A Practical Pipeline with Stacking Models for KKBOX's Churn Prediction Challenge

Zhinan Wang
Alibaba Group
yunyi.wzn@alibaba-inc.com

Wenming Xiao
Alibaba Group
wenming.xiaowm@alibaba-inc.com

Jun Wang
Alibaba Group
j.wang@alibaba-inc.com

ABSTRACT

WSDM - KKBOX's Churn Prediction Challenge refers to a binary classification task to predict whether a user will churn after their subscription expires on the service provider, KKBOX. As one of the most popular music streaming service, KKBOX holds a comprehensive collection of Asia-Pop music with over 30 million tracks. KKBOX provides unlimited services for millions of subscribers, earning revenue from advertising and paid subscriptions. In this paper, we designed a practical machine learning pipeline, including data preprocessing, feature engineering, as well as stacking of learning models, to tackle such a challenge. By extensive off-line evaluation, We are able to validate the performance of the proposed technique under various settings to optimize the proposed method. In addition, we analyze the results and gain useful insights into improving models for the prediction task. In the competition, we eventually receive the third place with an evaluation score of 0.09344.

CCS CONCEPTS

• **Computing methodologies** → *Classification and regression trees*; *Feature selection*;

KEYWORDS

features engineering, stacking technique, XGBoost

1 INTRODUCTION

Similar to the streaming music service providers Spotify and Apple Music in Europe and the United States, KKBOX is a popular music streaming service in Asia. The core business model of such type of service providers relies on advertising and paid subscriptions. For example, in 2015, Spotify generated 2 billion USD revenue, 89.9% of which is claimed by the paid subscriptions. In addition, the Ads revenue also heavily depends on the number of active subscribed users. Therefore slight variations in churn can bring a significant influence on revenue. It is critical to implement accurate churn prediction to achieve long-term business success for such streaming service providers.

In this draft, we will describe a practical machine learning pipeline to address the churn prediction problem. Note that the challenge provides the real-world data collected from KKBOX. It contains a variety of issues, such as incorrect records, heavily imbalanced classes, and missing values. Thus, we first develop a data cleaning and validation procedure to clean and improve the data qualities. Secondly, we perform feature engineering techniques to extract rich information from the processed data. The last procedure of the pipeline is to design a stacking technique based prediction model. In addition, without online AB testing strategy, the off-line evaluation platform is critical for the learner to achieve the best performance. In particular, we utilize the Log loss as the performance metric for the off-line evaluation.

Thereby quickly determining the validity of features. Second, the use of the distributed algorithm and sampling technique saved a lot of time for us. Third, in feature extraction, we transformed almost all the features based on the data distribution. Furthermore, stacking technique was used in learning algorithms so as to improve model accuracy.

The rest of the paper is organized as follows: Section 2 introduces the details of the challenge, followed by our proposed learning pipeline; In Section 3, we describe the off-line evaluation platform and the online performance generated by our models. Finally, we conclude our analysis of the task, as well as some additional discussion of the future directions in Section 4.

2 PROBLEM DEFINITION

The KKBOX's Churn Prediction Challenge of the WSDM Cup 2018 is to predict the probability to churn for a member, who's membership expire in the current month. Here the criteria of "churn" is defined as failing to renew valid service subscription within 30 days after the current membership expires. As shown in Table 1, the usable raw information includes user demographic data, past order transaction data, and user's listening log data. The setting of the challenge is to use the February and March results as training data and perform churn prediction for April.

For justifying the performance of the prediction, this competition uses Log Loss as the evaluation metric, which is defined as

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

where N is the number of observations, log is the natural logarithm, y_i is the binary target, and p_i is the predicted probability that y_i equals 1.

3 APPROACH

Considering the huge amount of data and iterative requirements, we carry out our experiment on DataWorks[1], a Big Data platform product launched by Alibaba Cloud, which provides one-stop Big Data development and a rich library of algorithms.

3.1 Preprocessing Data

There are several datasets provided for the competition: transaction details of users until 3/31/2017, daily user logs describing listening behaviors until 3/31/2017, and user information including age, city, and gender.

[1] https://www.alibabacloud.com/product/ide

Table 1: The summary of the KKBOX's data

Dataset	The number of data	Fields	Sample
transactions	22,978,755	msno, payment_method_id, payment_plan_days, plan_list_price, actual_amount_paid, is_auto_renew, transaction_date, membership_expire_date, is_cancel	ugLxwUWLxxJAky/qQPSa+sgrNXALHZPy7+aJH4MciTI=, 36, 30, 180, 180, 0, 20160824, 20160923, 0
members	6,769,473	msno, city, bd, gender, registered_via, registration_init_time	IsPZ1hDCz5Igcy4dV9QyBj581NN1OE9rpkFv6Th+xRI=, 1, 0, NULL, 7, 20120521
user_logs	410,502,905	msno, date, num_25, num_50, num_75, num_985, num_100, num_unq, total_secs	BFTYWX+yJQ1U0CHqMF2nffH7YqifwVldeTpge3jOUWg=, 20150816, 7, 0, 0, 0, 10, 17, 2762.455
train	1,963,891	msno, is_churn	waLDQMmcOu2jLDaV1ddDkgCrB/jl6sD66Xzs0Vqax1Y=, 1

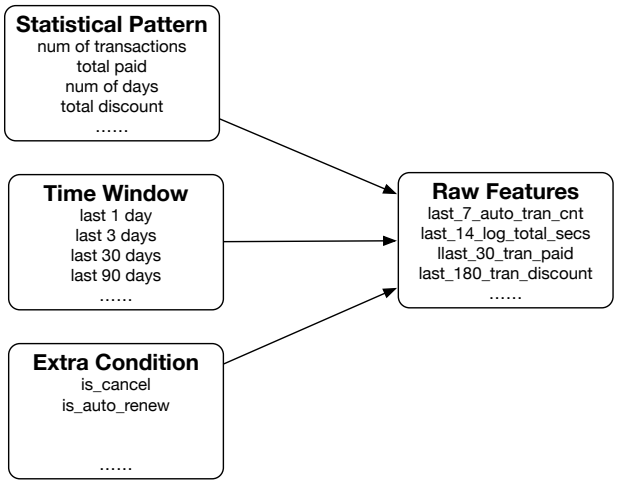

Figure 1: The feature extraction process with different settings.

The label of training data is_churn which defined as whether the user did not continue the subscription within 30 days of expiration. Two sets of training data can be found: one about users churned in February and one about users churned in March. Note that the raw information contains various types of dirty data due to the uncontrolled process of data capturing. For example, the field of age could be less than 0 or greater than 100. Other extreme values appear in different fields such as registration time and payment amount. We perform data cleaning process by either replacing the invalid values with reasonable estimations or delete the problematic data record if there are insufficient useful information in the fields.

In addition, there is a severe imbalance issue between positive and negative labels in the training data. For example, there are 992931 training samples in the February data, of which only 63471 data points are positive. In other words, only about 6.4% members stopped renewing their subscriptions. As discussed earlier, it is critical to identify such a small member population and encourage them to continue their subscriptions.

Conventional classifier algorithms like decision tree and logistic regression tend to have a bias towards classes which have a major number of instances. The data from the minority classes may be treated as noise and ignored during the learning process, thus we employ an under-sampling technique [5] on the negative data points as a data preprocessing step to adjust the ratio of positive and negative data. In particular, we applied random sampling from negative samples and set different values of the ratio between the numbers of positive and negative samples as $8:1$, $5:1$ and $3:1$. The sampled data set will be used to train the prediction models

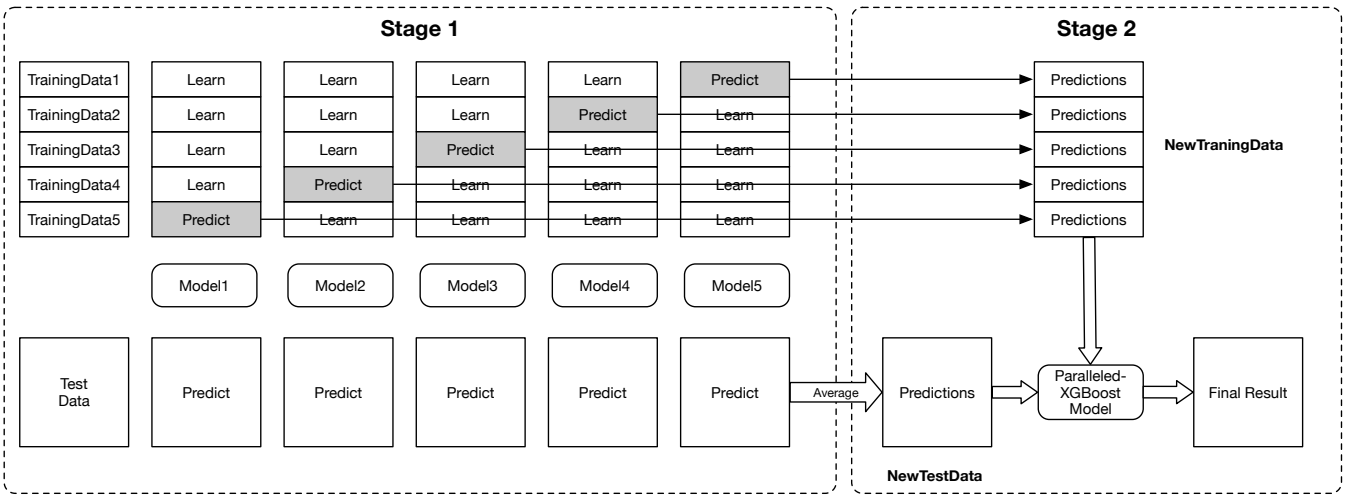

Figure 2: The 5-fold stacking strategy used in the two-stage training.

separately. The optimal value is determined based on the off-line performance evaluation of the training prediction models.

3.2 Features Engineering

Give the raw data representing user information, transaction log, and listening behavior, we integrate all the available information to extract features. There are three factors affecting the process of feature extraction, i.e., calculation method, the size of time windows, and all other conditions, as shown in Figure 1. For example, the extracted feature 'last_7_auto_tran_cnt', as shown in the right box in Figure 1, means the number of auto renew transactions in the last 7 days. Note that we also apply conventional ways for transforming the feature values. For some features with large values, we perform log-transformation. The one-hot encoder is utilized for categorical features.

In addition, an offline validation workflow was established in order to effectively justify the importance of features. In our evaluation, the followings features are identified as the most informative ones by the criterion of information divergence: i) the number of automatically completed transaction details in last 60 or 90 days; ii) whether the last transaction canceled or completed automatically; iii) the account registration method. Based on the calculated Log Loss using the off-line evaluation system, We have removed some features, which apparently contain very little useful information for prediction. Such a process can significantly reduce the storage cost and improve computational efficiency. Finally, we extracted more than 300 raw features and retain 204 numeric features for model training.

3.3 Learning Algorithms

Similar to our previous work in [6], we trained a two-stage model to perform the churn prediction. General speaking, in the first stage, features are fed into three models, logistic regression, random forest, and paralleled XGBoost. The parallel version of the XGBoost algorithm [1] has been proved to be efficient for large-scale applications [2, 3]. In our experiments, we have explored the implementation based on Alibaba Cloud. The outputs from the first stage will then be regarded as features for the second stage learning, forming a stacking model learning process.

In particular, for the stacking strategy, we follow the setting used in [7] to use a 5-fold stacking technique. The detailed process is illustrated in Figure 2.

In the first stage, we split the data into five folds and train five models using the leave-one-out strategy. The trained models are applied to the test data and generate one prediction for each model. In this way, on the training data, we obtain three different scores, generated from logistic regression, random forest, and paralleled-XGBoost, respectively. In order to perform prediction on the test data, we average these 5 groups of predicted scores from each model. Therefore we can compute the averaged prediction scores on the test data for three different models, i.e., logistic regression, random forest, and paralleled-XGBoost.

For the second stage, the prediction scores are treated as new training data and the average of the prediction on the test data from the first stage are used as new test data. The final results are generated using the new set of training and test with the paralleled-XGBoost model. The stacking technique [4] has the advantage in avoiding overfitting due to K-fold cross validation and interpreting non-linearity between features due to treating output scores as features.

3.4 EVALUATION RESULTS

Note that the proposed two-stage learning process contains a few technique components, such as prediction models (logistic regression v.s XGBoost), parameter optimization and stacking. We may apply different settings to obtain the final results. As discussed earlier, we built an off-line evaluation system, which can not only determine the optimal parameters but also help to design the best settings. In Table 2, we show five different settings and the corresponding performance. For instance, the pipeline that only applies logistic regression receives the Log Loss 0.2106 (the first line), while the combination of logistic regression and XGBoost can reduce the

Table 2: The score improvement each iteration make.

Features Enginering	Learning Algorithms	Log Loss on LB
Less than 100 features	LR	0.2106
Less than 100 features	XGBoost	0.1797
More than 200 features	XGBoost	0.1151
More than 200 features	Hyper Parameters Optimized XGBoost	0.0979
More than 200 features	Stacking Model	0.0934

loss to 0.1797. The optimal setting in the last line of the table derives the loss as 0.0934.

4 CONCLUSION

In this paper, we introduced a practical machine learning pipeline for the KKBOX's Churn Prediction Challenge of the WSDM Cup 2018. Our final model is ranked the third place on the private leaderboard. In the proposed pipeline, we highlight several key ingredients which play the critical role. For instance, in the data preprocessing step, we undersample the negative samples to alleviate the data imbalance issue and apply off-line evaluation to determine the optimal ratio. In the feature engineering part, we enrich the feature extraction by employing various types of data segmentation, such as time windows with different sizes and different calculation methods. Finally, in the model learning stage, we use a two-stage framework with stacking models to perform churn prediction. We also perform further analysis of the prediction results and conclude that the stacking technique indeed improves the accuracy. Our further work includes further optimization and understanding of the hyper-parameters, as well as the integration with recently developed deep learning algorithms.

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*. ACM, 785–794.

[2] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. 2012. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1223–1231. http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf

[3] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[4] Heikki Huttunen, Ke Chen, Abhishek Thakur, Artus Krohn-Grimberghe, Oguzhan Gencoglu, Xingyang Ni, Mohammed Al-Musawi, Lei Xu, and Hendrik Jacob Van Veen. 2015. Computer vision for head pose estimation: review of a competition. In *Scandinavian Conference on Image Analysis*. Springer, 65–75.

[5] Maureen Lyndel C. Lauron and Jaderick P. Pabico. 2016. Improved Sampling Techniques for Learning an Imbalanced Data Set. *CoRR* abs/1601.04756 (2016). arXiv:1601.04756 http://arxiv.org/abs/1601.04756

[6] Zhinan Wang Weiran Li Jun Wang Wenming Xiao, Mingdong Ou. 2016. Macro-Micro Scopic Temporal Regression of Academic Impact for Research Institutions: An Efficient Solution for KDD CUP 2016. In *The 2016 KDD Cup workshop of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[7] David H Wolpert. 1992. Stacked generalization. *Neural networks* 5, 2 (1992), 241–259.