Ensembling XGBoost and Neural Network for Churn Prediction with Relabeling and Data Augmentation

PKU Fresher at WSDM CUP 2018 Churn Prediction

Chence Shi, Zheye Deng, Yewen Xu, Weiping Song, Yichun Yin, Jile Zhu, Ming Zhang*
School of EECS, Peking University
{chenceshi,dzy97,xuyewen,songweiping,yichunyin,zhujile0918,mzhang_cs}@pku.edu.cn

ABSTRACT

This paper describes our solution in KKBOX's Churn Prediction Challenge, one of tasks in WSDM Cup 2018. The competition aims at predicting whether the KKBOX's users will churn after a period of time. To build a competitive system, we first enrich training set by data augmentation and relabeling, and then carefully design specific features for this problem. By ensembling the models of neural networks and XGBoost, our team *PKU Fresher* ranks 6th among 575 teams on the final private board[1].

1 INTRODUCTION

The goal of this competition is to predict whether a user will churn after his subscription expires. The dataset used is from KKBOX, which is one of Asia's leading music streaming services with millions of users, and it includes transactions of users, daily user logs describing listening behaviors and user profile information. A user may order or cancel service subscription at any time, which will update the expiration date. One user is considered as a churn one, if he or she doesn't renew service subscription within 30 days after the current membership expires. This competition uses Log Loss as the evaluation metric which is described by following function:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i)),$$

where N is the number of observations, y_i is the binary target(1 means churn, while 0 means renewal) and p_i is the predicted churning probability of the i-th member.

In this competition task, we design a framework to predict the churn users, which includes data preparation, feature extraction and model ensembling. Figure 1 shows the overview of our framework. We first relabel some data in the origin training set, which are incorrect according to their transactions. We also generate external training data within previous months by conducting data augmentation. After data preprocessing, we respectively extract the features from three original files (transaction, member, user-log), representing category features by one-hot represetations and concatenating these features. To enrich features, we also induce non-linear and dense representations of features, by applying one unsupervised neural model, Denoising Auto-encoder [7]. Finally, we adopt two kinds of commonly-used classifiers as our basic models for churn prediction, i.e. gradient boosted decision tree models (XGBoost [1], LightGBM [4]) and neural network models [3]. And we generate our final submission by a linear combination of several models.

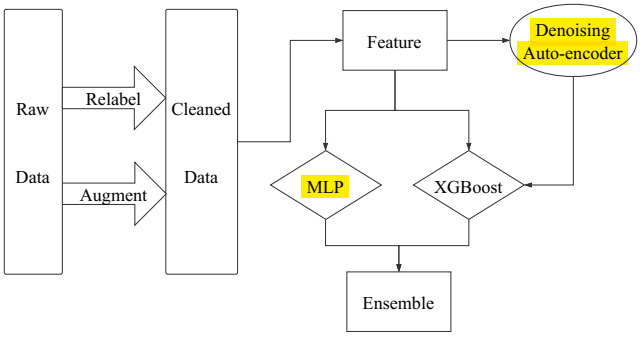

Figure 1: The overview of our framework.

The final evaluation results have been published on Kaggle. We finally score 0.09743 and rank 6th among 575 teams.

2 DATA PREPARATION

In this section, we introduce how we conduct data preprocessing, because the quality of data is critical for data mining tasks. We first give a formal definition of *churn* users based on official statement in Section 2.1. In addition, we generate our own labels and correct some wrong labels in original training set confidently. At last, data augmentation is performed for more training data, which empowers us to design more complicated models.

2.1 Problem Definition

A user is *churn* as long as he has no new valid subscription within 30 days after the current membership expires. Note that a new subscription is not sufficient in case of immediate abolishment. Given users whose subscription expires within the month of April 2017, we aim at predicting whether they will churn or not in the following 30 days.

2.2 Relabeling

When studying the training set, we found about 14000 users who were apparently renewal, yet labeled as *churn* incorrectly. For example, we list the last few transactions of user A[2] and user B[3] in Table 1.

Take user A as an example: her subscription expired at 2017/03/09 according to the first record. Next, there were two transactions of

*Corresponding author
[1]https://www.kaggle.com/c/kkbox-churn-prediction-challenge/leaderboard

[2]msno:*dW/tPZMDh2Oz/ksduEctJbsz0MXw3kay/1AlZCq3EbI=*
[3]msno:*pBXnYkKeo1HldqKCHfKN61EEwwEdZa4fD6c2AfK4a2k=*

Table 1: Examples of contradiction between user transactions and their original labels.

User	Transaction Date	Expire Date	is_cancel
A	2017/01/09	**2017/03/09**	0
A	2017/03/07	2017/04/08	0
A	**2017/03/28**	2017/07/07	0
B	2016/10/25	2017/02/22	0
B	2017/02/27	**2017/03/29**	0
B	**2017/03/30**	2017/09/29	0

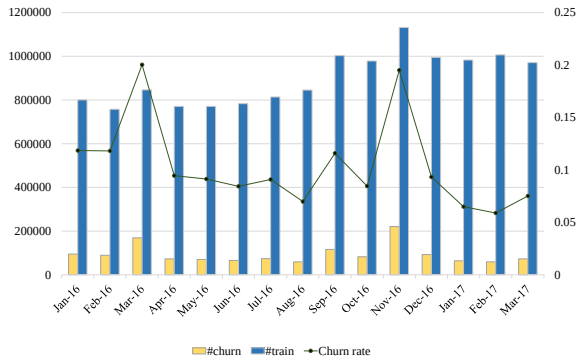

Figure 2: Churn rate of each month generated by wsdm-label scala.

her renewal to 2017/07/07, which were not canceled, so new valid subscription are observed within 30 days since her last expiration. For some unknown reasons, both of user A and user B are labeled as *churn* in the training set given.

In the stage of relabeling, we modify these incorrect labels to enhance data quality. The relabeling helped improve the performance of our model, which is shown in the Experiment Section.

2.3 Data Augmentation

Data augmentation is one of the effective ways to reduce overfitting by increasing the amount of training data. We propose a simple yet effective method of data augmentation, by generating a lot of new training data within different months. Specifically, in addition to two official data sets which contain the data of Feb and Mar 2017, we generated other 13 months of data from Jan 2016 to Jan 2017, based on the transaction records. The churn rates of each month are shown in Figure 2. It is noticeable that the time shift against prediction point of some augmented data is quite large, which may bring more noise. Consequently, instead of using all of the generated data, we carefully select only a few months of data based on the criteria of similar churn rate distribution.

3 FEATURE EXTRACTION

In this section, we detail the feature engineering process. The features are extracted from transaction records, user logs and user profiles. We also employ Denoising Auto-encode to induce the non-linear features from the original ones.

3.1 Transaction

The following features are generated using the transaction history prior to the specific time boundary.

Characteristic of a single transaction.

In addition to the original value of the most recent transaction record and some statistical features of the raw data (e.g. *pay_method* which is converted to one-hot vectors, *ave_plan_day*, *ave_plan_price*, *cancel_ratio*, *ave_pay*), we have also extracted some practical features.

- *pay_per_day*, the value of *actual_pay*/*plan_days*
- *plan_actual*, the value of *plan_prices* − *actual_pay*
- *auto_not_cancel_ratio*, the ratio of transactions which are auto pay and not cancel
- *nopay_ratio*, the ratio of free transactions
- *discount_i(i=80,90,100)*, the ratio of transactions which satisfy $plan\_price \cdot i\% \geq actual\_pay$

Characteristic of all transactions.

We extracted two potentially useful features:

- *ave_interval* means the average of the interval between consecutive transactions
- *max_expire_date* means the maximum expire date in a user's transaction records

The churn information of last month.

Because we need to use the transaction data and the rules as we mentioned in Section 2, we are going to classify these features here. In this part, we consider about whether the user appeared in the training data of last month and whether it was labeled as churn. By this way, we can get three features: *last_not_churn*, *last_churn*, *last_not_in*.

3.2 User logs

Our intuition is that the closer a record is to the predicted time, the more helpful it is to predict whether the user will churn or not. Besides, considering that a user's behavior may change from time to time, we generate user behavior features within different window sizes. Thus, we extract features according to four periods, i.e. a period of k month(s) from the month to be predicted, where $k \in \{0.5, 1, 2, 4\}$. The features listed below are generated for each period.

Features collected from the raw data.

We first generate some original features from the raw data and each feature has two types: the total count and average count within a period.

- *num_25*, the number of songs played less than 25% of the song length
- *num_50*, the number of songs played between 25% to 50% of the song length
- *num_75*, the number of songs played between 50% to 75% of the song length
- *num_985*, the number of songs played between 75% to 98.5% of the song length
- *num_100*, the number of songs played over 98.5%of the song length
- *num_unq*, the number of unique songs played

- *listen_time*, the time a user spend listening to music

Features generated from the raw data.

We combine numerical features and generate some statistical features which are the most important features in daily user logs.

- *log_cnt*, the aggregate number of daily user logs of a user
- *song_cnt*, the aggregate number of songs a user listened
- *listen_frequency*, calculated by the formula

$$listen\_frequency = log\_cnt/duration$$

- *song_per_day*, the average number of songs a user listened per day

3.3 Members

These features are from user information.

- *city_i*, whether the user is from i-th city, using one-hot encoding.
- *bd_valid*, whether his or her age is in $[10, 60]$.
- *bd_new*, equals $[\frac{bd}{5}]$ if bd is valid.
- *gender_str*$(str \in \{male, female, none\})$, the gender information.
- *registered_via_i*, the registration method.
- *registration_init_time*, the registration date, which is converted into the number of days.

3.4 Denoising Auto-encoder

In addition to the above features, we also consider non-linear features. These features are derived from Denoising Auto-encoder(DAE) in an unsupervised manner, with original ones as input. Compared with the linear features, the non-linear features can deal with curse of dimensionality and capture some underlying information.

Specifically, let $\tilde{x}$ be the partially corrupted k-dimensional feature vector of a certain user, while the size of the output vector y is the same as $\tilde{x}$. DAE aims to reconstruct the original representation x by neural networks. We use a DAE with six full-connected layers, where the first three layers are encoding layers and the last three layers are decoding layers. The dimensions are 128, 96, 64, 96, 128, respectively. When training, we do a feed-forward pass to compute activations at all hidden layers, then at the output layer to obtain an output y. To learn this model, the squared error loss between the output y and the input x is used as the objective.

After training the Denoising Auto-encoder, we use the 64-dimension vectors in the middle layer as the non-linear features.

4 MODEL

In this section, we describe two basic models we used and explain how to ensemble them.

4.1 XGBoost

We use XGBoost as our first model. Tree boosting is a highly effective and widely used machine learning method, which has been shown to give state-of-the-art results on many standard classification. It boosts performance by combining many weak tree predictors. XGBoost is one of the most competitive models among the competitions hosted by Kaggle. It is effective in dealing with a large number of features and non-linear interactions between the features and labels, which is suitable for churn prediction considering the features we have obtained. Moreover, it's convenient to get feature importance via XGBoost, which can help us to select useful features. XGBoost has three types of booster, namely Gbtree, Dart and Linear. Because of the non-linear between features and labels, we adopt Gbtree and Dart booster respectively. The depth of classifier is set to 7 and the learning rate is set to 0.02. The results are shown in section 5.

4.2 Fully Connected Neural Network

The second model we used is a simple fully connected neural network. Let $x^{(0)}$ be the k-dimensional feature vector of a certain user. The input is followed by four fully-connected layers. The first hidden layer has 512 units, the second hidden layer has 256 units and the third hidden layer has 128 units, the output layer is a sigmoid layer and calculates the probability of churn. Layer l can be represented as:

$$x^{(l)} = ReLU(W^{(l)} \times x^{(l-1)} + b^{(l)})$$

where $x^{(l-1)} \in \mathbb{R}^{n_l-1}$ is the input vector of layer l, $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix, $b^{(l)} \in \mathbb{R}^{n_l}$ is the bias vector, n_l is the size of vector $x^{(l)}$.

During training, we employ batch normalization [2] to accelerate the convergence. Besides, dropout [5] is used in each layer to prevent overfitting.

4.3 Model Ensembling

Both XGBoost and neural networks have their own advantages, for example, XGBoost can avoid over-fitting better while neural network is able to learn more complicated features. Therefore, model ensembling, which leaverages the superiority of both two models, can be very powerful to increase performance on this prediction task.

Based on the performance of each single model, we generate our final submission by a linear combination of three models, which can be represented as

$$Score = 0.4 \times nn + 0.4 \times xgb\_without\_dae + 0.2 \times xgb\_with\_dae,$$

where nn denotes neural network and xgb_w/o_dae respectively denote XGBoost models using and not using features generated by Denoising Auto-encoder. The combination coefficients are carefully tuned based on the performance of each single model on validation set.

5 EXPERIMENTS

In this section, we first demonstrate the effectiveness of the proposed data preparation, then we present the performances of each single model and the ensemble model. To tune the hyper-parameters, 5-fold cross-validation is used. We also visualize the user vector extracted from neural network and give a detailed discussion of five important features among the used features.

5.1 Performance of Data Preparation

As mentioned in Section 2.2, some labels of training data in Mar 2017 are wrong. We correct these wrong labels and compare

Table 2: Performance of Models with/without Data Relabeling.

Training Data	Private Board score
Mar.	0.13163
Mar. + relabeling	**0.10546**

Table 3: Comparison of model performance using different training data.

Training Data	Private Board score
Mar.	0.10546
Mar. + Feb.	0.10342
Mar. + Feb. + Jan.	**0.10202**
Mar. + Feb. + Jan. + Dec.	0.10304

Table 4: Performance of different models on augmented training data.

Model	Private Board score
NN:512 × 256 × 128	0.10102
DAE	0.10303
XGBoost:Gbtree	0.09989
XGBoost:Gbtree + DAE	0.09959
XGBoost:Dart	0.10030
XGBoost:Dart + DAE	0.10010
Ensemble	**0.09743**

the performances of models trained with original data and relabeled data using Multilayer Perception. The results are presented in Table 2, which shows that the relabeling achieves a significant improvement.

The dataset provided by organizers includes a lot of historical records, which allows us to do the data augmentation. To make full use of the data, we generate the training data of Feb. 2017, Jan. 2017 and Dec. 2016 using the label script offered. Table 3 shows the performance of a simple neural network (256 × 128 × 1) on each dataset. We find that simply adding the training data close to the month to be predicted can obtain a better performance, but adding the data far from now may lead to a decrease in model's performance. Based on these observations, we included the data of Feb. and Jan. into the new training set, which helps further improve performances.

5.2 Model Performance

We elaborately modify the architecture of fully connected neural network, tune the hyperparameters and achieved the best leaderboard score 0.10102 by it. We also train XGBoost classifiers (depth:7), add low dimension features generated by DAE and achieved the best single model score 0.09959. Based on model ensembling, we finally scored 0.09743 on the private leaderboard and ranked 6th among 575 teams.

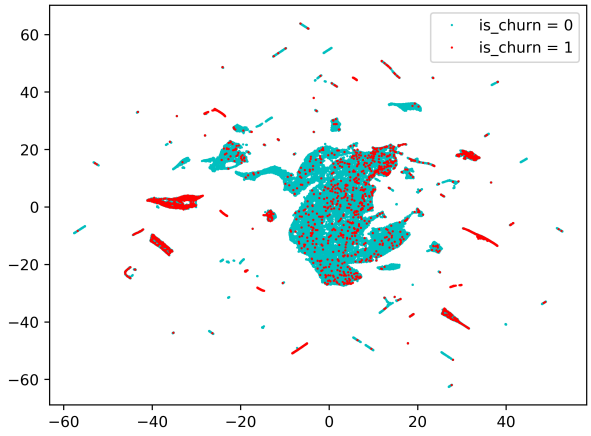

Figure 3: Low dimension visualization of user vectors generated by Multilayer Perception.

Figure 3 shows the 2-dimension visualization of user vector extracted from neural network generated by Largevis [6]. The churn users are divided into clusters apart from the users that are evenly distributed in the center of the picture. The figure indicates that the model can predict churn for users of certain behavior, but for those who act like most of the normal users, we still have to do more work to improve model's performance.

5.3 Feature analysis

Interpreting and analyzing important features enable us to gain profound domain knowledge. Figure 4 show the top 20 most important features evaluated by XGBoost. Most of them have good interpretability. Here we briefly interpret five important features.

Feature from transactions.

max_expire_date, a user's max possible expire date so far, which indicates the probability of being a long-time user.

last_trans_date, the latest transaction, which may indicate the activity level of a user within a certain period.

ave_interval, the average interval between every two transactions, which may indicate a user's subscribing habit. People of large *ave_interval* may renew their membership only when they need service, which may result in churn in this task.

cancel_cnt, the number of cancellations, which may indicate whether a user is satisfied with the service or not.

Feature from userlogs.

listen_frequency_k, a user's activity level within a certain period, which can directly reflect the demand of this kind of service.

From Figure 4, we can also see that most of important features are extracted from userlogs and transactions, which reflects that user behaviors are the most important information in churn prediction.

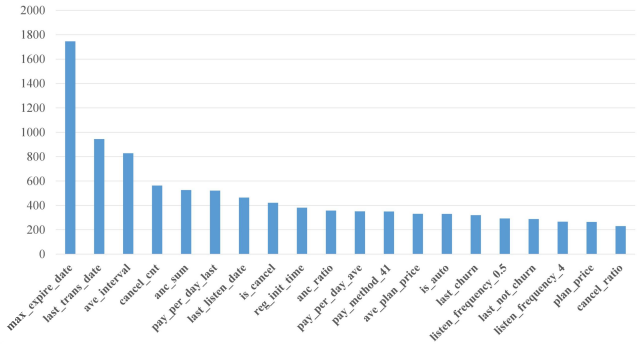

Figure 4: The 20 most important features evaluated by XGBoost.

6 CONCLUSION

In this paper, we describe our system in the KKBOX's Churn Prediction Challenge. We first conduct the data preprocessing, including relabeling and data augmentation, then elaborately extract features from three original files and apply Denoising Auto-encoder to get non-linear features. After that, we ensemble gradient boosting decision tree algorithms and neural networks to predict churn users. Finally, we analyze some important features to gain domain knowledge in this churn prediction task.

ACKNOWLEDGMENTS

This paper is partially supported by the National Natural Science Foundation of China (NSFC Grant Nos.61772039, 61472006 and 91646202). The first three authors are supported by the Top-notch talent program at Peking University.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[2] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 448–456. http://dl.acm.org/citation.cfm?id=3045118.3045167

[3] A. K. Jain, Jianchang Mao, and K. M. Mohiuddin. 1996. Artificial neural networks: a tutorial. *Computer* 29, 3 (Mar 1996), 31–44. https://doi.org/10.1109/2.485891

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3149–3157. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[6] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. Visualizing Large-scale and High-dimensional Data. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 287–297.

[7] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* 11 (Dec. 2010), 3371–3408. http://dl.acm.org/citation.cfm?id=1756006.1953039