Churn Prediction Analysis in a Music Streaming Service (KKBox)

Xiaohan Xue xx715@nyu.edu

Jie Gu jg6617@nyu.edu

Fan Zhang fz2068@nyu.edu

Haodong Ge hg2363@nyu.edu

Abstract

Effective customer churn predictive modeling is applicable to many aspects of business including proactive customer marketing, sales forecasting, customer acquisition and retention. In this paper, we designed a integrated practical machine learning pipeline to predict whether a user will churn after the membership subscription expiration on KKBox dataset. In particular, we exploited features creation across various time windows using large-scale temporal data. We implemented various machine learning techniques in feature engineering and modelling, including XGBoost to improve our final model performance. In addition, based on the model results, we proposed several deployment strategies to help address customer needs and business growth for the company.

1 Business Understanding

Churn is a common phenomenon for companies providing subscriptions. In general, it refers to the behavior that customers leave a product or a service over a given period of time. Since acquiring new customers is usually more expensive than retaining existing customers (Brisco,2019), companies are now actively seeking to discover means of retention strategies that can retain existing customers and reduce customer churn rate. Companies that can keep more long-term customers are regarded to have better brand loyalty and reliability, building upon revenue in a more predictive manner. There are many possible causes of abandoning the brand (1), for example, poor customer service making it hard for customers to communicate their needs or give feedback. The one-time user may be scared away by ambiguous information, and the long-term users may look for alternatives if they feel they are not appreciated. Thus, when making predictions based on customers' demographics and past behaviors, companies also value what factors contribute more to the predictions.

In our study, We build a classification model to predict which category, the labels (churn or not churn), a customer belongs to. More precisely, if a user does not make a new service subscription transaction within 30 days after the current membership expiration date, we consider this as "churn". Although KKBox is a leading music streaming service in the Asia, it is crucial to maintain its growth in the market, where there are many existing and rising competing music service providers such as Spotify. Among all the challenging opponents in industry, KKBox can enhance its competitiveness with help of data science. By means of machine learning technique, we can extract business insights from the users' log and transaction history and further apply churn prediction into real-world context and develop corresponding intervention strategies.

2 Data Understanding Exploratory Analysis

2.1 Data Description

Our dataset comes from KKBox's Churn Prediction challenge. The data provided contains two versions: Original version with data before Feb. 2017 and updated version with new data of Mar.2017. Each version of data is composed of four separate datasets: members, transaction, user log and labels. More specifically, labels contain about 900 thousands user id and their churn result. Members dataset contains demographic information of users like age, gender and city. Transaction and user log datasets contain transaction and daily activity of all users and thus have over millions of data instances. The entire data set was more than 30GB. After running pre-processing and feature engineering, the structure data used in modeling contains 700 thousands "user_id" and 80 features.

Datasets	*Size*	*Features*
Transactions	22,978,755	msno, payment_method_id, plan_list_price, actual_amount_paid, is_auto_renew, transaction_date, membership_expire_date,is_cancel payment_plan_days,
Members	6,769,473	msno, city, bd, gender registered_via, registration_init_time
User_logs	410,502,905	msno, date, num_25, num_50, num_75, num_985, num_100, num_uniq, total_secs
Train	963,891	msno, is_churn

2.1.1 Selection Bias

With over ten millions data instances collected in transaction and user logs, our datasets contain comprehensive information about KKbox users. On most dimensions such as time and payment, our data are less likely to be severely biased. However, according to the EDA of member dataset, most users are from several major cities of the country. In other words, the overseas users are basically excluded in the dataset, and existing data are severely skewed to major cities. Thus, there's a potential problem of generalizing our finding on the user population from suburban and overseas regions. Moreover, the transaction data are naturally collected through major payment methods of credit card and online payment. The transaction with a less convenient method to collect will be less likely to be included in the existing dataset. Finally, potential publication bias may occur because the KKbox may have business concerns to keep its privacy.

2.2 Exploratory Data Analysis

2.2.1 Target Variable

Target variable in our problem is whether the user will churn after the expiration date. It is a binary variable "is_churn": 1 if the customer churns and 0 if not. From Figure 1, it is clear that customers who churn are the minority, which only counts for 8% of the total population. In this way, the data is highly imbalanced. For this classification problem with imbalanced dataset, we have figured out several useful techniques to alleviate the hurt of ineffectiveness during the modelling process. Details of dealing with imbalanced data will be shown later in this report.

2.2.2 Member

In the member dataset, it lists the customer user id and corresponding city, age, gender, registration method as well as registration date. Each user only has one record, i.e., user id is the unique key in this dataset. From this dataset, we learned that the customers come from 21 different cities. Roughly half the users are from city '1'.

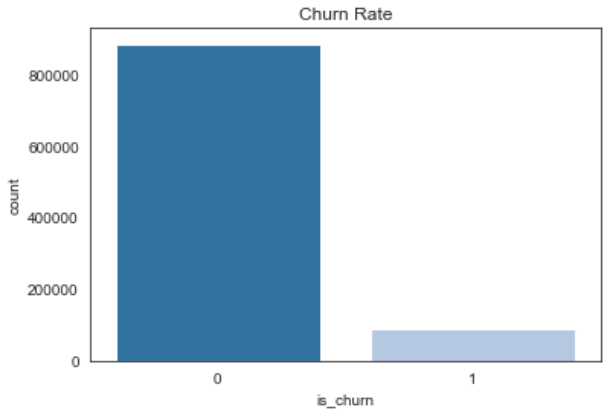

Figure 1: Distribution of Churn Behavior

Besides, out of all churn users around 39% are from city1, 14% are from city 13 and 11% of the churn users are from city5; out of all non-churn users around 59% of the people are from city1 and in the same way 10% are from city13. In terms of age, young users with age around 27 accounts for a large set, while elder users aging more than 60 only accounts for less than 5% of total population. From Figure 2, it can be concluded that customers with age around 20 are more likely to churn. Knowing age distribution in terms of churn and not churn is the first step to get a clear idea of who is our customer, which enables a better deployment. For the registration methods,

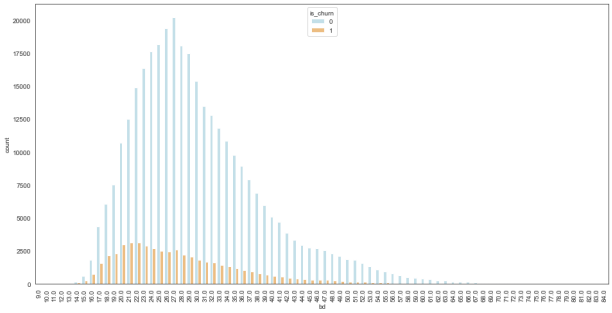

Figure 2: Distribution of Age for Churn Behavior

there are totally five different methods. A distinct churn pattern is also present in this variable, as Figure 3 shows.

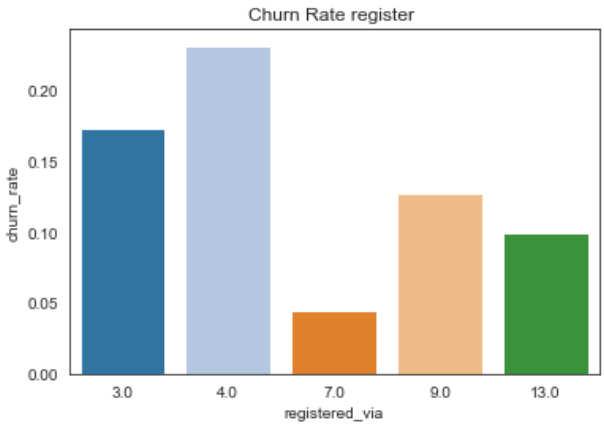

Figure 3: Churn Distribution for Register Methods

2.2.3 Transaction

The transaction dataset includes history transaction records for each user. Each user id may have multiple transaction records. For every record, it shows the information of the plan property (plan price, plan days, etc.) and transaction property (transaction date, payment method, etc.). For the payment plan days, customers with plan days 30 and 31 are less likely to churn compared with other plan days, as shown in Figure 4. We

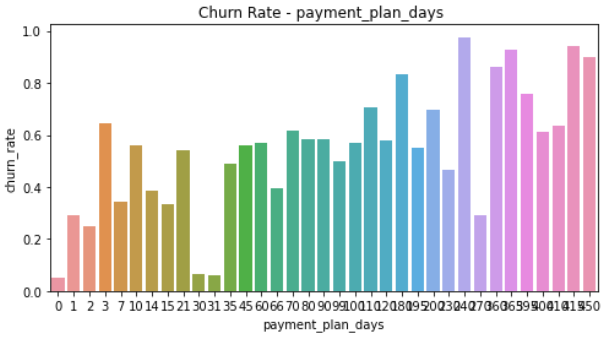

Figure 4: Churn Distribution for Plan Days

also explored the distribution for *plan_list_price* and *actual_amount_paid*. It turns out the difference between these two is not apparent. Furthermore, we found that customers who set auto renew are less likely to churn and customers who often cancel the transaction have a higher churn rate.

2.2.4 User-Logs

The user logs dataset is extremely large(30 G). It includes everyday logs activity for every customer. Specially, it listed out information like number of songs played and also total seconds listened on a daily basis. This dataset is very useful for churn prediction since it catches user behavior and reveals the customer's stickiness for the software. Even though there's a large amount of useful information encoded in these variables, few distinct patterns are discovered between customers who churn and who do not when exploratory data analysis is conducted. For further feature selection, more details are in the following sections.

2.3 Data Leakage Risk

Since we are trying to predict users' future churn inclination based on their past behavior, time order in the dataset is crucial and worth our special attention. The majority of our data including transaction and user log is time sensitive. In this case, data leakage risks from certain aspects should be considered in the first place. Firstly, since we have massive user logs across years and activity of some users can exist through past, present and future in the prediction, mistakenly including test data in the training set can be a disaster. Fortunately, this type of serious problem is also easy to be detected. Another more subtle risk is giveaway features. Including features that expose information in future behavior will cause data leakage. Finally, when we conduct pre-processing like normalizing, re-scaling and estimating missing values on training data, the test data must be untouched. Otherwise, the model will learn both the training and testing set.

Given the potential leakage risk spotted before implementation, we generate some strategies to avoid leakage problems in our project.

1. In the exploratory data analysis, we paid close attention to features highly correlated with target variables. We found a certain type of payment method has surprisingly high correlation and thus deleted this feature to prevent leakage.
2. Since we are dealing with time series data, we set a strict cut-off point on timeline to avoid miss manipulation during pre-process and modeling stages. It effectively prevents us from extracting any information after the prediction time point.
3. After training the model, we double checked if there're any suspicious features with very large weights.
4. Set the validation set to mimic the real-world scenario before we implement the model on the testing data.

3 Data Preparation

3.1 Data Pre-Processing

Before preparing the data, we need to be clear about what we indeed want to predict. Actually, we want to use the past churn to predict the future churn. Since we only have the churn labels (churn or not churn) in February 2017 and March 2017, we set the period between the past and the future as one month. Luckily, one month is a reasonable time period in most churn prediction problems. On the other hand, we care about the general patterns in users' churn. That is to say, by treating certain users' past churn information and their earlier behaviors as our train data, we not only want to make predictions on these users'

future churn information, but also want to predict other users' future churn information.

The ideal strategy is to use churn information in a certain month to train our model, validate on churn information in the next month, fit this month's churn information into our best model, and test on churn information in the month after the next month. Unluckily, as I mentioned, churn information is only available in Feb and March. Therefore, we will use the churn information in February as our training data and split the churn information in March into 30% validation and 70% testing data. To avoid using future churn to predict future churn, we will not fit the churn information in validation into our best model when making predictions on test data. In the following process of generating data, we will follow the rules I mentioned here.

3.1.1 Member, Transaction Pre-processing

For the member dataset, we found there are outliers in the variable *age*. Numbers like -999 and 150 are present in this feature column. To remove the outliers, we assign 10 to a number smaller than it and assign 85 to any number larger. Secondly, the feature *gender* has more than 60% null values, which is a clear sign of information missing. In this way we just drop this feature column. For transaction data, we filtered the customers whose membership expires between Jan.1 and Jan.31 as the users in the training data, and the one between Feb.1 and Feb.30 as users in validation.

3.1.2 User-Log Dataset Preprocessing

The user log dataset contains the information about everyday login activities for every customer. As we mentioned, we consider a user 'churn' if he or she does not make a new service subscription transaction within 30 days after the current membership expiration date. Then, for churn labels in February, users' corresponding membership expiration dates are in January, and thus we only care about their login activities before their membership expiration dates in January. Therefore, we take their all login activities before their membership expiration dates as a new dataset. Also, for the convenience of future feature engineering, we separately take their login activities 7 days, 14 days, 30 days and 60 days before their corresponding membership expiration dates as new datasets. Then, for churn labels is in March, we just repeat the same process for login activities before the corresponding membership expiration dates in February.

3.2 Imbalanced Data

Since the target variable "is_churn" is highly imbalanced(6/100), it is crucial to entail strategies like resampling or model class weight tuning to get better prediction performance on imbalanced dataset. For this project we used three different resampling techniques. The first technique is down-sampling. We took 10% samples without replacement from no churn customers and combined them with all churn customers. Up-sampling works in a similar way but increases the number of churn customers records. The third technique is SMOTE. Different from the two resampling methods mentioned above, it works by generating synthetic data for the minority class. It proceeds by joining churn customer records and its k nearest neighbors with line segments, followed by creating new data points which lie on the lines until the data is balanced(Baptiste,2019). Finally,

a balanced dataset with 1.5 million records was generated.

Additionally, besides traditional resampling techniques and generating synthetic data, different strategies are exploited to cope with imbalanced data at different stages. At the modeling stage, we tuned classifier weights to deal with the imbalanced target variable. We bring in the idea of Cost Sensitive Learning because the minority class *is_churn=1* which is underrepresented, and it increases the importances of correct prediction for positive class. Thus, we decided to use our own loss metric F_β. We employed weighted residuals and thus the minimum loss function is shifted to the minority class (is_churn = 1). Moreover, we decided to use several tree-based models including Random Forest(bagging) and XGBoost(boosting) because tree models generally perform better in processing imbalanced data. At the evaluation stage, instead of using the metric of accuracy, which will severely biased by data imbalance, we select F_β score to compare model performance and AUC to further evaluate final predicting results on testing data.

4 Feature Engineering

4.1 Members and Transactions

For the members data, we use the one hot encoding to the categorical variable *registered_via* since there are only five unique values in this column. Also, we use indicator variable *if_city_1* to indicate the feature city due to its high popularity. Besides, the feature *lifetime* is created by subtracting membership expiration date from registration date. This feature indicates the customer value to a large extent and can be used to distinguish new users from existing old users.

For the features in the transaction dataset, there are 39 unique values in terms of the categorical variable *payment_method_id*, which makes it inefficient to use one hot encoding. We binned them into three values based on its popularity, followed by a one hot encoding of this variable. *is_discount* feature is also created to indicate whether the user gets a discount in the payment. To make use of timestamp information in the dataset, all the features in the transaction dataset are redefined through a retrospective method. *Payment method id*, *payment plan days* are extracted by taking the mode of the transactions whose transaction date is between membership expiration date and prior 90 days. Besides, a new feature *amount paid per day* is created by using the plan days and amount paid information. *Transaction counts* and *transaction cancel counts* are generated on the weekly and monthly basis as well. Totally there are 14 new features generated in the transaction dataset.

4.2 User-Log Dataset

Following the pre-processing part of the user log data, besides the total history log data, different time periods version of log data are generated. More specifically, for each user whose subscription expired in the target month, his user log in previous 7, 14 , 30, 60 days are extracted and collected into separate and independent datasets. We think the combination of log data in different time periods can reflect more comprehensive information of users' behavior. In another word, both the user's service usage trend in recent days and the whole history can be represented and evaluated through features of different time periods and their interactions. More details of making

new features are described in the following paragraphs.

Primarily, some features are created within each dataset. By grouping the user logs by *user_id* and taking sums, cumulative statistics in a given period are generated like *total_seconds* and most log features. To be noticed, certain features are processed by taking means and standard deviation such as *num_unique* and *std_secs*. To some extent, they can represent diversity and variance of listening songs using KKBox in a given period. In addition to original features, some percentage features are generated based on existing frequency features. For instance, *percent_whole_song* denotes the percentage of songs listened over by users among all the songs they listened to. This percentage can be partly interpreted as the satisfaction extent of users when using KKBox's services. Likewise, the *percent_unique* is calculated when the number of unique songs divided by total listened songs and thus can demonstrate whether a user likes to play in order or put a song on repeat. Generally, features mentioned above are created in each time period version.

Besides features within each time period dataset, some interaction features across different time slots are generated. Specifically, many interaction features are calculated based on 30 days and 60 days datasets. The *listening_days_diff* and *unique_song_diff* demonstrated how users' listening habits changed in the prior month compared to the "prior prior" month. Similarly, ratio features are created to indicate the difference of percentage features mentioned above. *Percent_whole_ratio* and *Percent_len_ratio* can reflect how user's satisfaction changes from prior prior month to recent month. Additionally, a binary categorical variable called *is_new_user* is created to denote users who have activity log in recent 30 days but nothing before that. This indicator feature can help us identify new users, recognize quick churners and develop intervention strategy accordingly. Similar work is also conducted between 14 days and one month data. In summary, 42 new features are generated from the user log dataset compared to the original one.

5 Modeling and Evaluation

5.1 Evaluation Metric

Since our dataset is highly imbalanced, we will not consider evaluation metrics like accuracy and log loss. For a highly imbalanced dataset, if we predict all labels to be 0 (majority case), then our accuracy will be pretty high. However, this result is not meaningful. Also, log loss is not robust to imbalanced data. Instead, we will use F_β, a reasonable criterion for imbalanced data. The formula is:

$$F_\beta = \left(1+\beta^2\right) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

$$= \frac{(1+\beta^2) \cdot TP}{(1+\beta^2) \cdot TP + \beta^2 \cdot FN + FP}$$

We can see that F_β is a type of F score, which just adds a little modification to traditional F1 score. Thus, similar to F1 score, F_β considers both precision and recall. The difference is that it adds a weight beta to recall, meaning that we think recall is beta times as important as precision. Here, we choose a commonly used value 2 for beta since recall is more important for our churn prediction. In other words, under a reasonable precision, we want to catch the true churn as much as possible. To be more specific, the cost of acting is relatively low, maybe just a phone call or some discount. However, the opportunity cost of passing

up on a true churn is high, that is, we may lose a customer forever.

5.2 Modelling

5.2.1 Baseline: Random Forest

Our first attempt in the modelling phase is using the Random Forest algorithm to create the baseline model with default hyper-parameters, the full dataset and the full 80 features the assessment of the subsequent model performances. We choose Random Forest as the default model over logistic regression is because it is generally faster than logistic regression. As it takes random samples and random features for each tree, it could average the decisions from all the trees to form the ensemble model. Random forest algorithm also outperforms regression models in terms of complexity because it does not assume a linear relationship between covariates and the output target. We choose F_β score as our baseline evaluation metrics, and the predictions give us F_β of 0.1312. For the AUC metrics, the baseline performance is 0.6678. Therefore, we want to find more complexity models to improve the F_β and AUC scores.

Another reason we choose Random Forest is because we can extract more information from our dataset. Random Forest Classifier can calculate feature importance by ranking how well each feature performs to improve the purity of the nodes. Then we can refer back to feature importance in the process of feature selection in modeling. We first sort the features according to their importance and plot a bar graph with the horizontal axis being feature names and the vertical axis being the importance. In order to compare the features relation with the response variable, we further compute the Mutual Information to understand the dependency between the covariates and the response variable.

5.2.2 Feature Selection

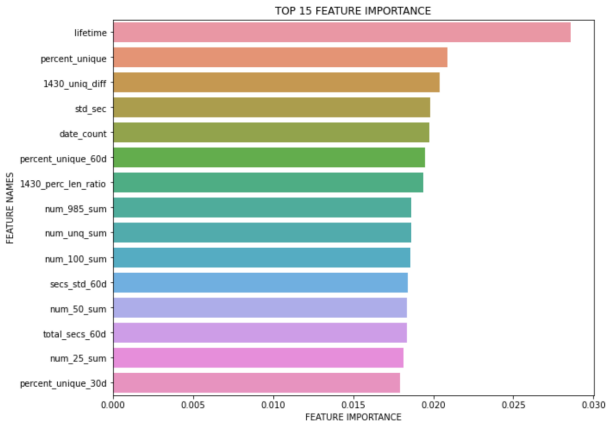

Figure 5: Top15 Important Features in Baseline Model

We selected our subset of features based on the combined results from the tree-model feature importance and mutual information. We decided not to apply PCA to select features due to its uninterpretable variables in nature. Furthermore, the number of features is too big to conduct step-wise selection efficiently. Since we build our baseline model of random forest, both of these two ranks can be efficiently obtained. However, there are still limitations. Mutual information rank can indicate how much each feature correlates with the target variable, but it cannot deal with collinearity problems. In other words, if two features are both highly correlated with target and also highly correlated with each other, they will be both ranked high by MI rank. But in fact, one of them can provide little additional information about the target on the foundation of another. Unlike mutual information, tree-model feature importance may rank two correlated features very low when they are actually good predictors. Therefore, we decide to drop those features not in the top 30 of both ranks. In summary, there are 29 features

considered less important and thus our selected subset data contains 51 favored features.

5.2.3 Logistic Regression

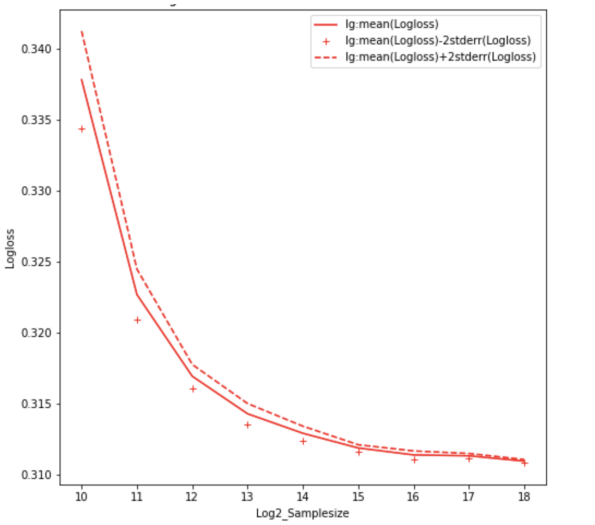

Figure 6: Logistic Regression Learning Curve

Logistic Regression is a simple but fast classification algorithm, and its performance is robust when the sample size is small. For the learning curve here, it is enough to use log loss as a criteria here since we just want to know how much large the sample size is sufficient for our model to learn some patterns. The learning curve also shows that logistic regression is able to learn enough information with the sample size 65000. However, since logistic regression is basically a linear regression on log odds, we need to consider the collinearity between features. Therefore, we generate a new dataset by removing highly correlated features and train our model on both the dataset with all features and the new dataset. For the hyperparameter, we consider the regulation term: lasso and ridge, and the Inverse of regularization strength C: from 10^{-5} to 10^{5}. Because our dataset is imbalanced, we also take the class weight into consideration. Higher weight on a class means a larger penalty for making a wrong prediction for that class. After validating our model, we find that the best hyperparameter is l2 norm, C = 0.001, and class weight = $\{0 : 0.1, 1 : 0.9\}$ for both of the models for two datasets. In addition, their F_β scores are both around 0.36. Thus, for the convenience of further implementation, we choose the simpler model, that is, the model on the dataset with highly correlated features removed. It is common that F_β is not ideal here, because logistic regression is a linear model and cannot detect the nonlinear pattern in the data.

5.2.4 Gradient Boost Decision Tree (GBDT)

We implemented the GBDT model after consideration for several reasons. 1). The tree-based model has an advantage in dealing with imbalanced data. 2). Our baseline model random forest is a form of bagging, so we think it would be meaningful to try boosting models. Unlike the random forest which builds trees in parallel, GBDT builds trees sequentially. GBDT has obvious limitations because it's very time consuming to train and more vulnerable to overfit. Thus, we try to alleviate these problems during the tuning process. The first step is tuning *n_estimator* because it's important to avoid overfitting and generally more independent with other hyperparameters. After learning hyperparamers, we tuned tree-related ones such as *min_sample_split* and *min_sample_leaf*. The resulted best possible parameter combination of grid search is: *max_depth = 9, n_estimators = 100, min_sample_split= 700 ,min_sample_leaf= 80* and the highest F_β score = for GBDT model is 0.2952.

5.2.5 XGBoost

Due to the slow training speed of GBDT model,we have moved to implement XGBoost algorithm (2) which is a form of high performance and high-speed implementation of GBDT algorithm. The reason that the GBDT algorithm is slow is because it trains sequential decision trees, where one tree is trained depending on the residual from the previous tree, and only one tree is trained at a time. This restricts the speed of computation of GBDT since we cannot train multiple trees simultaneously, whereas XGBoost achieves parallel computing at the tree-building stage. XGBoost groups features according to available CPU cores, where each core uses the assigned features to do calculations in node splitting. This is especially relevant to our project since we have a large number of features.

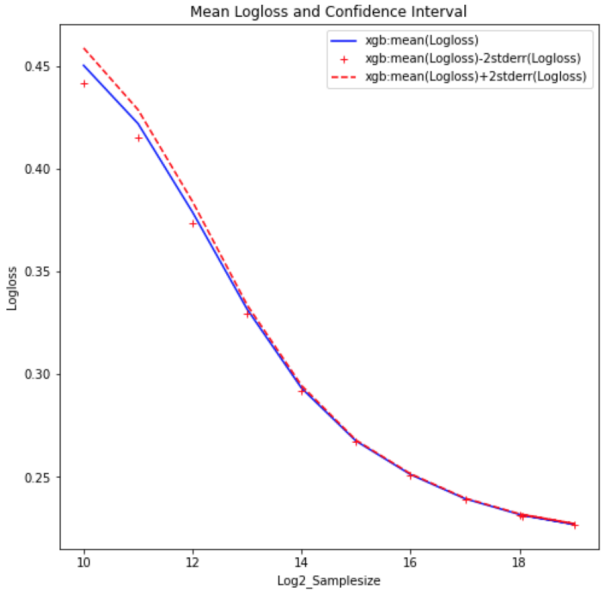

Figure 7: XGBoost Learning Curve

From the learning curve, we can see that 26000 is a reasonable sample size here. In order to tune the hyper-parameters of XGBoost, we implemented Grid Search to find the optimal-possible combination of the hyper-parameters. We have tuned *n_estimators, max_depth, scale_pos_weight, learning_rate* and etc using GridSearchCV package. We assign the training dataset with index -1 and the validation dataset with index 0 using *PredefinedSplit* so as to use our own validation set in the GridSearchCV. To maintain a relative complex model but avoid overfitting, we have found *n_estimator=800* and *learning_rate=0.001* is the balance point. The best performed F_β score obtained is 0.4547

5.2.6 LightGBM

There are two main reasons why we included Light GBM in our project.It uses histogram-based algorithms, which bucket continuous feature values into discrete bins. Most of the features extracted from the user log file (like number of unique songs listened daily) are continuous and wide-ranged. By binning these values, the cost of calculating the gain for each split as well as the memory usage are greatly reduced(Pushkar,2019). Secondly, Light GBM uses leaf wise splitting over depth wise splitting, which enables it to converge much faster than XGBoost. Given the extremely large size of the training data, it is more preferable to apply a model that could fit the data in a timely manner. In terms of tuning, parameters like "objective" and "metric" are set first. Then totally 12 other parameters each has on average 10 values are fined tuned using randomized search. The best performance in Light GBM reached F_β score 0.4333.

5.3 Model Result

Among the models we implemented previously, the XGBoost performs most outstanding with F_β score of 0.4547. After selecting best model based on validation data, we

Classification Models	*F-beta Score*
Random Forest	0.1312
Logistic Regression	0.3606
GBDT	0.2952
LightGBM	0.4333
XGBoost	**0.4547**

implemented it on test data. In result, the best model performed relatively stable regarding performances before. In testing, we obtained F_β score of 0.4395. Correspondingly, AUC score of 0.7560 further verified solid performance on testing data.

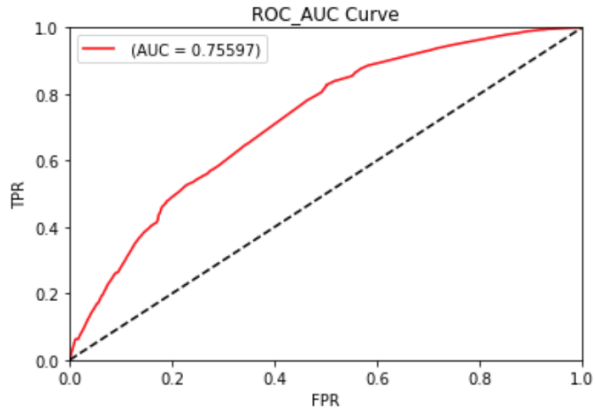

Figure 8: ROC_AUC Curve

6 Model Deployment

6.1 Deployment Structure

Before deploying the model, we want to emphasize again that our model split the churn data in March into a validation set and a test set due to the limitation of data source. The best way is to use churn information in a certain month to train our model, validate on churn information in the next month, fit this month's churn information into our best model, and test on churn information in the month after the next month. Then, the deployment process is shown as below.

User login Activities, user transaction and user information are all raw data collected by KKBox in various ways, and they are stored in database.

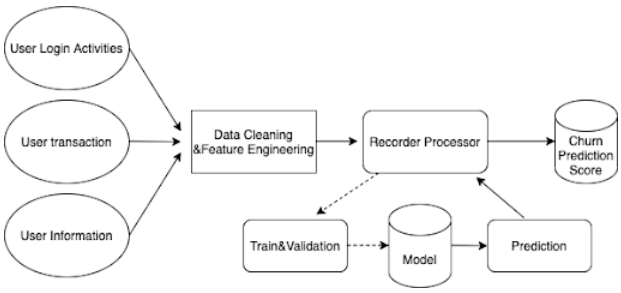

Figure 9: Deployment Process Flowchart

Data Cleaning & Feature Engineering is the code that is used to clean the data and generate the features.

Recorder Processor is used to build, record, and apply the model. It takes the train and validation data to fit the model, and then stores the model hyper-parameter into Model Database. After that, it uses the model to make predictions for the next month. We should notice that we do not need to update the model every time we get new data. Instead, we only need to update the model every three months, half year, or other time period based on the reality.

Churn prediction Score Database stores the prediction for the probability of churn from the Recorder Processor.

6.2 Deployment Problems

During the process of deployment, the company needs to be careful with several things. First of all, when we retrain the model, we can fit all historical data if data size is acceptable. Nevertheless, with time passing, the size of all historical data will become extremely large. Under this circumstance, it will take too much time and space for the company to feed all the data to train the model. One solution is to extract the most recent, reasonably sized data to train our model. Secondly, after taking actions on the potential churn, the company is supposed to check

the result regularly. For example, we may improve the churn rate by reaching out early to the customer at the risk of churn or by providing them discounts. However, when we wrongly reach out to people who do not plan to churn, some of them may get annoyed and complain about the phone call or email. Also, providing discounts to too many people may hurt our profit. In one way, we can measure the risk by a lift and recall plot to decide how much percent of potential churn we should intervene. In another way, we ought to consider adjusting the weight for recall in our evaluation metric F_β. However, the excessive emphasis on may penalize our result. Last but not least, we should not make strategies merely based on some features that we consider most influential. For our experiment, we just show these features are related to churn, but we haven't proved that there exists a causal effect. To prove the causal effect, more A/B testings need to be done.

6.3 Ethical Risk

Our model did not use features like race and gender to make predictions, so we avoided the potential ethical risk caused by using sensitive features.However, the firm should still be careful when they reach out to the users at the risk of potential churn. If the company mentions the potential churn directly or uses some terms related to that, some of those users may feel offended. These users will think that they are spied by the company once they know their activities and personal information are tracked. Hence, when calling or emailing to the users, the company should pay attention to the wording and perform in a more concealed way.

7 Conclusion & Future Work

In our study, we tried to build a reliable model to help KKBox predict potential churn. Around 80 features were generated to help us make a credible prediction. For the highly imbalanced data, we have tried down sampling and up sampling. We used F_β score, where β is 2, as our evaluation metric to deal with the imbalance and to pay more attention to the recall. For model selection, Random Forest is our baseline. We also tried other classification algorithms like Logistic Regression, Gradient Boost Decision Tree, XGBoost, and LightGBM, among which the XGBoost gave the best performance. Finally we designed a deployment process, including considerations related to deployment problems.

For future work, we need to acquire new churn data for the month after the month currently containing test churn data, so that we can split the train, validation and test perfectly by one month interval, and the experiment will become more convincing. Besides, more information about the users are needed since some information may be more influential than information we have at this stage. Moreover, We will also try Neural Networks on our data because they have many free parameters ,and this gives them the flexibility to fit highly complex data. Finally, we plan to do some A/B testing to investigate whether there exists some causal effects between certain features and churn.

References

[1] *Altexsoft. "Customer Churn Prediction Using Machine Learning: Main Approaches and Models." KDnuggets, 2019, www.kdnuggets.com/2019/05/churn-prediction-machine-learning.html.*

[2] *Ahmad, A.K., Jafar, A. & Aljoumaa, K. Customer churn prediction in telecom using machine learning in big data platform. J Big Data, 2019. https://doi.org/10.1186/s40537-019-0191-6.*

[3] *Brisco,K. The Importance of Predicting Customer Churn. Analytics | NICE Knowledge Base,2019, https://www.nice.com/engage/blog/nexidia-the-importance-of-predicting-customer-churn-2499/*

[4] *Rocca,B. Handling imbalanced datasets in machine learning. Towards Data Science, 2019, https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28*

[5] *Mandot,P. What is LightGBM, How to implement it?How to fine tune the parameters? Medium, 2017, https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc*

Appendix

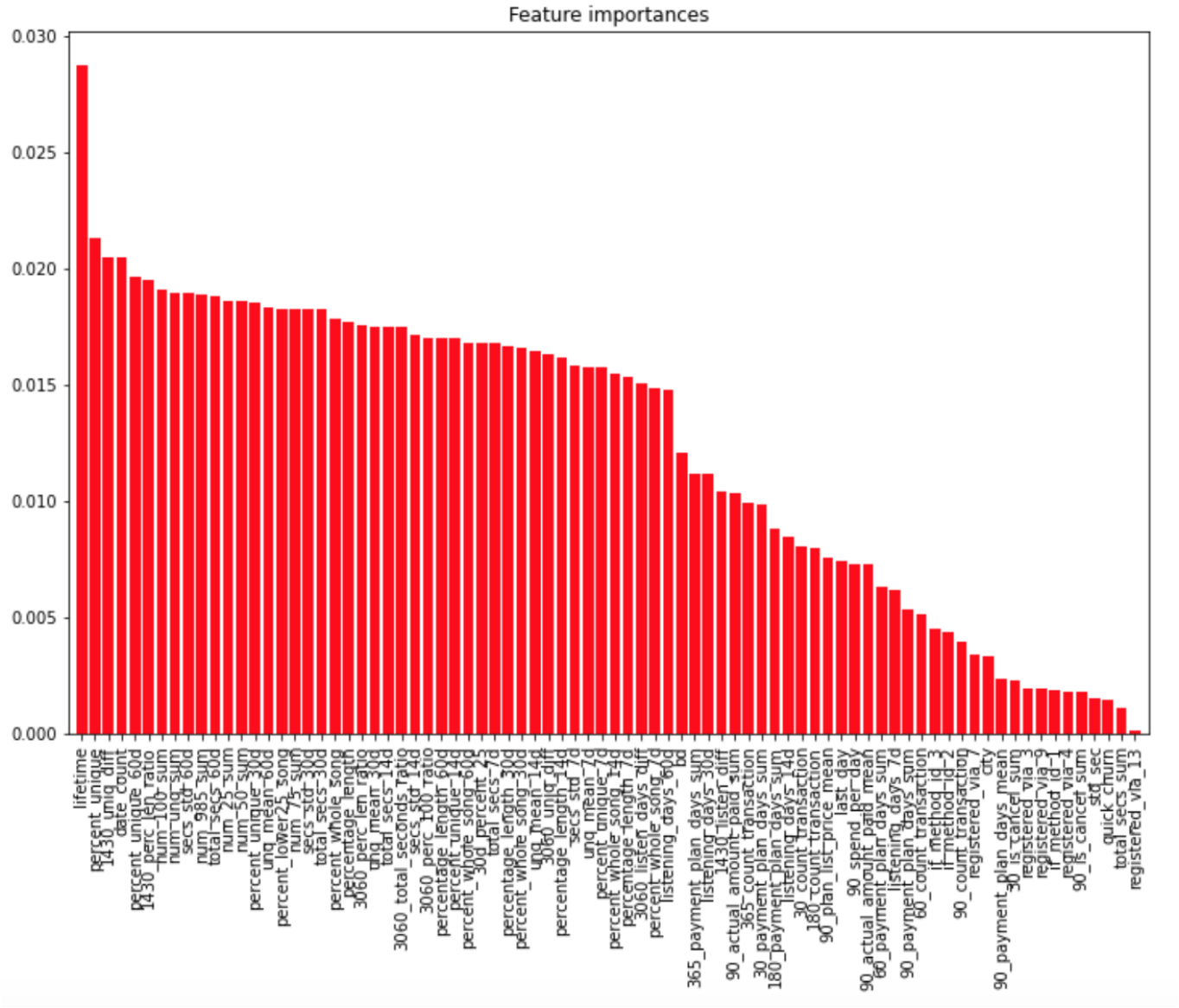

Figure 10: Feature Importance for Baseline Model

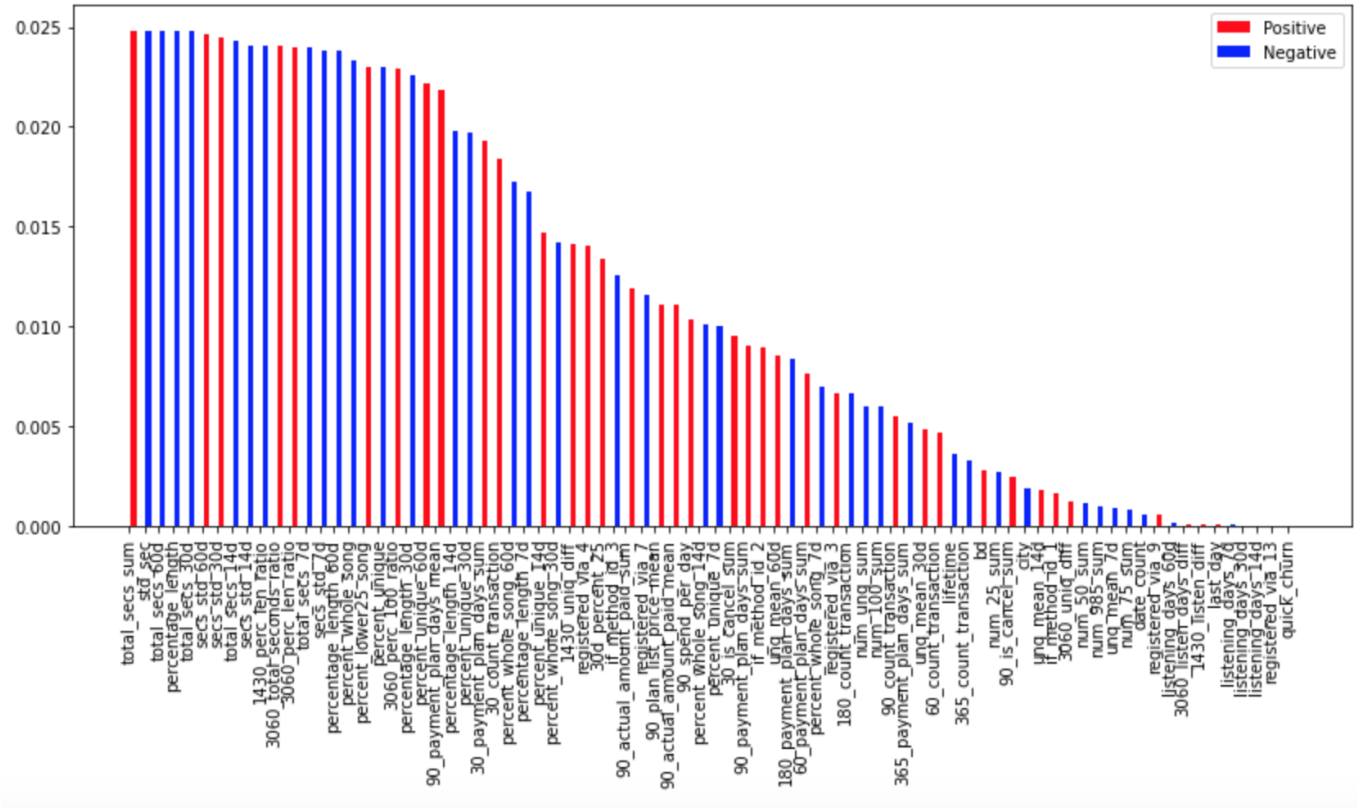

Figure 11: Mutual Information