Classification models for protein ligandibility prediction

Jakub Bahyl[1]
MFF, NPFL054

(Dated: 11 January 2018)

In this work, I'm introducing various automatic predictors, trained on development open dataset of protein points. List of basic applied methods are: *decision tree, random forest*, and *support vector machine.* All of them were tuned at it's best and cross-validated on development data.

Keywords: proteins, random forest, support vector machines

"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.

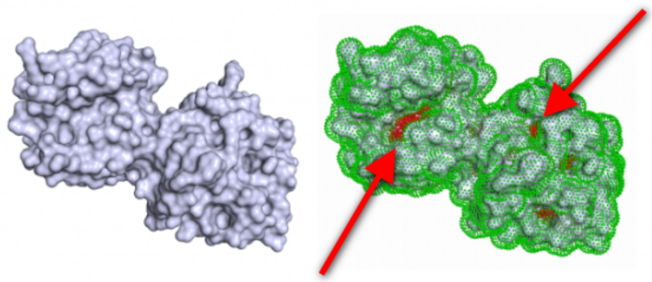

FIG. 1. On left: Generic protein. On right: Highlighted points with high ligandibility

I have been provided with development dataset of selected proteins, overall containing 35 different attributes (such as *hydrophobic, acidic, atom density,..*). There are also provided protein ids (aggregated data) and ligandibility class ("Yes", "No") for each point on each 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 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 > 1500 surface points of 35 attributes. 27 of them was extracted as a *training* set and the rest 3 as a *testing* set. Happily, none row contained any NULL data, so no cleaning was needed.

A. Table of proteins

Here I provide table of all protein ids, 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

In each protein there is no more than 10% of *active* area. Therefore we have to expect from the classifiers the ability to determine primarily such areas. Furthermore, they should be better than the **Naive classifier** (always assuming areas as *inactive*) - accuracy **98.11%**.

B. Model 1: Decision tree

A simple decision tree was trained with initially small complexity parameter $cp = 10^{-4}$. After training, tree graph was extremely dense, so the tuning of cp has to been done. Here is the graph of *xerror* (cross-validated error) against various cp:

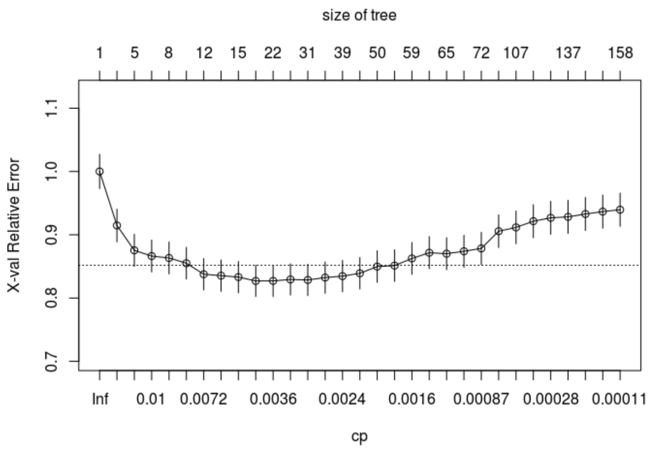

FIG. 2. Evolution of xerror with increasing cp.

The best cp is the one which minimises *xerror*: $bestcp = 0.00378$. After the pruning, the graph is smaller and more readable:

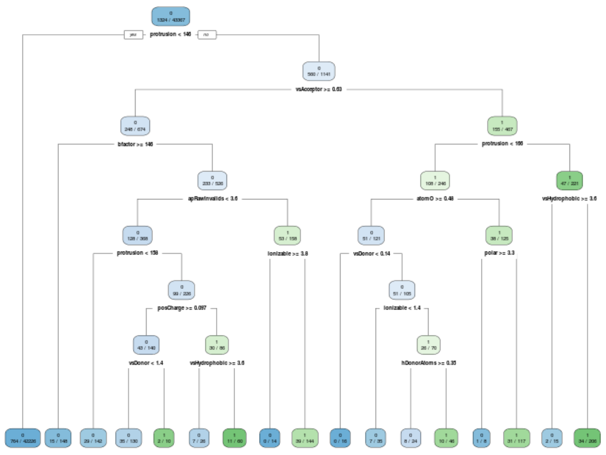

FIG. 3. Decision tree graph. Green areas denote decisions with high confidence.

Such tree model will return confusion matrix for testing data (6633 rows) as follows:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	54	64
	"No"	71	6444

The mean accuracy according to cross-validation is around: **97.09%**, which is quite worse, than Naive classifier. Apparently, instance-biased decision tree is not the wisest choice for protein dataset

C. T-tests of DT model

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

	t-tests		
Confidences	90%	95%	99%
Errors from mean	±0.41%	±0.51%	±0.74%

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 $764/42226 = 1.8\%$ error of classifying protein point as inactive. In other case there comes second decision about *vsAcceptor*, and so on. According to tree, one can identify several attributes as promising model features: protrusion, vsAcceptor, bfactor, vsHydrophobic,... Next model will tell more.

II. MODEL 2: RANDOM FOREST

Second model is an ensemble model operating on multitude of decision trees, outputting the mode of the classes. As first, Random forest (RF) was trained with default settings. The OOB error (out-of-bag error) decreases with increasing number of used trees:

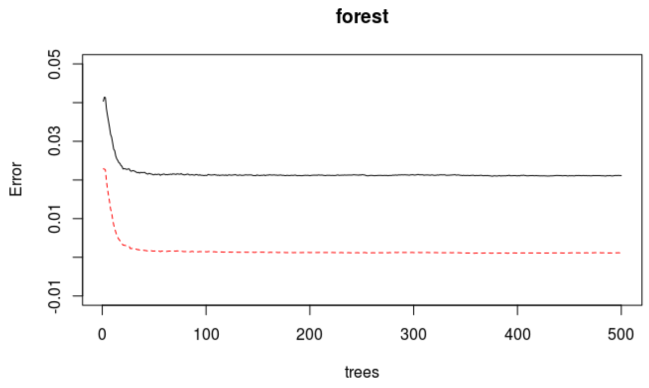

FIG. 4. Black line: OOB error. Red line: one of class' error.

Since RF automatically sets the parameter *mtry* - number of features randomly sampled as candidates at each split - as $mtry = \sqrt{\#features}$, the second step would be tuning this parameter. This was performed with smaller training sample:

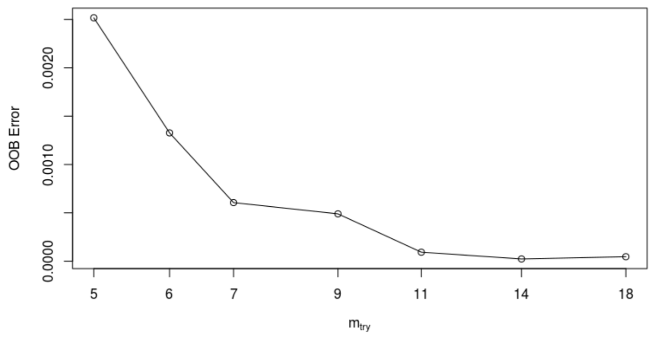

FIG. 5. OOB error according to increasing integer *mtry*

OOB reaches minimum around $mtry = 14$. This also provides estimate about the number of important features.

After re-training with tuned parameter, RF outputs confusion table:

Confusion		**Prediction**	
		"Yes"	"No"
Truth	"Yes"	59	41
	"No"	66	6467

RF's cross-validated accuracy is around: **97.30%** with confidence intervals:

	t-tests		
Confidences	90%	95%	99%
Errors from mean	±0.44%	±0.54%	±0.78%

Apparently, RF was doing better than single DT, but not exceeding Naive model's accuracy. RF also provides variable importance graph. This will be crucial in next section. According to this table, we can order attributes and gradually add them as features to other models.

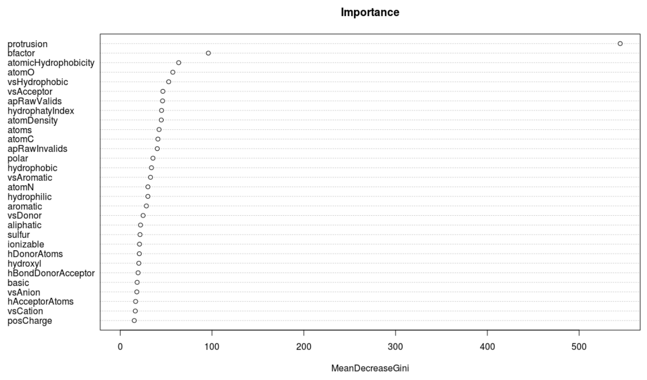

FIG. 6. Feature importance map.

Protrusion and *bfactor* seem again as the most important features. RF is more or less consistent with DT.

III. MODEL 3: SUPPORT VECTOR MACHINE

My third model was 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 trained on training data. Their default single-shot accuracies on testing data were:

Linear	Quadratic	Radial
98.11%	98.40%	98.42%

It's evident that accuracy of linear SVM is exactly as the Naive one's, quadratic SVM is quite better and radial is the best choice.

A. Tuning SVM radial kernel

To determine radial SVM's tuning parameters such as γ (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 $\gamma = 0.05$ and $cost = 2$.

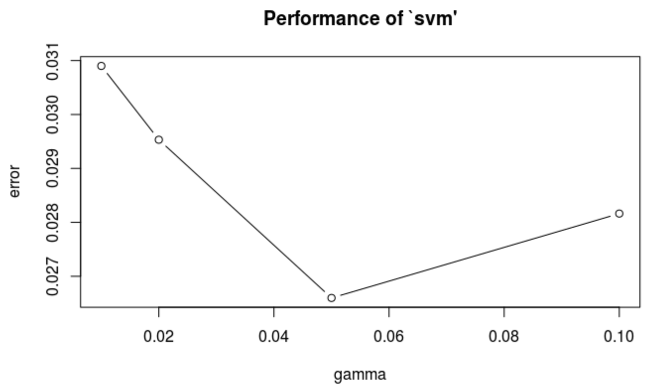

FIG. 7. Example of gamma tuning.

The resultant confusion matrix for tuned SVM model:

Confusion		Prediction	
		"Yes"	"No"
Truth	"Yes"	31	94
	"No"	26	6482

Cross-validated accuracy yields: **98.19%**, which is the best one so far. I'm summarising all performances at the end of article.

B. Forward selection

In this part I used the "feature forward selection" based on RF feature importance graph to create many SVM models with increasing number of used attributes. Since probably not all of them are useful for predicting ligandibility, the useless ones are only making noise. In each iteration I'm adding one more attribute as a feature and watching cross-validated accuracy of SVM.

Features	Accuracy
1	96.79%
2	96.90%
3	96.82%
4	96.99%
5	97.06%
6	97.07%
7	97.08%
8	97.15%
9	97.17%
10	97.33%
11	97.33%
12	97.29%
13	97.31%
14	97.32%
15	97.26%

Here I stopped the algorithm since the accuracy was decreasing. Maybe it would be worth of try to do the "backward" process instead of forward. (removing noise)

IV. ROC EVALUATION

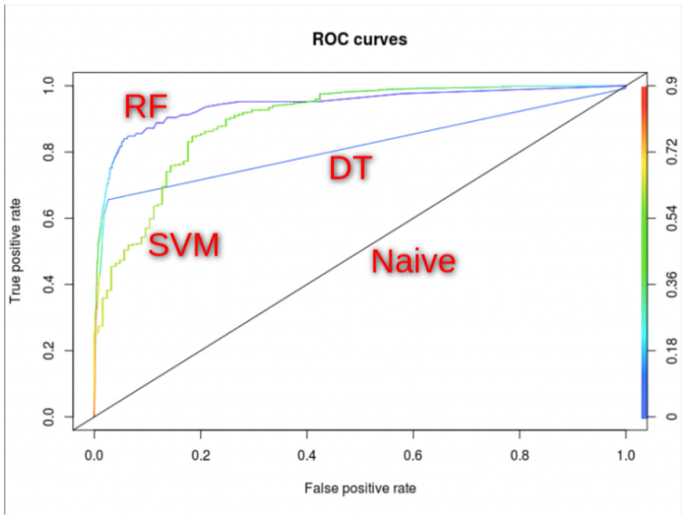

FIG. 8. ROC curves for all models in this work.

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:

	Naive	DT	RF	SVM	forw SVM
Accuracy	98.11%	97.09%	97.30%	98.19%	< 97.33%
AUC	0.50	0.82	0.94	0.90	-