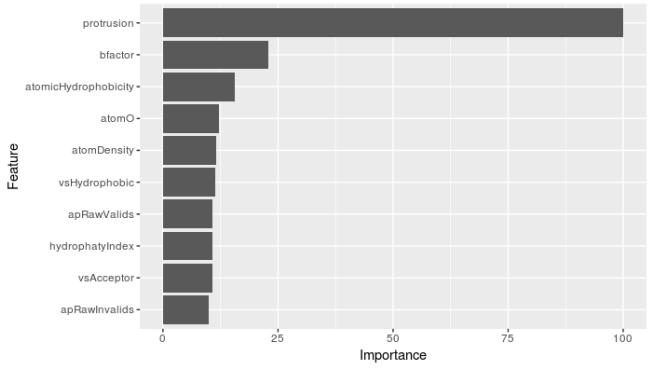


FIG. 7. Feature importance for the top 10 features

The resulting confusion matrix of tuned (*ntree*=500, *mtry*=12) RF yielded:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	43	76
	"No"	13	5138

RF's cross-validated accuracy mean: **97.24%** and F1 score: **33.5%** with confidence intervals:

mean=**97.24%**	**t-tests for accuracy**		
Confidences	90%	95%	99%
Errors from mean	±0.43%	±0.53%	±0.76%

mean=**33.5%**	**t-tests for F1 score**		
Confidences	90%	95%	99%
Errors from mean	±8.3%	±10.3%	±14.8%

III. MODEL 3: ADABOOST

Adaptive Boosting (AB) is another ensemble method combining decision trees predictors by adaptively weighting the samples. AB method contains three core parameters: *iter* - number of weak learners to train, *maxdepth* - maximum number of splits, ν - learning rate.

The first step is to measure their proper values through grid search:

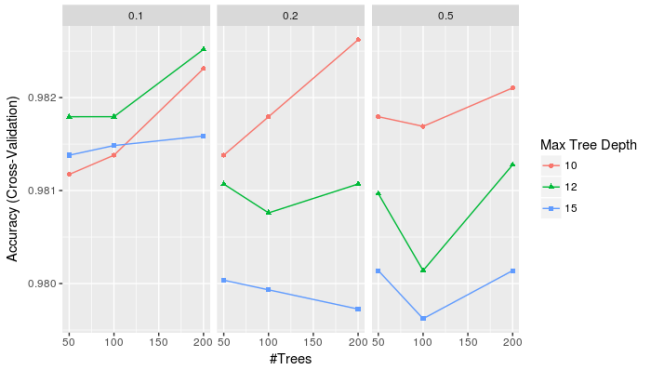


FIG. 8. Grid search for *iter*, *maxdepth* and ν

For further modelling I used the optimal values: *iter*=300, *maxdepth*=10 and ν=0.2. I chose bigger value of *iter* since it works similarly as in RF (more weak learners make higher accuracy).

The resultant confusion matrix for such tuned AB:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	36	83
	"No"	11	5140

After cross-validation on both training and testing side, the accuracy yields **97.16%** and F1 score **33.2%**. Confidence intervals:

mean=**97.16%**	**t-tests for accuracy**		
Confidences	90%	95%	99%
Errors from mean	±0.41%	±0.51%	±0.72%

mean=**32.2%**	**t-tests for F1 score**		
Confidences	90%	95%	99%
Errors from mean	±8.1%	±10.1%	±14.2%

After all, AB seems to have similar performance as RF. Although AB and RF are both ensemble algorithms, they are based on different principles. While RF is training many different DTs with many samples of data, AB is iterating the final model with many weak learners.

IV. MODEL 4: SUPPORT VECTOR MACHINE

The last used model was the well-known Support Vector Machine (SVM). I gave a shot to various type of SVM kernels - linear, polynomial (degree of 2) and radial. All of them were validated without cross validation. Their single-shot accuracies and F1 scores on testing data are:

Kernel	Linear	Quadratic	Radial
Accuracy	98.35%	98.27%	98.29%
F1 score	46.0%	61.6%	47.67%

Althought linear kernel seems to have the best accuracy and quadratical the best F1 score, I worked further with the radial one due to its capabilities of tuning.

A. Tuning SVM radial kernel

To determine radial SVM's tuning parameters such as σ (Gaussian exponential parameter) and *cost* (cost function value for soft margin), I ran tuning over grid of values for each one. The best choice was around $\sigma = 0.05$ and $cost = 12$.

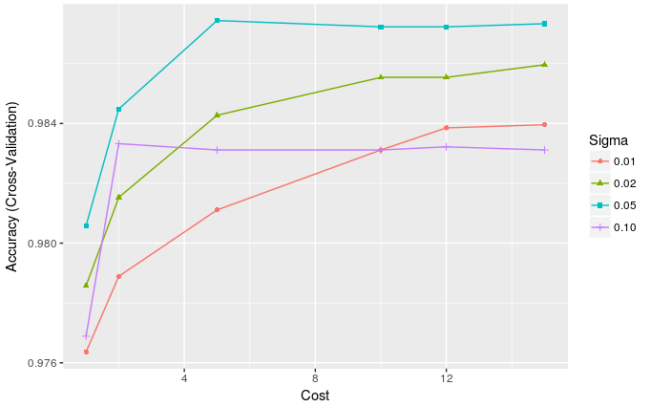


FIG. 9. Example of gamma and cost tuning.

The resultant confusion matrix for tuned SVM model:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	53	66
	"No"	37	5114

Cross-validated accuracy yields **97.05%** and F1 score **32.8%** which is still worse than RF so far. Confidence intervals:

mean=**97.05%**	**t-tests for accuracy**		
Confidences	90%	95%	99%
Errors from mean	±0.45%	±0.55%	±0.79%

mean=**32.8%**	**t-tests for F1 score**		
Confidences	90%	95%	99%
Errors from mean	±7.5%	±9.3%	±13.3%

B. Forward selection

In this part I tried to use the "feature forward selection" based on RF feature importance graph to create many radial SVM models with increasing number of used attributes.

I believe that not all of the provided attirbutes are useful for predicting ligandibility. The useless ones are just making noise and worsen the performance of classifiers.

In each iteration I'm adding one more attribute as a feature (according to the descending feature importance map) and watching the cross-validated accuracy of tuned radial SVM.

Feats	Acc	F1	Feats	Acc	F1
1	96.92%	38.8%	11	97.00%	40.5%
2	97.08%	39.1%	12	96.81%	37.9%
3	97.17%	39.3%	13	96.97%	41.5%
4	97.19%	39.0%	14	97.01%	**43.5%**
5	97.22%	39.4%	15	96.83%	40.7%
6	97.14%	38.9%	16	96.87%	41.6%
7	97.09%	38.7%	17	96.87%	41.1%
8	97.07%	40.1%	18	96.77%	39.6%
9	97.01%	39.8%	19	96.61%	35.8%
10	96.98%	40.0%	20	96.52%	35.9%

Here I stopped the algorithm since the accuracy and F1 score were already decreasing decreasing. The highest F1 value was obtained when the number of aused attributes was **14**, which is also consistent with the tuned *mtry* paramter.

Just for illustration I'm providing the graph of data projection on the two most important attributes *protrusion* and *bfactor*.

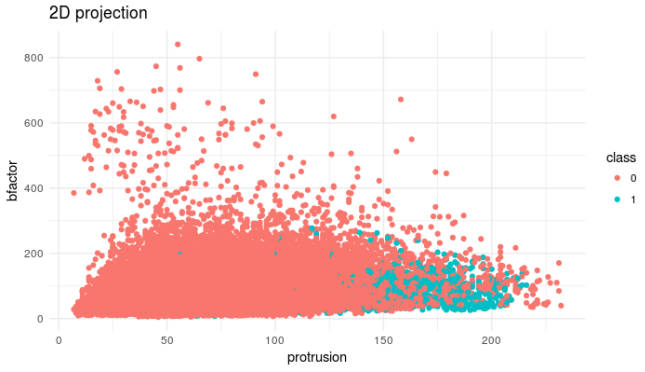


FIG. 10. ROC curves for all models in this work.

V. ROC EVALUATION

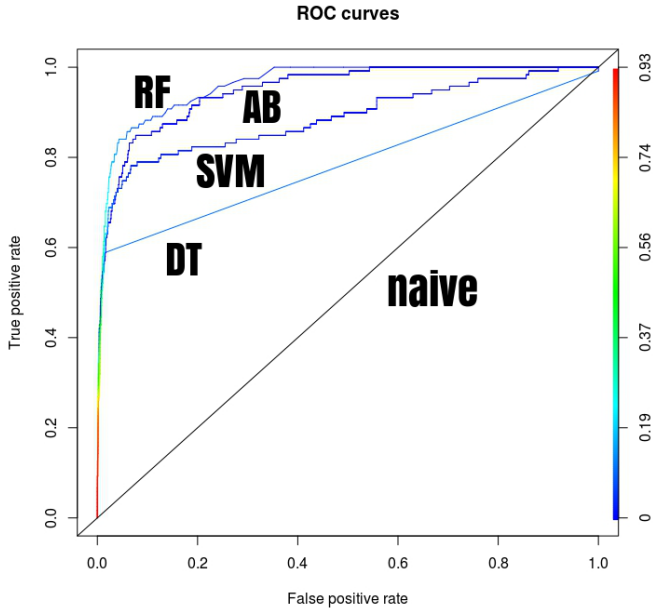


FIG. 11. ROC curves for all models in this work.

There is one interesting fact that even RF has better AUC than SVM, SVM got higher accuracy. This obviously may happen and it's the proof that ROC is not the best representation of model quality. Final table of accuracies and AUCs:

VI. CONCLUSION

Although it seems that forwarded SVM model was doing best and RF obtained the highest accuracy, we still have keep in mind the confidence intervals. These intervals for F1 score were usually in order of whole percents and therefore it's hard to tell whether the forwarded SVM model really achieved so that higher F1 score or not.

	Naive	DT	RF	AB	SVM	forw SVM
Acc	98.12%	97.20%	97.24%	97.16%	97.05%	$< 97.01\%$
F1	0%	34.5%	33.5%	32.2%	32.8%	$< 43.5\%$
AUC	0.5	0.79	0.96	0.95	0.89	–

VII. BONUS PART

In this part I worked with *dataset D*. There have been also 30 proteins, but with much less surface points ~ 200 and much bigger chance to contain active areas $\sim 50\%$.

A. Classification part

I tuned and trained once again RF, AB and radial SVM models. Tuned parameters were quite different ($mtry$=16 for RF, $maxdepth$=12 and $\nu = 0.5$ for AB, $\sigma = 0.1$ and C=12 for SVM).

Here I'm providing the final table of accuracies, F1 scores and AUC values. All results were obtained in the same way (methodology) as described in the earlier parts of this work.

	Naive	**RF**	**AB**	**SVM**
Acc	41.36%	66.02%	68.89%	62.09%
F1	0%	61.8%	72.9%	57.5%
AUC	0.5	0.76	0.73	0.72

Corresponding ROC curves:

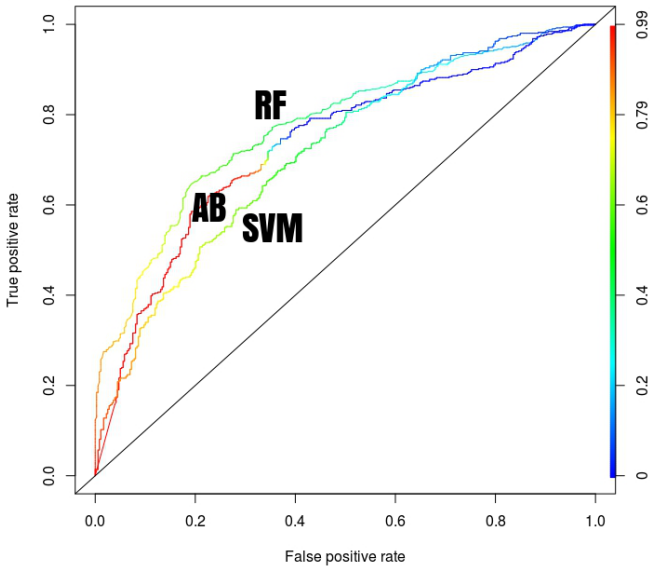
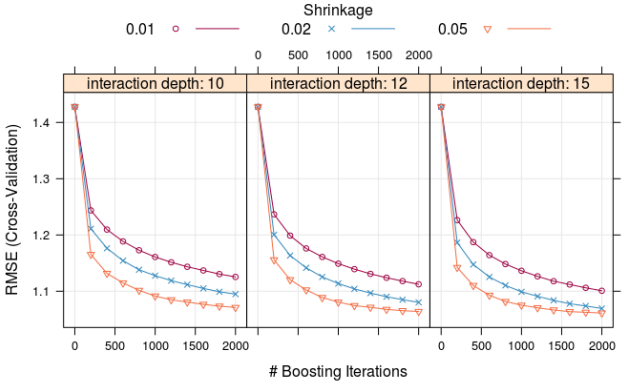


FIG. 12. ROC curves for all models in this work.

B. Regression part

The regression part was performed with Generalized Boosting Regression Modeling (GBM) in order to fit the training data according their *ligand_distance*. After all, I transformed the result a made the task as classifying again.

GBM produces a prediction model in the form of an ensemble of weak decision trees. It required 4 parameters tu be tuned: *interaction.depth* (15) - the maximum depth of variable interactions (complexity), *n.trees* (5000) - the total number of DTs to fit, *shrinkage* (0.05) - learning rate, and *n.minobsinnode* (kept on 10) - the min number of training set samples in a node to commence splitting.



The resultant confusion matrix for tuned GBM model:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	248	159
	"No"	111	350

Cross-validated accuracy yields **68.89%** and F1 score **72.2%** which seems to be better than RF and quite worse than AB. No explicit conclusion can be made.

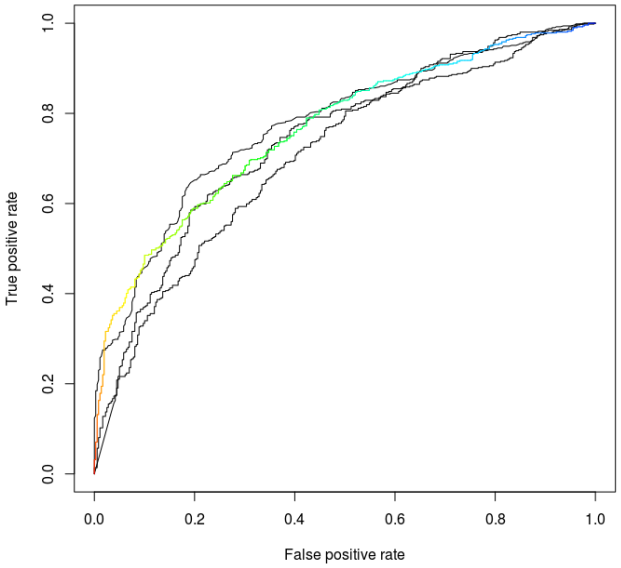


FIG. 13. The corol line coresponds to the GBM model.

In conclusion, AUC takes 0.76 which is exactly the same as in RF case.