My solution to the Loan Default Prediction Competition

Josef Feigl

1 Summary

I used a two-step-process to predict the loss. At first I predicted the probability for a default (PD). The so called $goldenfeatures$ posted by Yasser Tabandeh in the forums were crucial for this step. The PD prediction itself was done by two models: gradient boosted trees and a regularized greedy forest. In the second stage, I used the PD to select only those samples which are likely to default. For those cases, I used a regularized greedy forest to predict the logarithm of the loss. Different datasets were used in both stages of the process. These sets were quite large (about 600-1100 features) because I only used a few very basic methods of feature selection.

2 Features Selection / Extraction

The initial datasets ($train\_v2.csv$ and $test\_v2.csv$) consist each of 771 features. First of all, I removed all duplicated features (based on the first 25000 rows of the training set). This way 91 features could be removed. The set of the remaining 680 features is called $data_{all}$.

2.1 Golden Feature Set

Yasser Tabandeh posted a pair of features, $f527$ and $f528$, which can be used to achieve a very high classification accuracy ([1]). Furthermore $f527$ and $f528$ are very highly correlated. However, it is not needed to keep both features. Their difference $f527 - f528$ contains the same amount of information. I tried to find more features of this kind.
Therefore, I searched for all pairs of highly correlated features (correlation coefficient > 0.996) and kept there difference as a feature. This set is called $GoldenFeatureSet(GFS)$. It consists initially of 582 features, but lots of them were highly correlated. To reduce the size of this set I removed all highly correlated features (correlation coefficient > 0.99). After this cleaning step the final GFS consists of 226 features.

2.2 Aggregated Feature Set

I noticed the following relationship between the features $f67$, $f597$ and the golden features. Here is an example with the golden feature $diff\_f274\_f527$. It's defined as $f274$ - $f527$, which is a golden feature ($diff\_f274\_f527 \in GFS$):

id	*loss*	$f67$	$f597$	$diff\_f274\_f527$	$new\_feature$
52	0	11.9375	15	0.0599999999994907	0
53	0	11.9375	15	150.660000000033	0
54	0	11.9375	15	721.529999999999	0
55	0	11.9375	15	71.0100000000093	0
56	0	11.9375	15	10.75	0
57	0	11.9375	15	20.8800000000047	0
58	0	11.9375	15	646.14000000013	0
59	0	11.9375	15	5.36999999999534	0
60	0	11.9375	15	46.3700000001118	0
61	0	11.9375	15	96	0
62	0	11.9375	15	92.7400000002235	0
63	0	11.9375	15	428.060000000056	0
64	0	11.9375	15	1651.43999999994	0
65	11	11.9375	15	-737.639999999665	1
66	0	11.9375	15	-206.24	0
67	0	5.5019	6	-192.369999999995	0
68	0	5.5019	6	31.25	0
69	1	5.5019	6	-1027.1799999997	1
70	0	5.5019	6	158.32	0
71	0	5.5019	6	-372.639999999898	0
72	0	5.5019	6	0	0

The combination of $f67$ and $f597$ can be seen as some kind of a group identifier. For each of these groups there is mostly only one default.
In the upper example there are two groups $(11.9375, 15)$ and $(5.5019, 6)$. A default occurs often at the point where $diff\_f274\_f527$ marks the lowest value of the group. Therefore, I created a feature which marks this spot in each group.

Since $diff\_f274\_f527$ is already in GFS, I applied this process for all features in GFS. This creates 226 aggregated features (AFS).

3 Modeling Techniques and Training

3.1 Classification

3.1.1 First step

At first I tried to predict the probability of a default. Therefore the new target to predict is given by

$$target_{classification} = \begin{cases} 1 & \text{if } loss > 0 \\ 0 & \text{if } loss = 0 \end{cases} \tag{1}$$

For this task I used the following features: $f776$, $f777$, $f778$, $f2$, $f4$, $f5$, the first 200 features of $data_{all}$ (exluding those already named), the complete GFS and the complete AFS. This results in a set with 661 features.
The classification itself was done by gradient boosted trees. I used the gbm package in R with these settings: $distribution = bernoulli$, $n.trees = 250$, $shrinkage = 0.05$, $interaction.depth = 8$, $n.minobsinnode = 100$ and $bag.fraction = 0.8$.

To get the probabilities of default for the training set, I used a 12-fold-cross-validation. The whole classification process for the training and testing set was done 3 times and the results of all runs were averaged.

3.1.2 Second step

After the first classification step we got the PD for all training and testing samples. Lots of these probabilites are very low. In the second classification step I ignored all samples, which got a PD of less or equal than 0.00074. The remaining cases were classified again using all features of $data_{all}$, GFS and AFS. This results in a set of 1133 features. This second classification step was done by a regularized greedy forest with the following settings: $reg\_L2 = 0.5$, $algorithm = RGF$,$loss = LS$, $test\_interval = 25000$, $max\_leaf\_forest = 25000$, $reg\_sL2 = 0.005$.

The classification target for the RGF has to be in $\{-1, 1\}$. You have to make sure to rescale the predictions to [0,1]. I did this by:

$$forecast_{scaled} = \frac{forecast_{RGF} + 1}{2} \tag{2}$$

To get the probabilities of default for the training set, I used a 10-fold-cross-validation. The whole classification process for the training set was done 2 times and the results of both runs were averaged.
You get the final PD for all cases by merging the predictions of the RGF with those of the GBM: If the PD of the GBM is less or equal than 0.00074 keep those PDs; if not: take the PDs of the RGF.

3.2 Loss Prediction

I predicted the loss for all samples, which got a PD above about 0.51. I used loss = 0 for all other cases. The feature set for the loss prediction consisted of all features from $data_{all}$ and GFS. I tried to predict $log(1+loss)$ instead of $loss$.

The loss prediction was done by a regularized greedy forest with the following settings: $reg\_L2 = 0.5$, $algorithm = RGF$,$loss = LS$, $test\_interval = 20000$, $max\_leaf\_forest = 20000$, $reg\_sL2 = 0.005$. $NormalizeTarget$ was activated. The predictions of the RGF were rescaled by $exp(predictions) - 1$.

I did this process 3 times. On each run I changed the reg_L2 setting: $reg\_L2 = 0.25$, $reg\_L2 = 0.5$ and $reg\_L2 = 0.75$ were used. The final prediction was a simple average of all 3 predictions. Finally, I limited the prediction in the range $[0, 100]$ and rounded them.

4 Code Description

4.1 crossValidateParallel(input, target, train.model, predict.model, k, returnFits, packages, cores)

This function generates a k-fold cross-validation, which runs in parallel and returns a vector of all predictions for the out-of-sample folds in the same order and length as the target. It's a general framework, which uses the train.model-function to train a model on k-1 folds and predicts the out-of-sample fold using the predict.model-function. It can also return all k models (by setting returnFits to $TRUE$, but usually it's $FALSE$). *packages* refers to the name of any package that is needed to use the train.model or predict.model functions. *cores* is the number of threads that should run in parallel.

4.2 *createDatasets*

This script will creates all needed datasets. The following files should have been created if the script finished successfully: $final\_data\_full.RData$, $final\_golden\_features.RData$ and $min\_of\_golden\_features\_per\_group.RData$. This script may take about an hour to finish.

4.3 $final\_classification\_v1$

The first part of the classification process (see 3.1.1) is started with this script. It will create two files: The PD for all training cases is stored in $train\_classes\_gbm.csv$ and for all test cases in $test\_classes\_gbm.csv$. This script takes about a day on my workstation.

4.4 $final\_classification\_v2$

This script contains the code for the second step of the classification process (see 3.1.2). It's more or less the same as in the first classification script. Four files should be created: *train_classes_rgf.csv* and *test_classes_rgf.csv* contain the PD for some training and testing cases made by the RGF. *train_classes_full.csv* and *test_classes_full.csv* contain the final probabilities of default for all cases. This script can take days to finish.

4.5 *final_regression*

This script contains the code to create the loss predictions (see 3.2). It will create the final submission.

5 How To Generate the Solution

You will need at least a working versions of R (I used version 3.0.2) and RGF (see [2]). You also need lots of RAM (16 GB should be OK).
Here is a quick guide: Create a new folder *LoanDefault*. Create subfolders *scripts* and *data* in the *LoanDefault*-folder. Copy *train_v2.csv* and *test_v2.csv* in the *data*-subfolder. Install RGF in the *LoanDefault*-folder (it should look like '\LoanDefault \rgf1.2'). Copy the *crossValidateParallel* script to the subfolder *scripts*. Copy the files *train_classification.inp*, *train_regression_025.inp*, *train_regression_05.inp* and *train_regression_075.inp* into the folder '*LoanDefault* \rgf1.2 \test \sample'.

5.1 1.Step: Run script *createDatasets.R*

Open this R-file and adjust the path in the setwd()-command and optionally all paths to *train_v2.csv* and *test_v2.csv*. Please make sure that you have all needed packages installed.

5.2 2.Step: Run script *final_classification_v1.R*

Adjust the path in the setwd()-command and start the script.

5.3 3.Step: Run script *final_classification_v2.R*

Adjust the path in the setwd()-command and in the *model.fit* and *model.predict* functions.

5.4 4.Step: Run script *final_regression.R*

Adjust the path in the setwd()-command and in the *model.fit* and *model.predict* functions.

6 Licence

This content is released under the MIT License. Please see the file *LICENSE.txt*.

References

[1] `https://www.kaggle.com/c/loan-default-prediction/forums/t/7115/golden-features`.

[2] `http://stat.rutgers.edu/home/tzhang/software/rgf/`.