

Winning the KDD Cup Orange Challenge with Ensemble Selection

IBM Research

ANICULE@US.IBM.COM

Editor: Gideon Dror, Marc Boullé, Isabelle Guyon, Vincent Lemaire, David Vogel

Abstract

We describe our winning solution for the KDD Cup Orange Challenge.

Keywords: Ensemble Selection

1. Introduction and Task Description

The KDD Cup 2009 challenge was to predict, from customer data provided by the French Telecom company Orange, the propensity of customers to switch providers (churn), buy new products or services (appetency), or buy upgrades or add-ons (up-selling). The competition had two challenges: the Fast challenge and the Slow challenge. For the Fast challenge, after the targets on the training set were released, the participants had five days to submit predictions on the test set. For the Slow challenge, participants were given an additional month to submit their prediction.

The data set consisted of 100000 instances, split randomly into equally sized training and test sets. 15000 variables were made available for prediction, out of which 260 were categorical. Most of the categorical variables, and 333 of the continuous variables had missing value. To maintain the confidentiality of customers, all variables were scrambled. Also, there was no description of what each variable measured.

For the Slow challenge, a reduced version of the data set was also provided, with a subset of only 230 variables, 40 of which numerical. The organizers scrambled the small data set differently than the large one, and shuffled the rows and variables, in an attempt to encourage participants to treat it as a separate data set. Despite this, many participants, including ourselves, easily found the corresponding instances between the small and large data sets. In the end, however, uncovering these correspondences provided us little benefit, if any.

Submissions were scored based on the Area Under the ROC Curve (AUC) performance, with the average AUC across the three tasks being used to rank the participants. While multiple submissions per team were allowed, and feedback was provided for each of them in terms of performance on a fixed 10% of the test set, the competition rules stated that only the last submission from the team leader will count towards the final ranking. So the participants had to face the burden of model selection. The slow challenge presented one additional twist in terms of evaluation. Participants were allowed to make two sets of submissions, one on the large and one on the small data set, and the best of the two was considered toward the final ranking.

As a final note, we want to emphasize that the results we present in this paper reflect the particular choices we have made and directions we have explored, in some order, under the limited time of the competition, rather than a careful empirical study of the different methods we have used. So, while we will make a few comparative statements throughout the paper, we caution the reader against generalizing these results beyond the scope of this competition.

2. Our Story

Our overall strategy was to address this challenge using Ensemble Selection (Caruana and Niculescu-Mizil, 2004). In a nutshell Ensemble Selection is an overproduce-and-select ensemble building method that is designed to generate high performing ensembles from large, heterogeneous libraries of classifiers. Ensemble Selection has several desirable properties that made it a good fit for this challenge. First, it has been proven to be a robust ensemble building technique that yields excellent performance. Second, the generated ensembles can be optimized to any easily computable performance metric, including AUC. Third, it allows for loose coordination of the team members, as everyone can independently train classifiers using their preferred learning methods, and add those classifiers to the library. And fourth, it is an anytime method, in the sense that when the time to make predictions comes, an ensemble can be generated very fast using whatever classifiers made it into the library at that time.

Our results are summarised in Table 1. The first column indicates the classifier type (the best individual classifier we have trained, an ensemble generated by Ensemble Selection, or the submission of our competitors). The second column indicates what types of features were used (FS1 indicates that we used the set of features provided, after some standard preprocessing summarized in Section 2.1; FS2 indicates the use of additional features created to capture some non-linearity in the data, as described in Section 2.2.3; FS3 indicates the use of even more additional features described in Section 2.3.1). The next columns show the test set AUC on the three test problems, and the average AUC. Entries in the table are ordered by average AUC.

We will present our work in the next sections in close to chronological order to motivate our choices as we went along.

2.1 Preprocessing, Cleaning and Alike

Since the feature values were available prior to the target values, we spent our initial time on some fairly standard preprocessing that do not involve knowledge of the targets. The provided data set posed a number of challenges here. Aside from the sheer number of features, many had missing values and some of the categorical variables had a huge vocabulary (number of possible values).

Missing categorical values are typically less of a problem they can be considered just a separate value. Missing numeric values on the other hand are more concerning. We followed a standard approach of imputing the missing value by the mean of the features. We considered that the missingness itself might be predictive and added, for each of the 333 variables with missing values, an additional indicator variable indicating missingness. Another advantage of this approach is that some class of models (e.g. linear) can now estimate the

optimal constant to replace the missing value with, rather than relying on the calculated mean.

Most of the learning algorithms we were planning to apply do not handle categorical variable, so we needed to recode them. This was done in a standard way, by generating indicator variables for the different values a categorical attribute could take. The only slightly non-standard decision was to limit ourselves to encoding only the 10 most common values of each categorical attribute, rather than all the values, in order to avoid an explosion in the number of features from the variables with a huge vocabulary.

Finally, the features were normalized by dividing by their range, and the data was cleaned by eliminating all the features that were either constant on the training set, or were duplicates of other features.

To be able to evaluate the performance of the classifiers, and to build ensembles via Ensemble Selection, we adopted a 10-fold cross-validation approach. While we would have liked to perform all the 10 iterations of the 10-fold cross-validation, considering, in turn, each fold as a validation fold, this was unrealistic in the allotted time. Ultimately we only finished two iterations for the fast challenge, giving us a total of 10000 validation instances, and four iterations for the slow challenge, for a total of 20000 validation instances. To make predictions on the test set, given that we now have a version of a classifier for each fold (two for the fast challenge and four for the slow challenge), we average the predictions of the corresponding models. This has a bagging like effect that should lead to a performance boost.

In order to avoid overfitting the test set of the leader board, we decided early on not to rely on the feedback on the 10% for anything but sanity checks and final guidance in picking an ensemble building strategy.

2.2 The Fast Challenge

2.2.1 MANY DIFFERENT CLASSIFIERS

The first step in building an ensemble classifier via Ensemble Selection is to generate a library of base classifiers. To this end, we trained classifiers using using a range of learning methods, parameter setting and feature sets, as well as using post training calibration using Platt Scaling (Niculescu-Mizil and Caruana, 2005). We looked for learning algorithms that were efficient enough to be able to handle a data set of this size in the time allotted, while still producing high performing models. Guided in part by the results of (Caruana et al., 2008), we generated classifier libraries using random forests (Breiman, 2001) and boosted trees (Schapire, 2001) trained using the FEST package (Caruana et al., 2008), logistic regression trained using the BBR package (Genkin et al., 2007), SVMs trained using SVMPerf (Joachims, 2005), LibLinear (Fan et al., 2008) and LibSVM (Chang and Lin, 2001), decision trees, TANs and Naïve Bayes trained using Weka (Witten and Frank, 2005), Sparse Network of Winnows trained using the SNoW package (Carlson et al., 1999), and KNN, regularized least squares regression and co-clustering (Sindhwani et al., 2008) trained using in house code. We also trained some of these learning algorithms on several reduced feature sets obtained through PCA and through feature selection using filter methods based on Pearson correlation, and mutual information.

Table 1: Our journey.

Classifier Type	Feature Set	Challenge	Churn	Appetency	Up-Selling	Average
Slow Challenge Submission (ES)	FS3	Slow	0.7651	0.8816	0.9091	0.8519
Ensemble Selection	FS3	Slow	0.7629	0.8805	0.9091	0.8509
Fast Challenge Submission (ES)	FS2	Fast	0.7611	0.8830	0.9038	0.8493
Ensemble Selection	FS2	Fast	0.7611	0.8793	0.9047	0.8484
Best Competitor Slow challenge		Slow	0.7570	0.8836	0.9048	0.8484
Ensemble Selection	FS1	Fast	0.7563	0.8771	0.9038	0.8457
Best Competitor Fast Challenge		Fast	0.7565	0.8724	0.9056	0.8448
Best Single Classifier	FS3	Slow	0.7511	0.8794	0.9025	0.8443
Best Single Classifier	FS2	Fast	0.7475	0.8779	0.9000	0.8418
Best Single Classifier	FS1	Fast	0.7354	0.8779	0.9000	0.8378

Little or no attempt is made to optimize the performance of the individual models; all models, no matter what their performance, are added to the model library for the problem. The expectation is that some of the models will yield good performance on the problem, either in isolation or in combination with other models. In total, the classifier libraries were composed of 700-1200 individual models.

Judging from the two folds of internal cross-validation we were able to perform up to this point, the best individual classifiers on churn were boosted trees followed by regularized logistic regression, and random forests. On appetency, the best single method was random forests, followed by boosted trees and logistic regression, while on up-selling, boosted trees were best, followed by random forests and logistic regression. At this point, using the best single classifiers, as deemed by the internal cross-validation, yielded a test set AUC of 0.7354 for churn, 0.8779 for appetency, and 0.9000 on up-selling, for an average AUC of 0.8378. This was lower than the performance obtained by a many of the competing teams.

Interestingly, on all three problems, boosting clearly overfit early in the training, yielding peak performance after only about 10 rounds of boosting and decreasing significantly after that. We actually came very close to missing this peak performance, as in the beginning we focused on larger number of rounds. Also boosting shallow trees (two levels deep for churn and three levels deep for appetency and up-selling) performed better than boosting deeper trees.

Other interesting observations were that L_1 regularization was better than L_2 regularization for logistic regression when using the entire feature set. Feature selection via a Pearson correlation based filter method improved the performance of logistic regression models, with L_1 and L_2 regularization performing similarly. For SVMs, directly optimizing AUC in SVMPerf yielded better results than optimizing the hinge loss, but the performance was still lower than logistic regression. Initial attempts at using non-linear kernels were not at all promising, and computationally expensive, so they were abandoned.

2.2.2 ENSEMBLE SELECTION

Once a classifier library is generated, Ensemble Selection builds an ensemble by selecting from the library the subset of classifiers that yield the best performance on the target

optimization metric (AUC in our case). Models are selected for inclusion in the ensemble using greedy forward stepwise classifier selection. The performance of adding a potential model to the ensemble is estimated using a hillclimbing set containing data not used to train the classifiers. At each step ensemble selection adds to the ensemble the classifier in the library that maximizes the performance of the ensemble to this held-aside hillclimbing data. Classifiers can be added to the ensemble multiple times, allowing for a crude weighting of the different classifiers in the ensemble.

When there are a large number of models to select from, the chances of overfitting increase dramatically. Caruana and Niculescu-Mizil (2004) describe two methods to combat overfitting. The first is to initialize the ensemble with a set of N classifiers that have the best uni-model performance on the hillclimb set. The second performs classifier bagging—multiple ensembles are built from random subsets of the classifiers, and then averaged together. The aim of the classifier bagging is to increase performance by reducing the variance of the forward stepwise selection process.

We built an AUC optimized ensemble classifier for each of the three problems using Ensemble Selection. Following (Caruana et al., 2006), we combined the two validation folds from our internal cross-validation and used them as the Ensemble Selection hillclimbing set. Both overfitting prevention techniques described above were used. The test set performance of the ensemble models was 0.7563 on churn, 0.8771 on appetency and 0.9038 on up-selling, for an average AUC of 0.8457. This performance was already better than that of the other competitors on the Fast challenge. It is notable that this performance was obtained through fairly standard techniques without much need for human expertise or intervention, or tailoring to the particular problems addressed in this competition. One can easily imagine all these techniques being incorporated in a general purpose push-button application.

At the time of the competition, however, we did not know if we had the best performance or not. In fact, on the 10% of the test set that was used for feedback our performance was below that of other participants.

2.2.3 MORE FEATURES

So we had one more day to push forward. This was when we realised that there are notable discrepancies between measuring the quality of individual variables via mutual information with the targets, and via rank correlation with the targets. Some of the features with the highest mutual information (calculated by binning numeric variables into 10 bins) had quite a poor rank correlation. The most likely explanation is some form of non-monotonic dependence. So, while these features were very predictive, linear models could not take advantage of them. In this case, constructing new features can simply serve as a change of representation to allow expressing non-linear relationships in a linear model. To this extent we explored two approaches:

- **Binning:** As explained earlier, we observe higher predictive performances in terms of mutual information when binning was used. So the obvious solution was to include for each such feature 10 additional binary features corresponding 10 bins. However, it is unlikely that the equal size binning is optimal.
- **Decision Tree:** The second approach was motivated by using a decision tree to identify the optimal splitting points. We recode each feature by training a decision

tree of limited depth (2,3 or 4) using that feature alone, and let the tree directly predict the target. The probabilistic predictions of this decision tree were used as an additional feature, that now was linearly (or at least monotonically) correlated with the target.

The addition of these new features had a significant impact on the performance of linear models, with L_1 regularized logistic regression becoming the best model on churn and improving the test set performance of the best base level churn model by 0.0121 to 0.7475. It also had a positive impact on the performance of the ensembles build by Ensemble Selection for all three problems, resulting in a test set performance of 0.7611 on churn, 0.8793 on appetency, and 0.9047 on up-selling.

2.2.4 SUBMISSION FOR THE FAST CHALLENGE

Before the final submission for the Fast challenge, we analysed in more detail the ensembles built by Ensemble Selection. We realized that on appetency, after the initialization phase (where models with high uni-model performance were added to the ensemble), the first model Ensemble Selection was adding was some poor performing decision tree. We were worried that this indicates that Ensemble Selection was actually overfitting the hillclimb set. So, for appetency, we decided to use the ensemble model generated right after the initialization phase, containing only the six best models (as measured on the hillclimb set), and not continue with the forward stepwise classifier selection. The results on the 10% of the test set were also in accord with this hypothesis. In hindsight, it turned out to be the right decision, as it significantly improved our performance on the test set.¹

We have also investigated whether classifier bagging was necessary, by also running Ensemble Selection with this option turned off. We noted that on the 10% of the test set we received feedback on, classifier bagging provided no benefit on churn and up-selling, and was only increasing performance on appetency (which was consistent with our hypothesis that Ensemble Selection overfit on this problem). Being also guided by the results in (Caruana et al., 2006), which stated that, once the hillclimbing set is large enough, classifier bagging is unnecessary, we decided to use ensembles built without classifier bagging as our final submissions for churn and up-selling. In hindsight, this was not a good decision, as the test set performance on up-selling was slightly worse than if we were to use classifier bagging.

2.3 Slow Challenge

For the slow challenge, we first increased the hillclimbing/internal validation set by training on two extra folds, bringing us to four folds for a combined hillclimbing/validation set of 20000 instances.

Encouraged by the improvements we obtained in the last day of the Fast challenge, the main trust of our efforts was towards creating better and new features. The addition of these features, described below, in combination with the move from two to four folds, yielded an increase in test set performance for the best individual models to 0.7511 on churn, 0.8794 on appetency and 0.9025 on up-selling (0.8443 average across the three problems). The

1. This was not the case for the other two problems, churn and up-selling, where the forward stepwise classifier selection improved the performance significantly.

performance of the Ensemble Selection built ensembles rose to 0.7629 for churn, 0.8805 for appetency, and 0.9091 for up-selling (0.8509 average).

2.3.1 EVEN MORE FEATURES

Explicit Feature Construction: For a number of features with typical characteristics, we were able to isolate the signal directly: The positive rate of churn for all rows with 0 value was up to twice the positive rate for all other numeric values. This happened for a number of numeric features that overall seemed to be close to normally distributed, but shows under close inspection certain regularities, such as frequency spikes for certain values. The effect is not overly strong; typically only a few thousand examples have a zero value and a zero indicator for one such numeric feature only leads to an AUC of 0.515. We simply added one new feature that counts the number of times an example had a zero within one of these numeric features which seem to have an AUC of 0.62 on cross validation.

Features From Tree Induction We extended the decision tree based recoding approach to pairs of attributes in order to get two way non-additive interactions between pairs of variables. To this end, for each pair of attributes, we trained a decision tree of limited depth (3,4) to predict the target from only the two attributes. We then used the predictions of the tree as a candidate extra feature in our models.

Co-clustering We have also tried a new feature generation approach. When looking at missing values, we noticed that they were missing for groups of features at once. That is, for every instance, the values for all the features in the group were either all missing or all present. Inspired by this observation, we extend the idea further to other categorical/numerical features. For example, suppose that features f_1, f_2 , and f_3 take value of a, b , and c respectively across instances i_1, i_2, i_3 , and i_4 . We can then generate one binary feature, that takes 1 on i_1, i_2, i_3 , and i_4 and 0 on all other instances. The problem of identifying subsets of features/instances with this property, is known as the constant bi-clusters problem in the bio-informatics domain. We ran a fast probabilistic bi-clustering algorithm (Xiao et al., 2008) to identify promising bi-clusters which we then used to generate new indicator features.

2.3.2 SMALL DATA SET

We quickly trained all our models on the small data set, but the internal cross-validation results suggested that the performance obtained from the small data set was significantly worse than what we obtained from the large one. So we decided not to pursue the small data set any more, and focused our attention on the large data set. Nevertheless, we did unscramble the small data set for two main reasons: to get feedback on about 20% of the test data instead of only 10%, and to be able to make two distinct submissions using models trained on the better performing large data set (per competition rules, the best of the two submissions would count towards the slow challenge ranking). In the end, however, it turned out that unscrambling the small set did not give us any significant advantage as the feedback on the 20% of the data was barely used, and the two submissions we ended up making were very similar to each other.

2.3.3 SUBMISSION FOR SLOW CHALLENGE

Finally, we again analyzed the ensembles produced by Ensemble Selection in more detail, and noticed some strange behaviour with the ensemble initialization phase. Because the model libraries contained a large number of high performing, but very similar logistic regression classifiers, in the initialization phase, Ensemble Selection was adding all these classifiers to the ensemble, essentially overemphasizing the logistic regression models. Since we also had a larger hillclimb set, overfitting was less of a concern, so we decided to turn off the ensemble initialization. With the initialization turned off, we gained, on average, another 0.001 in test set AUC, for a final performance of 0.7651 on churn, 0.8816 on appetency, and 0.9091 on up-selling (0.8519 average AUC).

References

- L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen, and Dan Roth. Snow user guide. Technical report, 1999.
- Rich Caruana and Alexandru Niculescu-Mizil. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, 2004.
- Rich Caruana, Art Munson, and Alexandru Niculescu-Mizil. Getting the most out of ensemble selection. In *Proceedings of the 6th International Conference on Data Mining (ICDM '06)*, 2006.
- Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, 2008.
- Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
- Alexander Genkin, David D. Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49:291–304(14), August 2007.
- Thorsten Joachims. A support vector method for multivariate performance measures. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 2005.
- Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, 2005.
- R. Schapire. The boosting approach to machine learning: An overview. In *In MSRI Workshop on Nonlinear Estimation and Classification*, 2001.
- Vikas Sindhwani, Jianying Hu, and Aleksandra Mojsilovic. Regularized co-clustering with dual supervision. In *NIPS*, 2008.
- Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.
- Jing Xiao, Lusheng Wang, Xiaowen Liu, and Tao Jiang. An efficient voting algorithm for finding additive biclusters with random background. *Journal of Computational Biology*, 15(10), 2008.