JMLR: Workshop and Conference Proceedings 7: 65-75 KDD cup 2009

KDD Cup 2009 @ Budapest: feature partitioning and boosting

Miklós Kurucz Dávid Siklósi István Bíró Péter Csizsek
Zsolt Fekete Róbert Iwatt Tamás Kiss Adrienn Szabó
{MKURUCZ, SDAVID, IBIRO, CSIZSEK, ZSFEKETE, RIWATT, KISSTOM, ASZABO}@ILAB.SZTAKI.HU
Computer and Automation Research Institute of the Hungarian Academy of Sciences

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

Abstract

We describe the method used in our final submission to KDD Cup 2009 as well as a selection of promising directions that are generally believed to work well but did not justify our expectations. Our final method consists of a combination of a LogitBoost and an ADTree classifier with a feature selection method that, as shaped by the experiments we have conducted, have turned out to be very different from those described in some well-cited surveys. Some methods that failed include distance, information and dependence measures for feature selection as well as combination of classifiers over a partitioned feature set. As another main lesson learned, alternating decision trees and LogitBoost outperformed most classifiers for most feature subsets of the KDD Cup 2009 data.

Keywords: Feature selection, classifier combination, LogitBoost, Alternating Decision Trees.

1. Introduction

The KDD Cup 2009 task targeted for the propensity of customers to switch provider (churn), buy new products or services (appetency), or buy upgrades or add-ons proposed to them to make the sale more profitable (up-selling). For 50,000 anonymous customers a small data set of 230 and a large of 15,000 features was provided; in this paper we describe our various successful and failed attempts mostly over the large data set.

Telephone customer behavior analysis appears less frequently in publications of the data mining community. Some exceptions include machine learning methods for churn prediction on real data (evolutionary algorithm, Au et al. (2003); classifier combination by linear regression, Wei and Chiu (2002); Naive Bayes, Nath and Behara (2003); graph stacking, Csalogány et al. (2007)). The area is explored in more depth in marketing research including small sample survey results of Kim and Yoon (2004) and rule extraction and behavior understanding over a small 21-feature data by Ultsch (2002). Of closest interest, Neslin et al. (2006) present the overview of a churn classification tournament very similar to the KDD Cup 2009 task but with emphasis also on managerial meaningfulness and model staying power.

The difference in the KDD Cup 2009 large data set compared to typical classification problems is the abundance of features. For this end we had prepared feature selection and partitioning methods before the training label release. By the lessons we have learned from

Web Spam Challenges (Siklósi et al. (2008)), we had expected that feature partitioning and classifier combination would perform better than global classifiers.

Due to the large number of features it was also clear that feature selection methods are required. Our best performing methods have turned out to be very different from those described in some well-cited surveys as e.g. by Dash and Liu (1997): feature evaluation could only be used for a weak pre-selection while wrapper methods failed due to slow convergence and overfitting. Our final feature selection method is the union of the best features selected by LogitBoost of Friedman et al. (2000) over the feature partition that we have originally devised for classifier combination.

Our classifier implementation choice was mostly determined by the possibilities of the machine learning toolkit Weka of Witten and Frank (2005) that has wide variety in logistic regression, decision tree, neural nets mostly considered applicable for churn prediction described e.g. in Neslin et al. (2006). In addition we tested the SVM implementation of Chang and Lin (2001) considered most powerful for several classification tasks as well as Latent Dirichlet Allocation, a dimensionality reduction and generative modeling approach by Blei et al. (2003) that is believed to work well for skewed features. In our experiments the ultimate method turned out to be the combination of LogitBoost Friedman et al. (2000) and ADTrees of Freund and Mason (1999).

Next we describe our method and experimental results in detail including some partial results that appeared promising but did not perform as expected. In Section 2.1 we describe the way we partitioned features by their global properties. Then we describe successful and unsuccessful attempts first for feature selection (Sections 2.2–2.3), then for classifier ensemble construction along with the detailed AUC evaluation in Sections 2.4–2.5.

2. Experiments and Results

We describe our experiments, but successful and unsuccessful, over the large data set of KDD Cup 2009. The sole exception of a small data set experiment is one described in Section 2.2. As the key step of our final result, we have managed to select a powerful small feature subset by a LogitBoost based method described in Section 2.3.

An important ingredient of our method consists of an expert partitioning of the feature set (Section 2.1). While in our initial plans described in Section 2.5, the purpose of this partitioning was to train separate models and combine them, this approach was outperformed by our final submissions in Section 2.4. The partitioning however proved useful for our feature selection procedure in Section 2.3.

For classification we applied Weka implementations with intensive parameter search. In order to avoid overfitting for the online feedback from the 10% test set, we typically we tested performance over a random 10% **heldout set** set aside for method combination and another 10% **validation set** internal testing.

The heldout and validation sets were fixed once for the entire experiment. These test typically performed 2-3% better than the on-line feedback but more or less kept the relative order of the predictors. Note that our final submission consisted of two classifiers combined by taking the average score. In this case the final training was performed over the entire training set, including both the heldout and the validation set.

In our experiments we managed to avoid overfitting for the feedback on the 10% test set: except for a single task (appetency with the difference in the number of LogitBoost iterations) among all of our submissions, the relative order over the 10% and 100% test set was identical. Up to now however we are not able to explain why the relative order compared to other teams have changed; note that over 10% of the large test set our submission performed best with an AUC score 0.8457 while our final result is behind the winner by 0.0065.

We note that we did not apply feature preprocessing and did not try to capture the interaction effects between variables. While some internal tests involved cross-validation, due to time constraints we used our predefined heldout and validation sets since we observed no inconsistencies in their predicted performance.

We ran our tests on standalone multicore machines with more than 32GB RAM and a condor driven cluster of some older dual-core machines. We run in parallel different algorithms on different machines.

In what follows, in Section 2.1 we describe the expert feature partitioning. In Section 2.2 we briefly describe unsuccessful attempts for feature selection based on several methods generally considered effective in the literature as e.g. in the survey of Dash and Liu (1997). Our successful feature selection procedure is based on the expert partitioning and LogitBoost and is described in Section 2.3. The final submissions based on the combination of LogitBoost and ADTrees is found in Section 2.4. Finally in Section 2.5, we describe some unsuccessful attempts and in particular the most promising one consisting of a large classifier ensemble based on our expert feature partition. For reference, the parameters used in our procedures are summarized in Table 3 at the end of the paper.

2.1 Feature exploration and partitioning

As first step before test set release, we explored feature properties and partitioned the data. The partition is based on observable properties of the distributions that we have identified in the attempt of understanding the actual source of the features. Some sample histograms for a few less obvious classes along with the selection steps are found in Fig. 1. Our motivation of partitioning stems from our Web spam classifier of Siklósi et al. (2008) where for example content and link based features are best classified separately with the results combined by e.g. random forest. Although classification along the same lines as it will be described in Section 2.5 is outperformed by other methods, we used this partitioning for feature selection as described in Section 2.3.

The rules for defining the feature partition as also seen in Fig. 1 are the following.

Bad: the most frequent or missing value has frequency at least 49500 (10500).

Nomin: nominal with at least 500 non-missing values that is not Bad (269).

BinNum: numeric binary feature that is not Bad (1190).

Missing: numeric with missing values that is not BinNum or Bad (330).

Neg100: numeric with at least 100 negative values that is not Missing or Bad (85).

Cont10000: numeric with continuous range (at least 10,000 distinct values) that is not Missing or Neg100 (503).

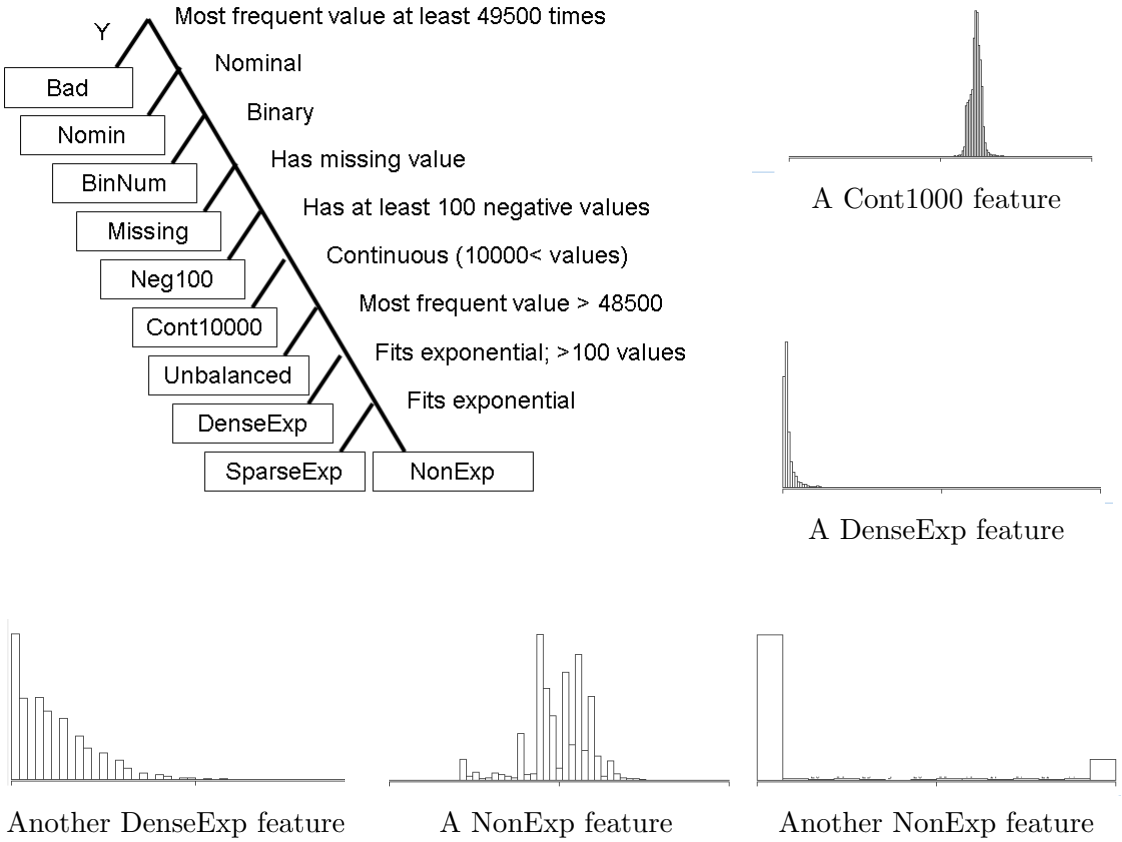

Figure 1: The feature partitioning tree obtained by investigating elementary properties, with some sample histograms. Over the tree the "yes" branch is always on the left side.

Unbalanced: numeric with the most frequent value appearing at least 48500 times that is neither Bad, Missing nor Neg100 (540).

DenseExp: numeric with good fit to the exponential distribution that has more than 100 distinct values and neither Bad, Neg100, Cont10000, Missing or Unbalanced (530).

SparseExp: numeric with good fit to the exponential distribution that has at most 100 distinct values and neither Bad, Neg100, Cont10000, Missing or Unbalanced (445).

NonExp: numeric that is not Bad, Neg100, Cont10000, Missing, Unbalanced, DenseExp and SparseExp (587).

2.2 Feature selection

Our first feature selection attempt relied on well known feature evaluation methods such as Information Gain, Gain Ratio, and Chi Squared Probe. These methods turned out to be useful only as a pre-selection of still a relative large number of features and we have

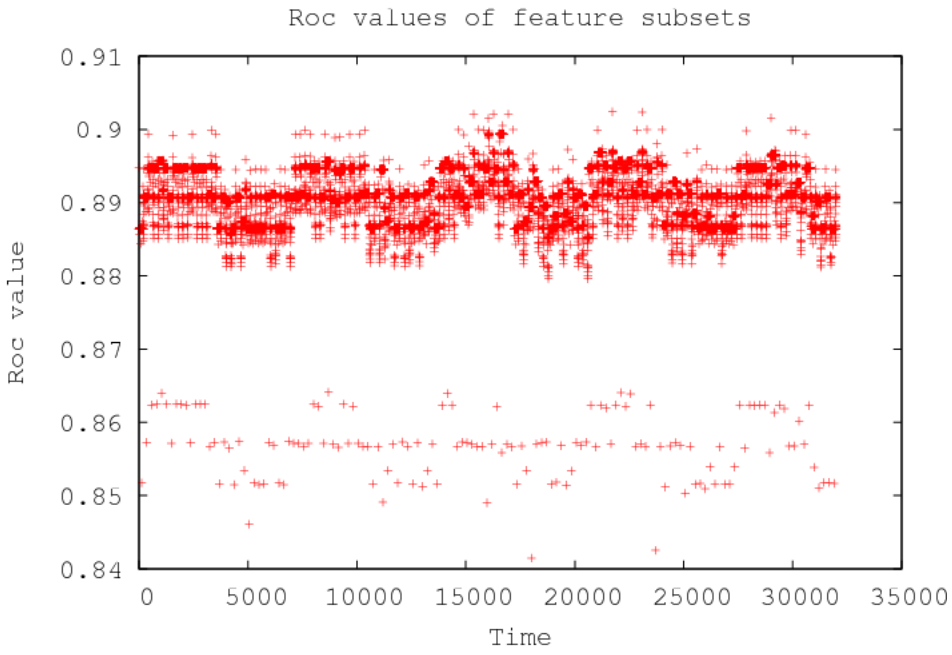

Figure 2: A sample 35,000 feature subset performance selected by our random walk wrapper method biased towards better AUC over the small data set.

completely dropped this approach from consideration. Information Gain and Chi Squared Probe tends to overscore features with many unique values while Gain Ratio that normalizes scores proportional to the number of unique values tends to overscore features with few unique values. We observed the following problems with this selection:

- Non-predictive features were selected in a high number.
- The threshold to drop features was generally hard to decide and justify.
- These methods tended to select highly correlated features.

Our second attempt was the classifier-driven approach. One possibility is to start a random walk in the feature space, starting from a preselected feature set, and at each step choose a feature to add or drop. The evaluation of every feature set can be made by a given classifier. Although this method ran efficiently in our parallel environment, it still took a long time to find a generally good feature set. In addition in our experiments the method was prone to overfitting: the difference between the exceptionally best and overall good feature subsets diminished when we switched between our heldout and validation sets. A sample run over 170 features of the small data set selected by feature evaluation is shown in Fig. 2.

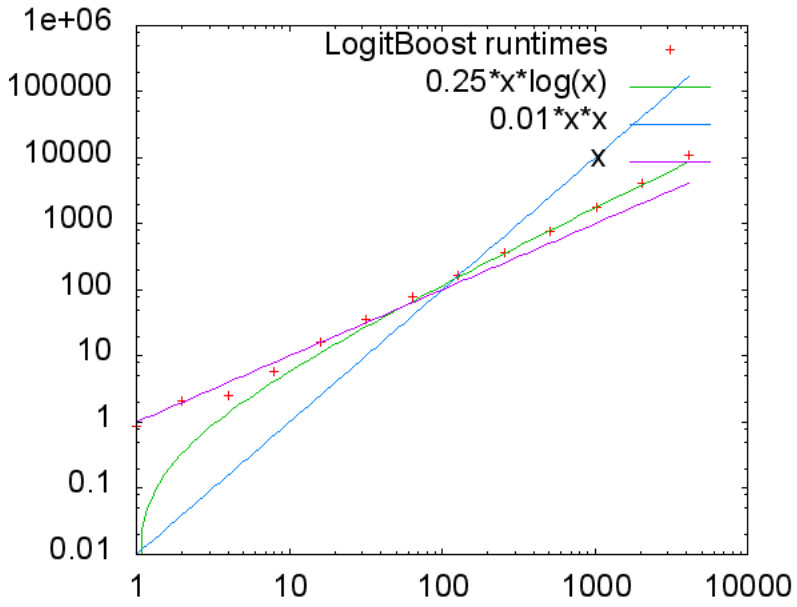

Figure 3: The measured running time of the LogitBoost implementation of Weka with decision stump as base classifier, 60 iterations, as the function of the number of features.

2.3 The final feature selection procedures

For our final submissions, LogitBoost as feature selection proved to be the most effective method. For each partition of features from Section 2.1, after preselection by feature evaluation we run LogitBoost of Friedman et al. (2000) with Decision Stump base classifier. This composite classifier chooses only a few (in our case 10–60) features to build a model. We run LogitBoost for all partitions of Section 2.1 with the following parameters:

- We used 60 iterations of boosting;
- We used bagging over 90% of the data with 10 iterations;
- We selected the best 10 features; the actual figure varies as LogitBoost tends to select the best features more than once but bagging yields multiple sets;
- For the largest partition BinNum we selected 40 features instead of 10.

We created our feature set from the union of those selected over the partition. After this procedure we were left with 75 features for churn, 90 for appetency and 65 for upselling. These figures reflect the hardness of classifying appetency: in this case there were no real best features and LogitBoost selected the same feature multiple times less often than for the other two tasks. We remark that these final features still contained 3-4 features with both zero IG and χ^2 and these could have been removed with no performance difference.

	churn	appetency	upselling	score
Slow Track Winner (U Melbourne)	0.7570	0.8836	0.9048	0.8484
LogitBoost + ADTree by partition (final)	0.7567	0.8736	0.9065	0.8456
Fast Track Winner (IBM Research)	0.7611	0.8830	0.9038	0.8493
LogitBoost by partition (final fast)	0.7496	0.8683	0.9042	0.8407
Combination LogitBoost	0.7409	0.8561	0.8894	0.8288

Table 1: The AUC value of selected final methods over the test set. LogitBoost with and without ADTree is trained over the entire training label set and by using the features selected within each of the partitions of Fig. 1. The last entry consists of the combination of individual classifiers over this same partition as described in Section 2.5.

One rationale for running feature selection separately for each partition is that the running time grew superlinearly with the number of features, as seen in Fig. 3. A linear fit to the log-log plot of the Figure produces a slope of 1.2911. Even without point before the leftmost, which is probably an outlier, the slope is 1.1338.

Another reason for the expert partitioning of the features is that we could even achieve marginal improvement (see Table 2 in Section 2.4) over random partitioning. For comparison we also tested a random partition of about 250-300 features in each set that could fit into single lower capacity machines of our cluster. Since the number of sets in the partition was large, we had to iterate selection until only a smaller number of approximately 200 features remained. Note that this method was also much slower since we could not count on an appropriate grouping of the features and we had to keep larger candidate subsets in intermediate steps. For this reason we did not even compute results for all tasks with this method.

We found that the two methods find almost the same features. We observed that the features with great imbalance are generally non-predictive: they were only exceptionally selected into the final feature sets.

We also identified some problems with our LogitBoost-based feature selection method:

- Some discarded features could have been useful for other classifiers—this is a common problem for classifier driven feature selection.
- Non-predictive features can still be selected, although unlikely.
- Uneven distribution of predictive features in partitions may cause dropping some of them, although unlikely.

2.4 The final classifier ensemble

After the feature selection procedure of Section 2.3, over less than a hundred features, we used LogitBoost of Freund and Mason (1999) with decision stump as base classifier for the fast track and additionally ADTree classifiers of Friedman et al. (2000) for the slow track. Both classifiers were used with 10 iterations of bagging over 90% of the data points. Cost

	churn		appetency		upselling	
	heldout	valid	heldout	valid	heldout	valid
Ensemble combin. by LogitBoost		0.7667		0.8537		0.9100
LogitBoost by expert selection	0.7557	0.7649	0.8668	0.8509	0.9122	0.9099
LogitBoost random selection	0.7540	0.7612			0.9064	0.9069
Ensemble log-odds by LogitBoost		0.7583		0.8361		0.9026
Linear SVM	0.6764	0.7026	0.8028	0.7987	0.8845	0.8760
Ensemble combin. by BayesNet		0.7012		0.7905		0.8057
LDA with BayesNet	0.6008	0.6246	0.5995	0.6278	0.7598	0.7539
Churn predictor			0.5995			
Missing w/ LogitBoost	0.7232	0.7318	0.8394	0.8217	0.8855	0.8931
NonExp w/ AdaBoost	0.7188	0.7359	0.8551	0.8332	0.8835	0.8815
Nominal w/ LogitBoost	0.6657	0.6696	0.8385	0.7868	0.7623	0.7649
Cont10000 w/ LogitBoost	0.6465	0.6631	0.6564	0.6712	0.7419	0.7474
BinNum	0.6369	0.6187	0.7204	0.7233	0.8016	0.8126
DenseExp w/ LogitBoost	0.6294	0.6473	0.6398	0.6591	0.7251	0.7391
NonExp w/ Bayes	0.6230	0.6531	0.5870	0.6393	0.7330	0.7224

Table 2: The AUC value of different classification methods over the whole set (top) and certain feature subsets (bottom) for the three subtasks.

matrices improved performance for appetency only where the optimal false negative to false positive cost ratio was 30. The actual values were tuned over our predefined 10% heldout set. For a summary of parameters, see also Table 3.

We summarize our results for the KDD Cup large test set, both 10% and the whole, in Table 1. Classifiers over the best features selected over our expert partition performed best. The combination of LogitBoost and ADTree improved performance. Best combination turned out to be the plain average that constitutes our final submission of AUC 0.8456 ranking sixth in the competition.

2.5 Combination over the feature partition

In this section we describe our unsuccessful attempt of defining a classifier ensemble over our expert partition as well as the behavior of the individual feature subsets. We summarize the results for our 10% heldout and validation sets in Table 2. Note that over these sets the final ensemble method (top row of Table 2) outperformed LogitBoost on this set, but LogitBoost trained over the entire set performed better for the Cup test set as seen in Table 1.

For all feature subsets we have tested a variety of Weka classifiers, and libsvm with various kernels. The top part of Table 2 shows performance by using all features while the bottom part by using given subsets only. No classifier has managed to outperform LogitBoost for any of the feature subsets nor did they yield improvement in any combination.

We describe the best performing ensemble combination method in Algorithm 1. The optional line marked with $*$ corresponds to the log-odds based combination proposed by

noend 1 Classifier ensemble method.

for all feature subsets **do**
  train models on training - heldout - validation (80%)
  tune parameters on the heldout set
  compute log-odds for the entire training set*
end for
combine over the validation set

Lynam et al. (2006). The results correspond to the first and fourth rows of Table 2 that show a surprisingly deteriorated performance of log-odds. In these cases the heldout set was used for combination and hence results for the validation set are given. Ensemble combination with BayesNet performed much worse.

In comparison to the ensemble methods, the second and third rows of Table 2 show a single LogitBoost classifier applied to the features of the two different successful feature selection methods (Section 2.3). Note that in the final submission, we trained LogitBoost over the entire test set that changed the relative order of the two methods for the final submissions in Table 1.

Among the individual feature subsets surprisingly those with missing values performed best. In our guesses these may include responses to questionnaire or call center operators with predictive value stronger than generated features based on service usage. These were followed by the "regular" numeric features with non-exponential distribution, the only exception in that here AdaBoost performed better than LogitBoost. Other classifiers performed much worse for all subsets.

Global classifiers such as BayesNet of linear SVM performed poor. Latent Dirichlet Allocation for dimensionality reduction as in Bíró et al. (2009) performed surprisingly weak as well.

The only successful feature evaluation method, in the sense of Section 2.2, turned out AUC itself. In this run we passed 95 features with best individual AUC to LogitBoost.

As a final unsuccessful trial we experimented with using training data from one task to improve prediction for the other. A simplest example is the application of the churn predictor for appetency in Table 1. The rationale is that a user who churns will not buy upgrades and vice verse. We have also observed that positive labels were disjoint for the three tasks. We tested two methods but could not improve our results:

Classifier combination: We used the results of several final and partial classifiers across tasks in combination. Note that the AUC of one classifier for another task never reached even close to 0.6 and hence failure is no surprise.

Case weighting: If we classify appetency, those who churn are "more negative" than those who just do not buy new services. For a decision stump classifier we may introduce three classes and use a cost matrix with penalties higher for classifying churned customers positive for appetency than non-churned negatives. In this way decision stump acts as regression for the outcome 1 for appetency, 0 for no appetency, and a larger negative value for churn.

parameter	value
iterations, LogitBoost	60
bagging data fraction	90%
bagging iterations	10
final feature set size, churn	75
final feature set size, appetency	90
final feature set size, upselling	65
FN/FP cost ratio, appetency	30

Table 3: The AUC value of different classification methods over the whole set (top) and certain feature subsets (bottom) for the three subtasks.

Conclusion

In our experiments we observe a clear gain of two classifiers, LogitBoost with decision stump and ADTrees. LogitBoost also performed well for feature selection. We used a feature partitioning method based on statistical properties. The combination of the classifiers over this partition performed close to best (in some cases even better on our heldout sets) and thus we believe that a partition relying also on the meaning of the features (e.g. traffic, sociodemographic or neighborhood based) may outperform the blind anonymous classifiers of the Cup participants. We also expect the call graph extracted from the call detail record can boost performance via the graph stacking framework as e.g. in Csalogány et al. (2007).

Acknowledgments

This work was supported by the EU FP7 project LiWA—Living Web Archives and by grant OTKA NK 72845.

References

Wai-Ho Au, Keith C. C. Chan, and Xin Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Trans. Evolutionary Computation*, 7(6): 532–545, 2003.

István Bíró, Dávid Siklósi, Jácint Szabó, and András A. Benczúr. Linked latent dirichlet allocation in web spam filtering. In *AIRWeb '09: Proceedings of the 5th international workshop on Adversarial information retrieval on the web*. ACM Press, 2009.

D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(5):993–1022, 2003.

C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines, 2001.

Károly Csalogány, A.A. Benczúr, D. Siklósi, and L. Lukács. Semi-Supervised Learning: A Comparative Study for Web Spam and Telephone User Churn. In *Graph Labeling Workshop in conjunction with ECML/PKDD 2007*, 2007.

M. Dash and H. Liu. Feature selection for classification. *Intelligent data analysis*, 1(3): 131–156, 1997.

Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *In Machine Learning: Proceedings of the Sixteenth International Conference*, 1999.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of statistics*, pages 337–374, 2000.

H.S. Kim and C.H. Yoon. Determinants of subscriber churn and customer loyalty in the Korean mobile telephony market. *Telecommunications Policy*, 28(9-10):751–765, 2004.

T.R. Lynam, G.V. Cormack, and D.R. Cheriton. On-line spam filter fusion. *Proc. of the 29th international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130, 2006.

S.V. Nath and R.S. Behara. Customer churn analysis in the wireless industry: A data mining approach. *Proceedings of the 34th Annual Meeting of the Decision Sciences Institute*, 2003.

S.A. Neslin, S. Gupta, W. Kamakura, J. Lu, and C.H. Mason. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of Marketing Research*, 43(2):204–211, 2006.

Dávid Siklósi, András A.Benczúr, István Bíró, Zsolt Fekete, Miklós Kurucz, Attila Pereszlényi, Simon Rácz, Adrienn Szabó, and Jácint Szabó. Web Spam Hunting @ Budapest. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

A. Ultsch. Emergent self-organising feature maps used for prediction and prevention of churn in mobile phone markets. *Journal of Targeting, Measurement and Analysis for Marketing*, 10(4):314–324, 2002.

Chih-Ping Wei and I-Tang Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert Syst. Appl.*, 23(2):103–112, 2002.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005. ISBN 0120884070.