INTRODUCTION TO MACHINE LEARNING (NPFL054)
Homework #3

Name: Kateřina Břicháčková

Year: 2018/2019

Date of submission: 8.1.2019

PART 1 – Data analysis and feature filtering

1a)

The proportion of the target values in both D1 and D2 is **1:20**.
In D1, there are **6000** negative and **300** positive examples.
In D2, there are **2000** negative and **100** positive examples.

1b)

The number of **continuous** features is **16**. There are **104 discrete** features. Total number of features is 120.

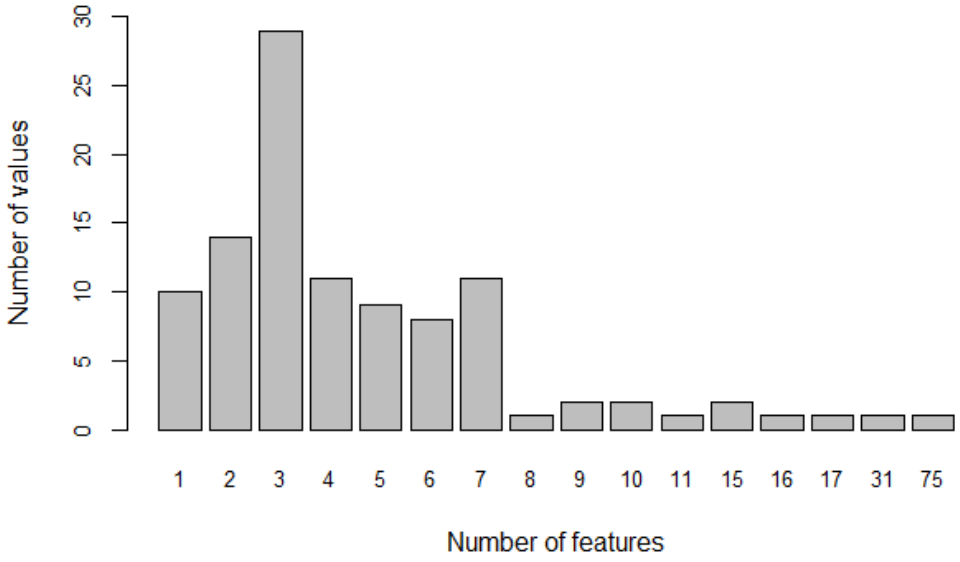
1c)

There were **10 constant** discrete features. After removing, **110 features remain** (94 discrete + 16 continuous)

1d)

Counts of different values counts for discrete features:

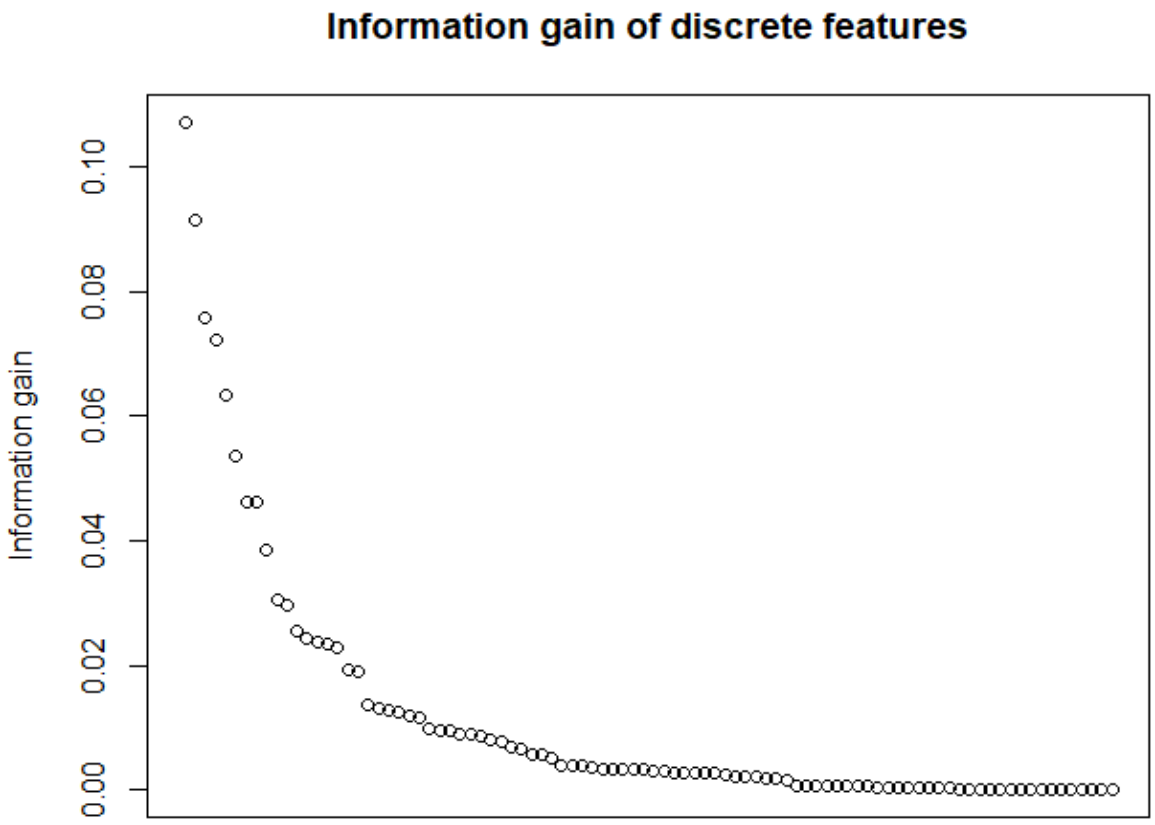
Number of values	1	2	3	4	5	6	7	8	9	10	11	15	16	17	31	75
Number of features	10	14	29	11	9	8	11	1	2	2	1	2	1	1	1	1



1e)

Two features, *"fr_phos_acid"* and *"fr_phos_ester",* were filtered out, so **92 discrete** features remained. Total number of remaining features is 108.

1f)



The graph pictured here is without x-axis labels, as it was too wide for this document. For better readability, see attached file "*1f_graph.png*".

PART 2 – Baseline model for automatic classification

2a)

Proportion of all negatives is 20/21, proportion of positives is 1/21. This proportion is fixed. The best precision is when we have 100% recall. If FPR is 10%, then we have 2/21 (10% out of 20/21) false positives and 1/21 true positives. In this point, precision = TP/(TP+FP) = 1/3. Thus, in ideal case it can be 1/3. If we go further with FPR, maximal precision goes down.

2b)

Results for the **Decision tree model with default parameters** in 10-fold cross-validation:

- mean of $AUC_{0.1}$: 0.0869
- standard deviation: 0.0064
- confidence interval for the mean: 0.0823 – 0.0915

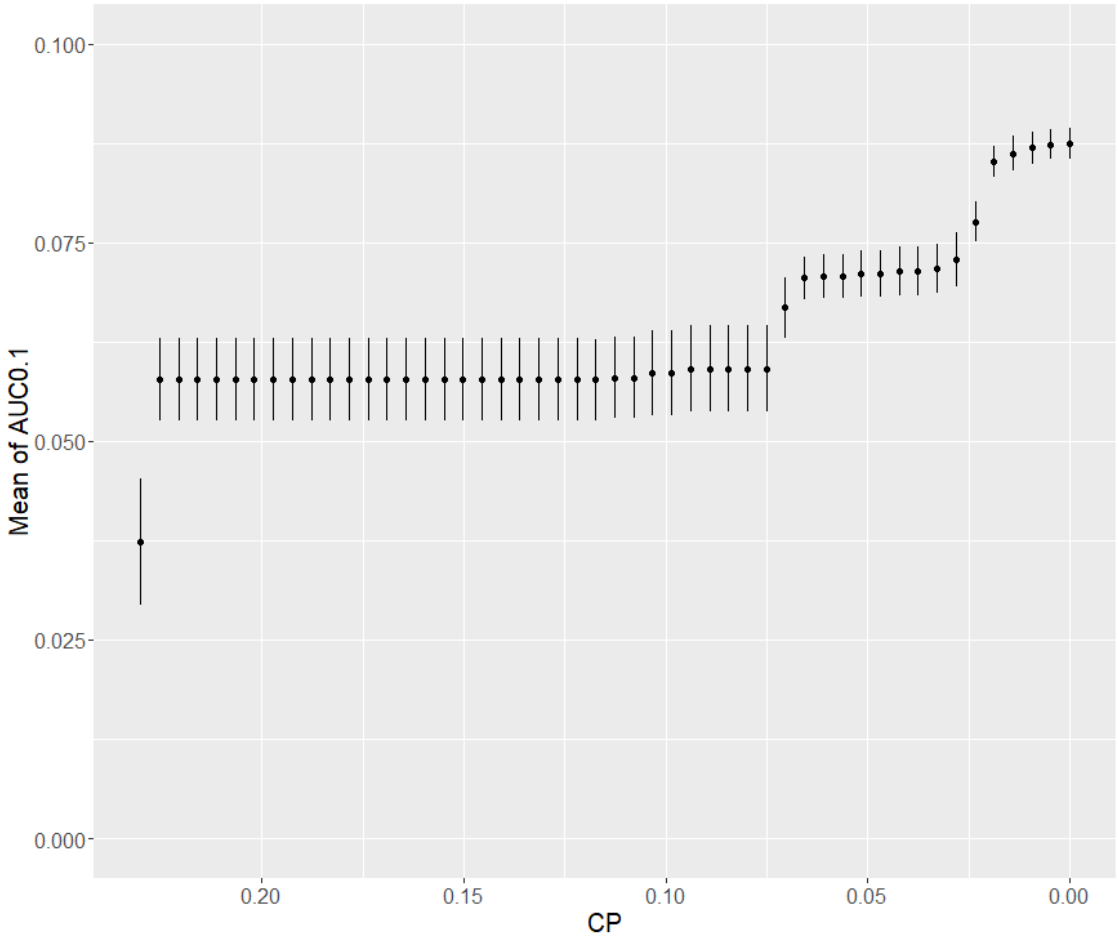
Default model trained on D1 and tested on D2:

- 0.0301

From the results, we can see that the performance is much worse when tested on D2 data set. It agrees with our hypothesis that D1 data set is different from D2.

2c)

In this plot, we can see dependency of $AUC_{0.1}$ mean from 10-fold cross-validation on the cp parameter, plus its standard error.



Results for the most significant cp values are given in the following table:

CP value	Mean of AUC 0.1	Standard deviation	Standard error	Confidence interval for the mean
0.23	0.0373	0.0258	0.008	0.0192 - 0.0554
0.17	0.0577	0.0164	0.0052	0.0459 - 0.0694
0.07	0.0668	0.0122	0.0038	0.0581 - 0.0754
0.05	0.0711	0.0093	0.0029	0.0645 - 0.0777
0.023	0.0776	0.008	0.0025	0.0718 - 0.0834
0.014	0.0862	0.0069	0.0022	0.0812 - 0.0912
0.0047	0.0873	0.0059	0.0019	0.0830 - 0.0916
0.00001	0.0875	0.0062	0.002	0.0831 - 0.0919

As we can see, **the lower the cp, the higher the mean**. Normally, the AUC mean should be lower for the smallest cp values because of overfitting. This result tells us that the data is very similar to each other and that even the most complex models are good enough. However, we suppose that the D2 data set has slightly different properties and it could be a problem if we chose too small cp value.

Thus, for next experiments, I chose the **best cp value to be 0.23**, which is not the best one according to the graph, but theoretically should be better when tested on D2.

For comparison, I also tested cp 0.14 and 0.05 in 2d).

2d)

Three DT models with different cp values were trained using the whole D1 set and tested on D2. The results are following:

CP value	AUC 0.1
0.23	0.0374
0.14	0.037
0.05	0.3556

It agrees with the assumption made in 2c), the best cp value really seems to be 0.23. The performance is a bit better than with the default cp parameter (see 2b), $AUC_{0.1}$ was 0.0301 there), but it is still rather poor. It is not in agreement with the estimate from 2c) and again, it shows us that the D2 data set differs from D1.

PART 3 – Logistic regression (LogR)

3a-b)

Firstly, the **LogR model without regularization** was created. The results of the **10-fold cross-validation** are following:

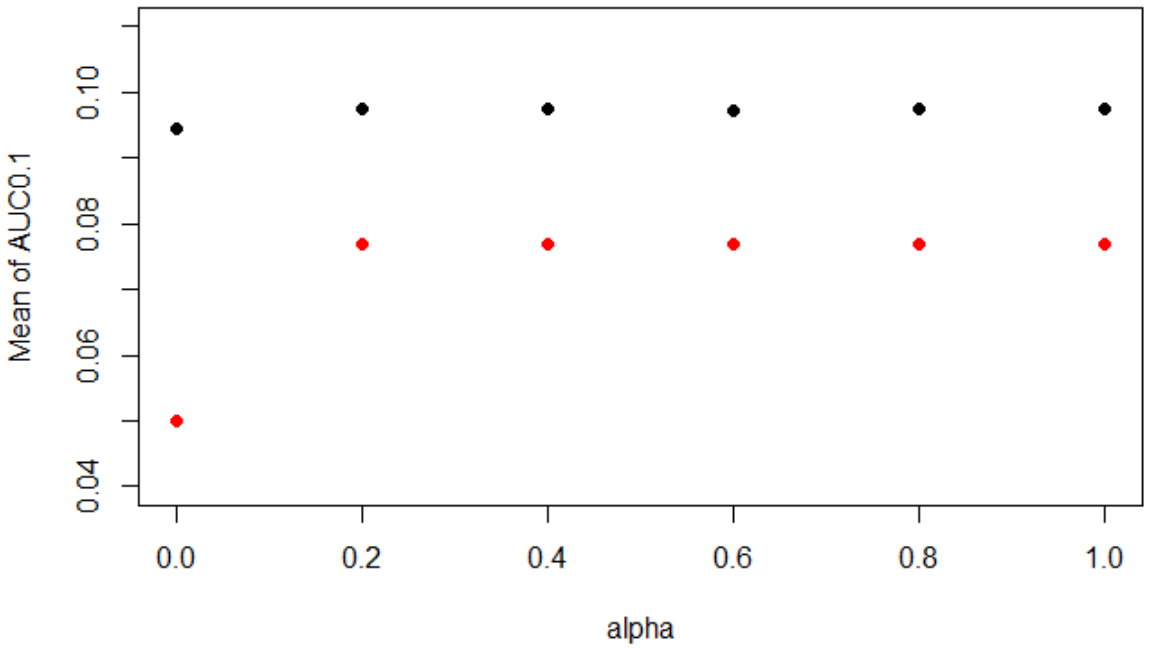
- mean of $AUC_{0.1}$: 0.087
- standard deviation: 0.0063
- confidence interval for the mean: 0.0831 – 0.0916

LogR model without regularization trained on D1 and tested on D2:

- 0.067

Then, the elastic regularization model was created with different values of alpha parameter. For each alpha, I run the cross-validation process and in each step, I took the best $AUC_{0.1}$ from all models that were build (for 100 different lambda values).

Here is the plot that shows dependency of $AUC_{0.1}$ mean on the alpha parameter. Black points show the mean of $AUC_{0.1}$ in the cross-validation. The red points show $AUC_{0.1}$ when trained on D1 and tested on D2:



As we can see, the results are more or less the same. The performance for alpha = 0 is the worst. The difference is small for the CV results, but it is significantly bigger for the test set. Let's take **alpha = 1** as the best one – the performance seems to be very similar (except when we are close to alpha = 0) and it is better to have **less complex** model.

Finally, let's build **the logR model with alpha = 1** on the whole D1 set and test it on D2:

- $AUC_{0.1}$ = 0.077

This result was achieved with lambda = 8.62e-05.

3c)

Here is the comparison of models performance:

Model	AUC 0.1 - CV mean	AUC 0.1 - test on D2
Default DT	0.0869	0.0301
Tuned DT (cp = 0.23)	0.0776	0.0374
LogR without regularization	0.087	0.067
LogR with regularization (alpha = 1)	0.0974	0.077

In both methods, the **models with tuned pamaters have better performance** when tested on D2.

When we compare the default DT model and logR model without regularization, the results of the cross-validation are practically the same. However, when tested on D2, the performance of the default DT model is significantly lower.
Logistic regression seems to have **much better** performance in both cross-validation and testing on D2.

The **best model** seems to be the **LogR model with regularization with alpha parameter set to 1**. It has very high $AUC_{0.1}$ mean in CV and it also performs pretty well for the test data set.

PART 4 – Experiments with different data sets

This table shows results of 5-fold cross-validation with the LogR model (alpha=1), using different training and test sets:

Experiment	CV - training set	CV - test set	CV - Mean of AUC 0.1	Standard deviation	Confidence interval	Test on D2 - AUC0.1
4a)	4/5 D1	1/5 D1	0.0972	0.0013	0.0956 - 0.0988	0.077
4b)	D1 + 1/5 D2	4/5 D2	0.0866	0.0018	0.0843 - 0.0889	X
4c)	D1 + 4/5 D2	1/5 D2	0.0902	0.0063	0.0824 - 0.0980	X
4d)	4/5 D2	1/5 D2	0.094	0.0021	0.0913 - 0.0966	X

Again, the results agree with our hypothesis **that D1 and D2 data sets have slightly different statistical properties**. The best mean of $AUC_{0.1}$ was achieved when we did not mix examples from D1 and D2 together. On the contrary, in 4b), only a small part of D2 was used and the estimation of performance is the worst one.
We also learned from all the parts that **examples in D1 are very similar to each other**. The $AUC_{0,1}$ mean is almost maximal (close to 0.1) for the LogR model and the performance grows with smaller cp parameter for the DT model. I assume that it will be the same for the D2 data set, according to the 4d) experiment – the mean of $AUC_{0.1}$ is also close to 0.1.

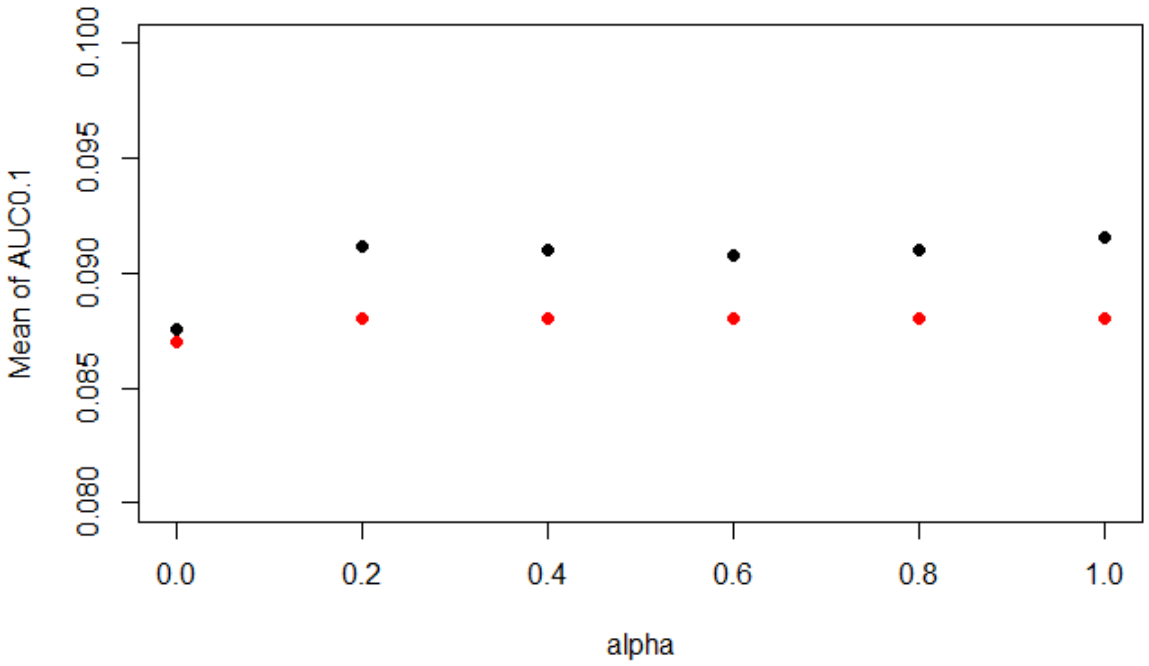
PART 5 – Final model selection and prediction on the blind data set

5b)

I decided to build the **LogR model with regularization** and tune its parameters. LogR model performance was much better in all cases. For all the reasons mentioned earlier (mainly in Part 4), I decided to use **only D2 data set for the training**. In my opinion, it is better to use only the best (the closest) examples we have and that adding examples from D1 would only make the performance worse.

I build the model the same way as in exercise 3), but I did **5-fold cross-validation**, as there are fewer examples in data set D2 than in D1. I did CV for 6 different values of alpha and reported the mean of $AUC_{0.1}$.

Then I tested the performance on D1 data set. If we assume that D1 differs from D2 as much as D2 differs from T, then it can make good sense to use D1 as test set. The results of the experiment are shown in following graph (test results in red):



As we can see, both train and test performances look satisfying. $AUC_{0.1}$ means are very similar to the exercise 3) and test values seem to be even higher.

As in exercise 3), let's choose **alpha = 1** for the final model. The **corresponding lambda value** (with the best performance on D1) **is 0.001013061.**

5c)

If we assume that the "distance" (dissimilarity) of D1 from D2 is similar to the distance of D2 to the blind test, we could estimate precision of our predictions using D1 as test set. The problem is that D1

is much bigger than T (D1 - 6300 examples, T – 1722 examples), and thus we cannot predict precision on the whole D1 set, as there are three times more positives (if we assume that the proportion of positives/negatives is the same).
I sampled D1 test to get a subset with the same number of examples as in T set, and with the same proportion of positives / negatives. Thus, in the subset there are 86 positives and 1636 negatives. I did 5000 iterations of sampling and computed means of precisions. Final values are following:

- Estimated precision for T50: **0.946**
- Estimated precision for T150: **0.522**
- Estimated precision for T250: **0.331**

We can assume that precision will be a bit worse for T, as lambda parameter is optimized on D1 and not T.

5d)

We optimized model building for the highest precision. It means that the predictor recognizes positives with high success. However, there can be more false negatives, because the predictor is more careful with classifying positives.

If the data set T has the same proportion of positives and negatives as D1 and D2, there should be 86 positives in T. It mean, for T50, there will be at least 36 false negatives, which makes recall lower – in the best case, the recall will be 50/86.

For T150 and T250, recall should be very high, because there should not be many false negatives.

In my testing, recall was around 0.5 for T50 and over 0.9 for T150 and T250.