NPFL054 Introduction to Machine Learning

Charles University, December 2018

Final Homework Exercise

A) GENERAL DESCRIPTION OF THE ML TASK AND THE PROVIDED DATA SETS

Introduction

You will do the task of *Virtual Ligand Screening* (VLS), which comes from the field of cheminformatics. Virtual screening is a computational method used in early stage of the drug discovery process. It searches very large libraries of molecules to identify a selection of molecules which are most likely to bind to a target receptor, i.e. the target of the examined drug. The selected molecules are then laboratory tested, which is financially demanding.

Ligands are molecules that can bind to a receptor, while *decoys* either bind only temporarily or do not bind at all. Based on previous research, a set of already recognized ligands and decoys is available and will be used as a *development data set*. You will use this set in the VLS task for developing predictors that classify molecules as either ligands or decoys.

Ligands, i.e. *active molecules* are necessary for living organisms. However, it naturally happens that some active molecules lose their effect and therefore they should be replaced. Since the human body is a dynamic environment, the active molecules are replaced by slightly different ones with similar but not really identical properties. For some of the new molecules, however, we do not know yet whether they are ligands or not. They form the *test data set* in your task.

Data sets

All data available for this task were obtained from our colleagues who specialize in bioinformatics and are related to one particular receptor called *AA2AR*. In both development and test data sets each example represents a small molecule and consists of a number of given quantitative characteristics, which are specific to each molecule and make a feature vector. The last attribute "`active`" represents the target value. In the development data sets each example is labeled with a true prediction, while the test set is *blind*, which means *without* true predictions. You should build an automatic predictor that, using a provided development data set of labeled examples, should be able to make good prediction also for the molecules in the (blind) test data set and decide whether they tend to be ligands (`1`), or decoys (`0`). So, your task is a binary classification task.

You will get a development data set `D`, which consists of two disjoint parts/subsets `D1` and `D2`. Using the development examples in `D` you should do all your experiments and tune your machine learning models. Finally you will choose your best model and use it for prediction on the given test set `T`. Then you will submit your prediction as a vector of binary values and it will be evaluated. The size of the provided data is given in the following table.

	Data set	File	Data set size
Development data (labeled examples)	D1	devel1.csv	6300 development examples
	D2	devel2.csv	2100 development examples
	D = D1 U D2		8400 development examples in total
Test data (blind)	T	test.blind.csv	1722 test examples

Why did we split the development examples into two parts and why do we make distinction between them? Technically, the reason is that the test data set T and the development data set D are *not* two representative samples of an *identical* population, although you should consider them to be *similar*. The test data set corresponds to the "new molecules" that were not recognized yet, and their population differs from the "known" population of D. The difference between D1 and D2 should help to tune your model towards the unknown population of T. In fact, the "new molecules" in T were collected by the same technique as the molecules in D2 but later on. Therefore they are more similar to the molecules in D2 than in D1. In the previous research, D2 was firstly used as a test set, and only then it was laboratory tested to confirm ligands.

Please keep in mind that data sets D1 and D2 look very similar, but have slightly different statistical properties. Generally you can mix them, but in some experiments you should use them separately and differently. The relationship between D2 and D1 is similar to the relationship between T and D.

Brief comment on features

All features are quantitative characteristics of molecules related either to their structure or to other chemical properties. Just for illustration here is a short description of several of them:

- MolWt – The average molecular weight of the molecule
- HeavyAtomMolWt – The average molecular weight of the molecule ignoring hydrogens
- ExactMolWt – The exact molecular weight of the molecule
- NumValenceElectrons – The number of valence electrons the molecule has
- NumRadicalElectrons – The number of radical electrons the molecule has
- FractionCSP3 – The fraction of carbons that are sp3 hybridized
- HallKierAlpha – Hall-Kier alpha value for a molecule
- HeavyAtomCount – Number of heavy atoms a molecule
- NHOHCount – Number of NHs or OHs
- NOCount – Number of Nitrogens and Oxygens
- Num(*) – Number of certain pattern/atoms/(*) in a molecule

Technical hints

- To load the data sets quickly into the memory use `fread()` (R package `data.table`).
- To learn your predictors use the following R packages: `rpart` for Decision Trees, `randomForest` and `adabag` or `fastAdaboost` for ensemble learning, `glmnet` for (regularized) logistic regression, and `e1071` for SVM.
- Variable importance values are computed by methods implemented in the mentioned packages:
    - `importance()` function extracts variable importance measures as produced by `randomForest`
    - `boosting()` function directly gives the importance values in `model$importance`
- To draw ROC curves and to compute AUC values we recommend using package `ROCR` and function `performance()`.

B) EXACT SPECIFICATION OF YOUR TASKS

Part 1 – Data analysis and feature filtering

1a) Look at the proportion of the binary target values in D1 and D2. You can assume that in test set T this proportion is approximately the same.

1b) Determine the number of discrete and continuous features.

1c) Some of the discrete features can be constant. Remove them. How many features remain?

1d) For each discrete feature determine the number of different values. Make a table that shows how many features have a certain number of values, like this

number of values	2	3	4	. . .
number of features	?	?	?	?

Then visualize the numbers using a barplot.

1e) For a *binary* feature A define fr(A) = #{**x** | A(**x**) > 0}, which is called *feature frequency* (in a training data set). A(**x**) is the value of A in the feature vector **x**.

Then remove all features that do not fulfil the condition min(fr(A), *n* – fr(A)) ≥ *frt*, where *n* is the training set size, and *frt* is a parameter called *frequency threshold*. For this exercise choose *frt* = 4 as the default. *Optionally*, you can later test the influence of different *frt* values on your experimental results.

Now, what is the number of discrete features that you will use?

1f) For each discrete feature A compute its *information gain,* i.e. the mutual information I(A;C) between the feature A and the target class C. Then sort all discrete features decreasingly according to information gain and draw a nice plot that shows its distribution.

Part 2 – Baseline model for automatic classification

Important general remarks (valid also for Parts 3 and 4)

- Everytime when you run a cross-validation process, you should *randomly* divide your data into a given number of folds. Since the positive and negative examples are highly *unbalanced*, you should make the division *carefully*. You should always keep the *identical number of positive examples in all your folds.*
- We should keep in mind the purpose of this classification task. Molecules from the test set that are classified as ligands are to be laboratory tested, which is rather costly. This is why your models for classification should prefer high precision. Hence to optimize your model you will try to maximize a particular area under the ROC curve rather than to simply minimize error rate. Since precision naturally decreases with increasing FPR, you will measure and optimize the AUC just up to FPR ≤ 10%, hereafter denoted by $AUC_{0.1}$.

  Technical hint: Using function `performance()` for computing AUC you can apply and set parameter `fpr.stop=0.1`.
- Whenever you compute confidence intervals (CI) for mean in this exercise, use CI based on t-test and set the significance level α = 5%.

2a) Think about the particular threshold FPR ≤ 10%, how it relates to precision. If FPR was more than 10%, what could be precision then?

2b) Build a Decision Tree (DT) model for binary classification using `rpart()` function. Use only D1 for training and evaluate it using 10-fold cross-validation. You will always have 5670 training examples and 630 test examples. In each run of the CV process compute $AUC_{0.1}$. Then report its mean, standard deviation, and confidence interval.

2c) Tune the complexity parameter (`cp`). Use the same training data as in 2b) and optimize the mean of $AUC_{0.1}$. Report the mean of $AUC_{0.1}$, its standard deviation, and confidence interval for the mean for different cp values, and arrange the results in a nice table. Then choose an "optimal" cp value.

> *Recommendation*: First find the cp value for which the mean of $AUC_{0.1}$ reaches its maximum, and then choose a maximum cp value for which the mean of $AUC_{0.1}$ does not decrease more than by 1SE below its maximum. SE stands for *standard error*, estimated as sample standard deviation divided by square root of the sample size.

2d) Now use the tuned cp value and train your DT model using the whole D1 set. Then make prediction on D2. Again, compute $AUC_{0.1}$. Is it in line with your estimate computed in 2c)?

Part 3 – More advanced models

In this part different students will work with different ML methods. Details will be delivered by email. Anyway, you will do the following.

3a) Run a particular learning method, evaluate it and tune its parameters to maximize $AUC_{0.1}$.

3b) Draw a plot that shows how the performance depends on selected parameters.

3c) Compare the results with the baseline model obtained in Part 2. – Which models are better and why?

Then you will use your tuned model in the following Part 4 and Part 5.

Part 4 – Experiments with different data sets

Now you will do more experiments with your model and you will examine the influence of different training sets and different test sets on the model performance. In each experiment you will do 5-fold cross-validation. The following table shows the division of training and test subsets that will be used in each CV run. Experiments 4b) and 4c) use a mix of D1 and D2 for training.

Experiment	**CV – training set**	**CV – test set**	**Estimation of performance ($AUC_{0.1}$)**
4a)	4/5 D1	1/5 D1	CV average and whole D2 as test set
4b)	D1 + 1/5 D2	4/5 D2	CV average only
4c)	D1 + 4/5 D2	1/5 D2	CV average only
4d)	4/5 D2	1/5 D2	CV average only

Experiment 4a) is almost the same as that in Part 2. They differ only in the number of CV folds. Here D2 serves as a test set. Since other experiments do not have separate test sets, their performance is estimated only by the cross-validation mean. You should always report the mean of $AUC_{0.1}$, its standard deviation, and confidence interval for the mean. Results should be arranged in a nice table.

4e) Now look at all your results and compare them. What is your conclusion? What did you learn about the data sets D1 and D2? Can you explain the differences of the results?

> *Important*: Keep in mind that finally you want to use both D1 and D2 and develop a model that should be as good as possible at making predictions on the blind test examples, which are more similar to examples in D2 rather than to those in D1.

Part 5 – Final model selection and prediction on the blind test set

In the previous parts you were more or less forced to do exactly described things. Now, here we are at the most creative part of this exercise.

5a) Build your final model for ligand prediction, which will be evaluated using the blind test set T. To train your final model you can use even whole development data sets, or any part of them. Take into consideration all your previous knowledge and experiments and make your decision about using training data and parameters of your model as well.

When your "best model" is trained, take the blind test set T and compute three prediction vectors using three different prediction cut-off values:

1. set the cut-off so that you get just 50 predicted positives (= vector T50)
2. set the cut-off so that you get just 150 predicted positives (= vector T150)
3. set the cut-off so that you get just 250 predicted positives (= vector T250)

All three vectors with predictions will be submitted to the contact teacher, who will compute their precision as the evaluation measure. The better precision values, the better predictor!

> *Remark on the meaning of this kind of evaluation*: Imagine that you need to find ligands among the molecules in the test set. Each molecule that is classified as ligand using your predictor has to be laboratory tested. Since you have a limited budget, you can try only a limited number of molecules (50, or 150, or 250). Hence, the higher precision, the more ligands discovered for the given money.

5b) Describe *how* you built your final model and *why* – write explicitly your main reasons for your choice.

5c) What is your *estimate* of the model performance on the test set T? Are you able to estimate precision of the three required predictions? How can you compute it? Try to do it and report three estimates of precision for T50, T150, and T250. These estimates will be compared with the truth.

5d) How would the evaluation using the prediction vectors T50, T150, and T250 change if we used recall instead of precision?

Important technical remark

- The prediction vectors should be submitted in separate plain text files named

  ```
  T.prediction.50
  T.prediction.150
  T.prediction.250
  ```

  The number of lines in each file should be exactly equal to the number of examples in the test data set T. There should be just one number (`0/1`) on each line.

C) FINAL REMARKS

The hard deadline for your submission is January 8, 2019. You should submit a report with your solution by email to holub@ufal.mff.cuni.cz. Later submission will be penalized.

Important directions

Make just one zip file named `surname_name_hw3.zip`. The submitted zip file should contain the following obligatory items:

- Text document (pdf file) with all answers, tables, plots, etc.
    - Be as clear as possible and explicitly answer all explicit questions required in this specification.
    - Do not change the order of the questions/subtasks in this specification and subsequently answer 1a–f), then 2a–d), then 3a–c), then 4a–e), and finally 5a–d).
    - Each of the Parts 1–5 should start at a new page.
- All source files you used to prepare the pdf document (.odt, .docx, .ods, .xlsx, .tex, etc.)
- All R codes that you wrote and used
- Any other illustrative attachments/appendices of reasonable size, if you want/need
- Three prediction vectors in the strictly required format (see Part 5)

Scoring

In total you can get 50 points at maximum. The proportion according to Parts 1–5 is 10–10–10–10–6. Moreover, we add up to 4 points for the overall assessment of your report – its good structure, clarity, nice (typo)graphics, and language.