



WILL A USER CHURN NEXT MONTH?

- A machine learning approach towards Online subscription Churn Analysis

Team: JCYY_KKBOX

Jiarui Tang: jt3869

Chutang Luo: cl5293

Yutong Chen: yc4127

Yingtian Liang: yl6479

1 CONTENTS

2	Business Understanding.....	2
3	Data Understanding	3
3.1	Translating the business problem into data analytic problem.....	3
3.2	Data source and description	3
3.3	Concrete Data Understanding through EDA.....	5
3.3.1	transaction.csv	5
3.3.2	member.csv	5
3.3.3	user_log.csv	5
3.4	Data Preparation.....	6
3.4.1	Challenges of processing big data.....	6
3.4.2	Dealing with abnormal and missing values	6
3.4.3	Recoding Categorical Features	6
3.4.4	Initial Feature Extraction based on domain knowledge and EDA	7
3.4.5	Train/test split	8
3.5	Modeling & Re-Feature Engineering.....	9
3.5.1	Baseline Model Selection.....	9
3.5.2	Baseline Model Performance	10
3.5.3	Model Improvement with baseline features.....	10
3.5.4	Clearing invalid entries of training data.....	11
3.5.5	Further Feature Engineering	12
3.5.6	Further Iteration	13
3.6	Model Evaluation & Deployment.....	16
3.6.1	Final Model Configuration.....	16
3.6.2	Final Model Performance.....	16
3.6.3	Deployment.....	16
3.7	Conclusion and Future Improvement Ideas	18
3.8	Appendix.....	19
3.8.1	Team Member Contributions	19
3.8.2	Kaggle Performance.....	19
3.8.3	Reference	19

2 BUSINESS UNDERSTANDING

KKBOX is a subscription-based streaming music service. In this project, our goal is to accurately predict whether a user will churn after their subscription expires. Similar to other companies with subscription-based service, user churn poses challenges on the long-term development of KKBOX. Thus, timely identifying whether a given user would churn is critical to KKBOX. Identifying churn accurately can help improving the company's efficiency in customer relationship management (CRM). An intelligent CRM framework can lead to mounting business value. Firstly, an agile process results in better resource allocation and cost management. For example, the company can actively take actions to retain users that are likely to churn, as well as get feedback from them to optimize the current product in order to avoid further churn. Secondly, successful CRM can boost customer loyalty and extend the time a customer stick with a company to maximize customer lifetime value (CLV) of our existing customers.

We are using supervised machine learning techniques to estimate the probability of whether a given customer is going to churn. By using key features of user information (i.e. previous subscription, music streaming history, personal information, etc.) as predictors, we can estimate how likely a given user is going to churn. We first carry out an exploratory data analysis (EDA) to explore each variable in the raw data sets. According to the results of EDA, we move on to the data cleaning process. Besides preprocessing abnormal and missing values, we also do some basic feature engineering, exploring features that could best capture churn. With the cleaned dataset, we first run a baseline model, based on which we develop more features. Then we iterate our baseline model by tuning features, algorithms and parameters, and eventually we get our final model to make churn prediction. Our model would allow the company to identify users with high probability of churn and develop specific strategies towards them. In the meantime, the interpretation of the model can help us understand user behavior (what makes a user churn).

3 DATA UNDERSTANDING

3.1 TRANSLATING THE BUSINESS PROBLEM INTO DATA ANALYTIC PROBLEM

Churn analysis is a classic classifying supervised learning problem. Our target variable is a binary variable `is_churn`. To be more specific, `is_churn = 1` if the user has churned. The exact decision criteria for churning is no valid new subscription record within 30 days from last expiration date.

3.2 DATA SOURCE AND DESCRIPTION

Our dataset is from KKBOX Music Recommendation Challenge in Kaggle. KKBOX is a company providing subscription-based streaming music service from which our data is donated. The KKBOX dataset is composed of 9 different files, including `train_v1` as train data of users whose subscription expires in February

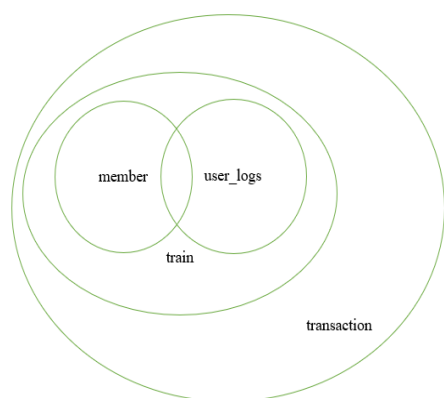


Figure 1

2017, `train_v2` as an update of training data with users whose subscription expires in March, 2017. `transaction_v1` and `transaction_v2` corresponding to transaction records before 2017/02/28 and records after 2017/03/31. `Members` contains all the general information available for users. `User_log_v1` and `user_log_v2` contains user log information (songs playing behavior).

The data comes in a neat tabular form. The VENN diagram shows the mutual overlap and containment relationships among the distinct user ids in those 4 tables. Obviously, when joining those tables by user id, missing values will be generated due to the different set of users included in different tables. We will discuss it in the following parts.

Dataset	Fields	# of data
transactions_v1 transactions_v2	msno, payment_method_id, payment_plan_days, plan_list_price, actual_amount_paid, is_auto_renew, transaction_date, membership_expire_date, is_cancel	21547746 1431009
members	msno, city, bd, gender, registered_via, registration_init_time	6769473
user_logs_v1 user_logs_v2	msno, date, num_25, num_50, num_75, num_985, num_100, num_uniq, total_secs,	392106543 18396362
train_v1 train_v2	msno, is_churn,	992931 970960

Figure 2

3.3 CONCRETE DATA UNDERSTANDING THROUGH EDA

3.3.1 transaction.csv

Most features in transaction are self-explained, except for is_cancel. With a cross table of the frequency of auto_renew and is_cancel, we found that when is_cancel = 1, auto_renew = 1. Hence, we concluded that is_cancel is a feature describing whether the user cancels his/her auto_renew.

3.3.2 member.csv

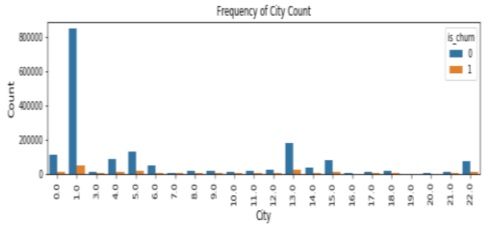
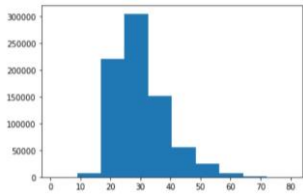
fieldname	distribution/explanation	note
msno(userid)	1.distinct in member.csv 2. some user doesn't have member information	
initial_registration_date	6 users whose initial registration date is later than their expiration date	
city		large spread across 22 different cities, with majority condensed in several large cities
bd(age)		1. crazy outliers (i.e. 2000, - 3100). 2. Skewed towards younger population, with a median at 26 and IQR within 22 and 35

Figure 3

3.3.3 user_log.csv

The user_log file shows user's log-in and music streaming behavior for each day. It includes log-in date and corresponding streaming activity such as total song listened, num_25(# of songs played less than 25% of the song length), total time spent that day, etc.

3.4 DATA PREPARATION

3.4.1 Challenges of processing big data

The first challenge we faced is the considerable scale of our data. Our dataset contains 41.3GB with the `userlog_v1` table taking up ~28GB, which is beyond the analyzing capacity of most personal computers. It's certainly impossible to read in data directly to pandas dataframe, thus we started by using pandas `chunksize = 5,000,000` and `chunksize = 1,000,000`, which led to memory error. After the failing attempt, we used PySpark and had successfully input all the files. However, we could only read the data, and we were unable to do other operations such as filtering, joining tables. Through active research and exploration, we finally identified our savior- Google Cloud Platform, BigQuery and Datalab. Using google cloud service and bigquery interface, we were finally able to process the `user_logs` data and export the results for further analysis.

3.4.2 Dealing with abnormal and missing values

We start our data preparation process by first removing abnormal values. Specifically, we remove the 6 members whose initial registration date is later than their expiration date, as well as whose age is outside of the range 3-year-old (kindergarten entrance age) and 90-year-old.

As for missing values, we add a new category for missing values in categorical data (i.e. city 0, register via 0), and we replace numerical null values with 0.

3.4.3 Recoding Categorical Features

In order to avoid redundant levels in the categorical variables and deal with rare levels to make our data less sparse, we combine levels into bins that collectively reflect the splits of the categories and encode them with new dummy variables.

1. For city, since city 1 is the majority, so we make it into a separate category. The frequencies of city 4, 5, 6, 13, 14, 15 are similar, thus we combine them into the second category. For other cities with only a few instances, we combine them into the third category. We also encode the missing value into 0

2. For Age, according to the age distribution, we recode age from 3 to 26 into '0', from 26 to 90 into '1', and other values into '-1'.
3. We also drop gender in member.csv. We discard gender feature because of its high missing rate (i.e. 60%) and similar frequency between each sub-group (i.e. ~50% male / female).

3.4.4 Initial Feature Extraction based on domain knowledge and EDA

We use features adapted from variables in 3 tables: user_logs, member, and transaction to run our baseline model. Based on domain knowledge and EDA, we did a baseline feature engineering and added some new features:

fieldname	distribution/explanation	note
total_songs	Total number of songs the user have listened	In last 30 days
like_rate	proportion of songs that are played over 75% of the whole song	In last 30 days
unlike_rate	proportion of songs that are played over 75% of the whole song	In last 30 days
avg_secs	average seconds per song listened	In last 30 days
registration_days	last day of the expiration month - members.registration_init_time	
n_trans_uniq	Number of distinct payment method	In last 30 days
have_userlog/have_members	Whether the user have records in user_logs/member	
n_cancel	Times that user has canceled his/her auto renew	

Figure 3

Why have_userlog/have_members: if the user has user_logs data, the proportion of churn closely resembles the proportion of overall data. However, if the user doesn't have user_logs data, the fraction of churn would be significantly higher than the overall fraction. Similar case also applies to have_members.

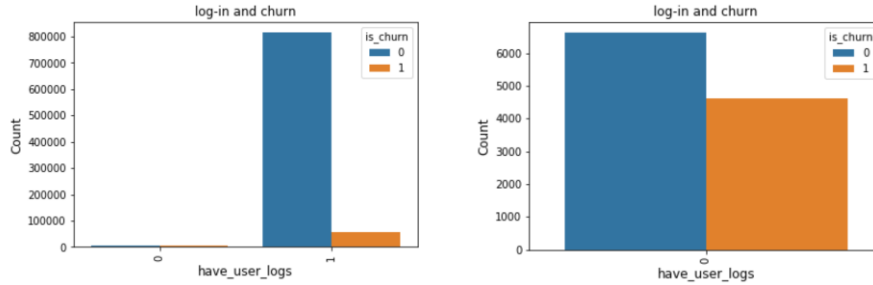


Figure 4

In conclusion, our baseline features include the original features in dataset and new features described above.

We use dummy coding for categorical variables.

3.4.5 Train/test split

In the process of identifying unique instance, we need to group targets by UserId and the expiration time. Thus, we need to perform separately on files with different expiration months. What's more, we only have transaction data and user_log data until 2017/03/31 while the prediction problem has included users with expiration dates in April, hence it is infeasible for us to look at user activities for time interval exactly within 30 days before the expiration. Due to the constraint, we decided to extract user behavior of the natural month before the expiration month, e.g. for user with expiration date 2017/03/23, we operate with his/her behavioral data before 2017/02/28.

As for users who don't have user_log but are both in train_v1 and train_v2, we need to delete their record in train_v1. We also find that some users have records in both months, probably because the user forgot to cancel the auto renew subscription in February, but then canceled in March. If a user does not have log in activity, it might not be a fan of the app. Thus, users without user_logs records are less likely to be committed users who are more likely to churn. For this kind of users, we only keep their latest churn records in order to capture their latest activity.

The process of splitting train and test data are as follows: firstly, we join the train data with transaction, user_log and member information by user id. We remove the records of users in train_v1 who have no behavior data but appear both in train_v1 and train_v2. To make sure in training and testing sets the proportions of users without user_logs record is the same, we use stratified sampling. We use sklearn to split train, test and validation data with proportion of 75% and 25% respectively, then we concatenate the data from different stratifications and get the final version of training and test data. As a Kaggle project, the reasons why we don't use submission data as our test data are: firstly, we can't obtain direct access to the score of submission data, which made it inefficient for us to process. Secondly, we want to look at the generalizability of our model into next month, thus the submission result is an important indicator of prediction accuracy of the following month.

3.5 MODELING & RE-FEATURE ENGINEERING

From the requirement of the challenge, the evaluation metric we are using for modeling is Log loss. AUC score was used to select model. When the performance of models is very similar, we may choose 3-fold cross validation to help with model selection.

3.5.1 Baseline Model Selection

To select a model that best capture the features, we deployed models that are widely used for classification on our baseline features, including decision tree and logistic regression.

3.5.2 Baseline Model Performance

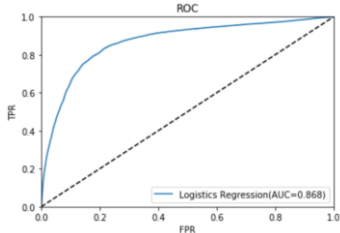
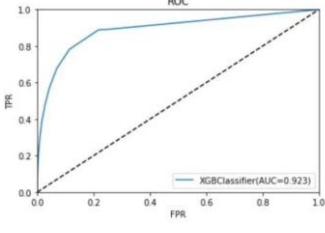
	Logistics Regression	Decision Tree
AUC	0.869 	0.708 
Log loss	0.201	2.991

Figure 5

Since the logistics regression performs better on both AUC scores and log loss, we chose logistic regression as our baseline model.

3.5.3 Model Improvement with baseline features

Since logistic regression model has some limitations when we have many features (e.g. possible collinearity between features), we decided to employ more advanced classifier model to train our data. Demonstrating their capacity of prediction many times, random forest and XGBoost may improve our model performance. Therefore, we fit one random forest model and one XGBoost model with default hyperparameter and baseline features, whose results are shown below:

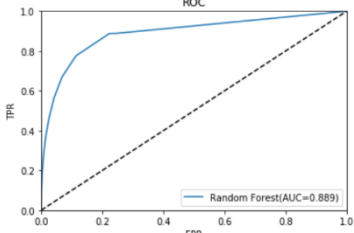
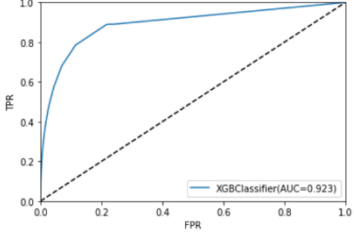
	Random Forest	XGBoost
AUC	0.889 	0.708 
Log loss	0.450	0.165

Figure 6

From the table shown above, we can see that XGBoost has slightly lower AUC score but much lower log loss. Therefore, we have decided to use XGBoost and focus on its performance in future model improvements. Since the algorithms we deployed have already reached its utmost performance, we tend to think about adding more useful features to improve the model.

3.5.4 Clearing invalid entries of training data

According to data description, there are some invalid entries in training data which do not fall in our scope of prediction in a specific month. For example, we have a sequence for a user with the tuple of (transaction date, membership expiration date, and is_cancel): (2017-01-01, 2017-02-28, false), (2017-02-25, 2017-04-03, false), (2017-03-15, 2017-03-16, true), (2017-03-18, 2017-04-02, false)

Note that even the 3rd entry has membership expiration date falls in 2017-03-16, but the fourth entry extends the membership expiration date to 2017-04-02, not between 2017-03-01 and 2017-03-31, so we should not include this user in those whose expiration date was in 2017 March.

This kind of invalid entries included in the training dataset may mislead our model and give wrong predictions on other datasets. Therefore, we cleaned this kind of invalid entries within the original training data and run the models again on the baseline features. The result is shown below:

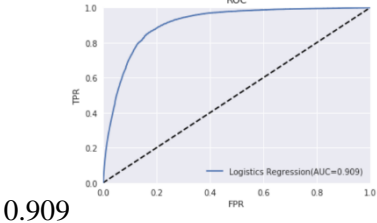
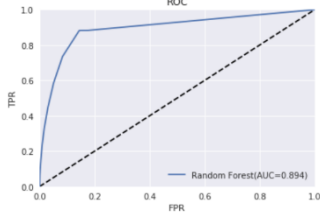
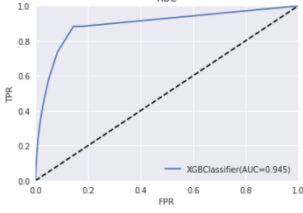
	Logistic Regression	Random Forest	XGBoost
AUC	 <p>0.909</p>	 <p>0.894</p>	 <p>0.708</p>
Log loss	0.130	0.289	0.165

Figure 7

From the table above, we can see that the AUC score of different models has increased, and The log loss has greatly decreased. Cleaning the invalid entries is of great necessity and helps to achieve better model performance.

3.5.5 Further Feature Engineering

Based on our baseline model, the correlation of existing features, and further research on churn analysis, we started brainstorming new features. With new feature engineering, we obtained 36 new features as follows:

Firstly, we observed that avg_list_price and AVG_actual_paid has a high correlation of 0.995311. Looking closer at their difference, we found that nearly 99% of $\text{avg_list_price} - \text{AVG_actual_paid} = 0$. Thus, we decided to drop avg_list_price based on feature importance given by decision tree results. Next, with reference to several papers and projects on Kaggle, we come up with some new features on transaction and userlog:

3.5.5.1 transaction

last_churn_n: At the beginning, we performed aggregation on transactions for each userid as new feature indicating average behavior of transactions that user has made in his/her history. However, we found out that these features are intuitively correlated. We need some new features to depict the user's pattern of churn. Hence, we come up with last_churn_1:5 which represents the churn wave of users based on their most recent 5 expiration dates. If churn on the last nth expiration: last_churn_n = 1, if not churn or doesn't have that much expiration: 0. We choose n=5 because the highest number of expiration records within the time interval is 6, and since the sixth churn is already encoded with is_churn, we only use the 5 records prior to that.

last_auto_renew: What's more, since auto_renew users will be auto charged subscription fee, thus the status of auto_renew would be a strong indicator.

3.5.5.2 user_logs

Intuitively, we believe that user_logs

should be a powerful source of

information. In baseline model we

obtained basic description of user's

average behavior. Similar to

transactions, we want to analyze user's

behavior over time. The specific

procedure is concluded as follows with

explanations:

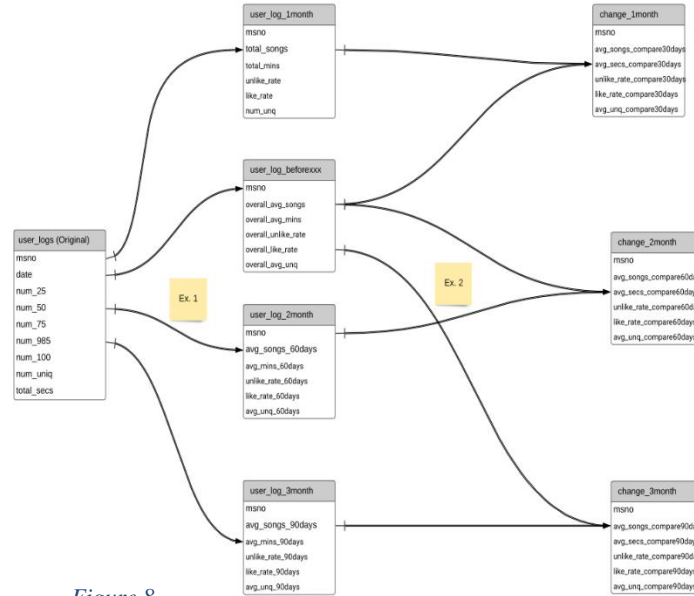


Figure 8

Ex.1. Wrap user_log data into chunks of

1 month (the month before expiration), 2 months before expiration, and 3 months before expiration

month. Scale with month: we would like to encode two aspects of user behavior: user's log-in activity and

music-streaming activity. Scaling with day would only reflect the pattern of user's music-streaming

activity, while scaling with month(30.5days) could reflect both activities.

Ex.2. To show change of behavior, we wrap each feature as $\text{avg_feature_change} =$

$\text{user_log_before_expmonth.feature} - \text{user_log_thattimeperiod.feature}$. In order to avoid data leaking, we

made sure that the time interval has no overlap.

Ex.3. Also, we changed the unit of seconds to minutes.

3.5.6 Further Iteration

3.5.6.1 Fitting with new feature set

When fitting XGBoost models, we found that XGBoost is slow in computation. Therefore, we tried to

find whether there is a model easier at computing but have great performance. LightGBM is a fast,

distributed, high-performance gradient boosting framework improved over XGBoost. In general,

LightGBM has similar performance as XGBoost but much more efficient. Considering the computational

limitation, we decided to try LightGBM to see whether we can change our model to LightGBM, which may greatly improve the speed of tuning.

The table below shows the model performance on test data we fit with XGB and lightGBM:

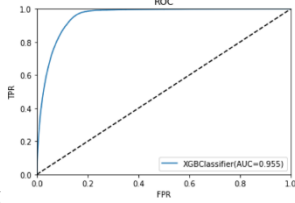
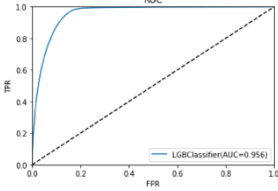
	XGBoost	LightGBM
AUC	 <p>0.955</p>	 <p>0.956</p>
Log loss	0.0997	0.09964

Figure 9

3.5.6.2 Feature selection

Observing from the performance, we can see that XGB and LightGBM perform very similar. To further check the performance of both models, we combine our training data and test data, and then do 3-fold cross validation to get the test performance. The difference between AUC scores of LightGBM and XGB is not too much (one 0.827 and the other 0.934) but the log loss of LightGBM is much worse than that of XGB, which may indicate LightGBM is overfitting. Therefore, to avoid serious overfitting problem, we choose to stay on XGB algorithm.

After determining the algorithm we are using, we put all the features we've got so far into XGB model to select a feature set for our final model. Then we get feature importance scores from XGB results and use backward stepwise selection method to deploy feature selection. The general procedure of the backward stepwise selection is:

Step 1. Start with all features, fit XGBoost model and get AUC, log loss score.

Step 2. Drop the feature with the lowest feature importance score, fit XGBoost model again and get AUC, log loss score.

Step 3. Repeat Step 2 until there is only one feature left.

Step 4. We choose the optimized feature subset with the optimal AUC and log loss score.

Finally, we obtained a new feature set with 19 new features. Fitting the XGB model with these new features, we obtain feature importance of each features as follow:

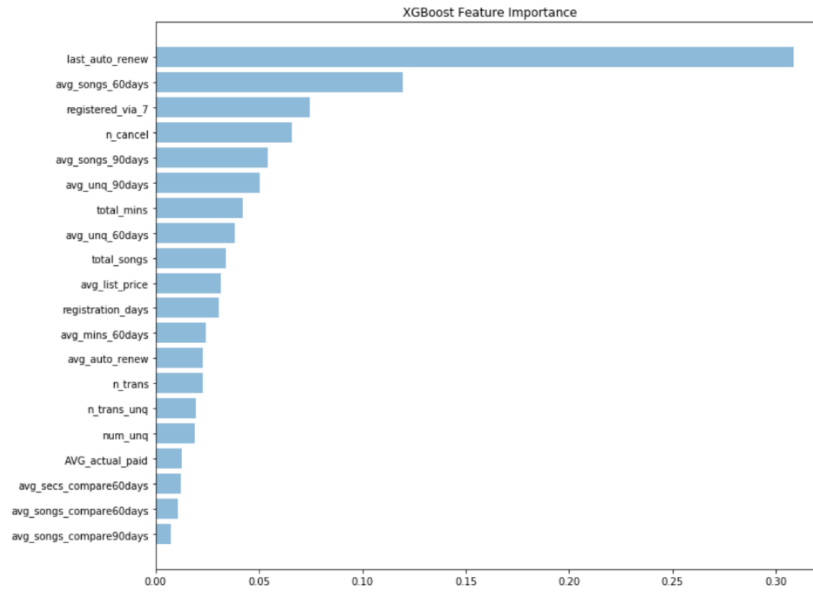


Figure 10

3.5.6.3 Hyperparameter Tuning

We use GridSearch to iterate through every combination of `learning_rate`, `max_depth` and `min_child_weight`.

3.5.6.4 Stacking

Since our auc score and log loss do not have much improvement after feature selection and tuning, we decided to use Stacking ensemble learning technique, combining XGBoost and LightGBM classification models via a meta-classifier (we choose XGBoost as our meta-classifier model), to make our final model less subject to overfitting and more able to make accurate predictions. However, the stacking model does not outperform the XGBoost model, with an AUC of 0.678 and log loss of 0.164. Therefore, we finally determined our final model as the XGB model with 19 features shown in the feature importance plot.

3.6 MODEL EVALUATION & DEPLOYMENT

3.6.1 Final Model Configuration

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bynode=1, colsample_bytree=1, gamma=0,
               learning_rate=0.1, max_delta_step=0, max_depth=6,
               min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
               nthread=None, objective='binary:logistic', random_state=0,
               reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
               silent=None, subsample=1, verbosity=1)
```

3.6.2 Final Model Performance

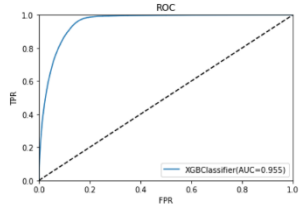
Model	AUC	Log loss
XGBoost	0.955 	0.098

Figure 11

3.6.3 Deployment

We aim at training a classifier to predict churning users in April, which can be generalized into predicting user churn. What's more, based on significant features, we can provide recommendations for companies to prevent churning in advance.

1. Model implementation and generalization

Our model can be deployed as a churn prediction model, which can be integrated into the automatic customer management system connected to database. The program can generate alerts informing staff about churning users so that timely customer retention activity can be performed. To effectively monitor and update the process, newly obtained data must be fed into the algorithm and the prediction model should be constantly iterated and re-trained based on the latest predictions and actual results.

However confident we are with our model, there are still some issues needed to be stressed.

Firstly, although our prediction for April churn has a log-loss of 0.118, we are only confident in generalizing our model to the next month, since the features are correlated with time, we can't safely apply it with large time gap.

Next, similar to traditional churn analysis, we emphasize the use of customer panel data to model the dynamic evolution of a customer base. The model is based on the premise that companies maintain a storage base of user information. Storing customer information and use them for prediction would pose privacy issues concerning private information safety. To address the issue, the company must ensure masking methods.

2. Business implementation based on selected features

Based on the feature importance, we have identified several important values in churn prediction, from which we can derive three important business insights.

a. `last_auto_renew` and `n_cancel`: Users who actively cancel their `auto_renew` is more likely to churn, which implies that the company should work towards getting members register for `auto_renew` and pay attention to users who just cancel their `auto_renew`. For example, discounts can be applied for users who enroll in `auto_renew`. Also, send emails to users who just canceled their `auto_renew` about renewing their subscription.

b. `registration_via_7`: Surprisingly, `registration_via_7` shows a large feature importance, which shows that 7 is a popular platform for loyal users. Thus we recommend KKBOX to invest more with platform 7 for advertising.

c. When looking into music streaming activity, we use features that describe three behaviors: number of songs, total time spent listening to music, and whether the user like the song(finish the song). From the ranking of feature importance we can observe that features describing number of songs outperform other `user_log` features, which means that as long as users maintain the level of songs they listen per day, they are still enjoying the app. Thus the company doesn't have to worry much about customized music recommendation.

3.7 CONCLUSION AND FUTURE IMPROVEMENT IDEAS

Due to resource and computational constraints, we did the best we can do. However, we do bear in mind that there's neither perfect model nor accurate prediction in the real world. After reflected upon our efforts, we had come up with some ideas for future improvement.

After adding new features depicting user behavior throughout time, our model is significantly improved, which means that even including information of the same behavior in different time series could make a difference to the model. Thus, in the future, we could incorporate time series analysis into our feature engineering process in order to better utilize the panel data.

Next, except from plainly conducting binary classification problem of churn, we can further extend the problem into survival analysis. For example, we can delineate the pattern of a user's survival cycle , facilitating a more delicate customer relationship management.

We are dealing with a massive and dirty dataset. Throughout the project, we dedicated most of our effort to feature engineering. But we believe that there's other creative ways of wrapping new features. The process of exploration, understanding data and feature engineering is essential in the project and we all enjoyed brainstorming new features. However, with more feature we risk overfitting the model (which we believe is one of the problems in our final model). Due to time constraints we have made two extreme approaches towards bias-variance tradeoff. In the second model improvement, we sacrificed variance to minimize bias by using all newly-created features. For our final model, we selected a relatively small subset of features to mitigate over-fitting.

We hope that with future studies, we would be able to come up with better solution addressing aspects such as bias-variance trade-off, trend analysis and dealing with dirty data in general.

3.8 APPENDIX

3.8.1 Team Member Contributions

As a team, we enjoyed the process of bringing structure to this dataset and finding compelling patterns, we enjoyed working together and we all learned a lot.

We all participated actively in discussion towards wrapping new features and model selection throughout the whole project

We all contributed to coding and write-up efforts

First step towards big data solution and initial data exploration analysis is carried out by Chutang and Yutong

Model fitting and selection is carried out by Chutang and Jiarui

Final write-up, formatting and proof-reading is mostly carried out by Yutong and Yingtian

3.8.2 Kaggle Performance

After submitting our prediction results to Kaggle on another new dataset, our log loss is 0.11851, which ranked 50 on the private leaderboard and 48 on the public leaderboard (before 7:00am Dec 9th). There are totally 575 people on the ranking leaderboard.

3.8.3 Reference

Brownlee, J. (2019, August 21). Feature Importance and Feature Selection With XGBoost in Python. Retrieved from <https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/>.

Headsortails. (2018, November 4). Should I stay or should I go? - KKBox EDA. Retrieved from <https://www.kaggle.com/headsortails/should-i-stay-or-should-i-go-kkbox-eda>.

jeru666. (2017, November 16). Memory Reduction and Data Insights. Retrieved from <https://www.kaggle.com/jeru666/memory-reduction-and-data-insights>.

joshwilkins2013. (2017, October 29). Churn Baby Churn - User Logs. Retrieved from <https://www.kaggle.com/joshwilkins2013/churn-baby-churn-user-logs>.

Kasturi, S. N. (2019, July 11). LightGBM vs XGBOOST: Which algorithm win the race !!! Retrieved from

<https://towardsdatascience.com/lightgbm-vs-xgboost-which-algorithm-win-the-race-1ff7dd4917d>.

Kevinbonnes. (2017, September 21). [R] Churn prediction baseline. Retrieved from <https://www.kaggle.com/kevinbonnes/r-churn-prediction-baseline>.

Rastaman. (2017, September 30). Churn or No Churn - Exploration Data Analysis. Retrieved from <https://www.kaggle.com/rastaman/churn-or-no-churn-exploration-data-analysis>.

Swalin, A. (2019, June 11). CatBoost vs. Light GBM vs. XGBoost. Retrieved from <https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db>.

WSDM - KKBox's Churn Prediction Challenge. (n.d.). Retrieved from <https://www.kaggle.com/c/kkbox-churn-prediction-challenge/discussion/46078>.