THE UNIVERSITY OF TEXAS AT AUSTIN

MASTER OF SCIENCE IN BUSINESS ANALYTICS

ADVANCED PREDICTIVE MODELING

KKBox Churn Prediction Challenge

Authors

Corey HAINES, Reece WOOTEN, Anurag AGARWAL, Deeksha YENNAM, Shreya TRIVEDI

December 14, 2017

Abstract

Churn is a common problem to many industries. The inability to retain customers can have a drastic impact on profits. This is especially true for streaming services, who rely on subscription renewals and loyal customers. KKBOX, an Asian-based music streaming service, is one such company. Due to their natural interest in churn prediction, KKBOX's is hosting a Kaggle competition where the goal is to predict the probability of a customer churning given users' demographic information, their transaction history, as well as their session logs. Given both the heavy predictive modeling component as well as business applicability of the competition, our team thought this would be an apt final project. While initial exploration of the data did reveal some features contained pertinent information for churn prediction, many of the base features were noisy with respect to whether the user churned. Therefore, a large amount of time was spent on feature engineering, specifically with the transactions data and user session logs. Both of these data sets were rolled up into sets of representative aggregate statistics. For each engineered feature, the weight of evidence was calculated in order to provide an indication for how much information value our engineered features contribute to churn prediction. A logistic regression classifier was the first model fit to the data, but poor performance forced us to move on to more sophisticated methods. These included Kaplan-Meier Estimator, Nelson-Aalen estimator, Cox Proportional Hazards Model, Random Forest, and XGBoost. Out of all methods used, XGBoost performed the best in terms of negative log loss. The top features were examined from this model in order to inform us of potential plans of action KKBOX may want to take in the future to enhance customer retention.

Contents

1 Introduction

Based out of Taiwan, KKBOX is a cloud-based music streaming service that serves over 10 million music listeners in the Asian market, most notably in Taiwan, Hong Kong, Malaysia, Singapore, and other countries with Mandarin-speaking citizens. With the support from more than 500 international major and local independent music labels, KKBOX features 40 million, including the most comprehensive Asia-Pop music library. Like Spotify, KKBox operates on a monthly subscription model, meaning the majority of their subscription lengths are 30 days. Within their subscription model, they offer a tiered pricing structure in which their base services are free but customers must pay for features such as offline mode and no advertisements.

Under KKBox's current membership structure, users can choose to renew or cancel the service when their membership is about to expire. They also have the option to auto-renew but can still cancel their membership at any point in time. As Asia's leading music streaming service, KKBOX has seen a steady increase in the number of knew subscriptions each month as well as having many users re-subscribe at the end of every month. However, some customers are inevitably lost due to churn. Churn is the number of customers or subscribers who cut ties with your service or company during a particular period. Usually this number is divided by the number of customers at the beginning of the same period to express the churn rate over some duration of time. Knowing this metric is key for subscription-based companies, as even a slight fluctuation in churn rate can have a large effect on profits. Furthermore, knowing what features affect the likelihood of churn within a company provides insight into why customer choose to leave. This in turn can inform long term retention initiatives such as flagging at-risk customers with email campaigns or targeted promotional discounting.

Currently KKBOX uses survival analysis techniques to determine the residual membership life time for each subscriber. Some key fields in their current models for determining churn/renewal are the transaction date for the user's subscription, the user's membership expiration date, and whether the user has canceled a past membership. Note, even if a user has actively canceled a subscription, the cancellation does not imply the user has churned. A user may cancel their current service subscription due to change of service plans or other reasons. KKBOX's criteria of "churn" is no new valid service subscription within 30 days after the user's current membership expires.

As part of the 11th ACM International Conference on Web Search and Data Mining (WSDM

2018), KKBOX has donated a dataset for a Kaggle competition that challenges contestants to build an algorithm that predicts whether a subscription user will churn given the features provided. Specifically, the goal is to predict if a user makes a new service subscription transaction within 30 days after their current membership expiration date. To do this, contestants develop binary classifiers that assign churn probabilities to each user and compare that with whether or not the user actually terminated their subscription with KKBOX. The goal is to create a predictive classification model that minimizes the negative LogLoss of the data and hopefully provide new insight to KKBOX that helps retain customers at a higher rate in the future. Given the importance of churn to business profitability [and the chance at winning $5000], our group decided to try and tackle this binary classification problem. Before diving into the competition, it was important to understand what factors drive churn within the streaming service industry and see what type of predictive modeling work has already been explored with regards to churn prediction. Therefore, we looked for primary literature that would guide us in our strategy for developing our classifier. Since KKBOX currently employs survival analysis techniques, we first sought out literature on this class of models in the context of churn prediction.

In the telecommunications industry, Lu (2002) found that if the shape of the survival distribution and hazard function is known beforehand, or can be reasonably estimated, then modeling log of censored survival times against a range of features [i.e., $log(T_i) = \beta_0 + \beta_1 z_{i1} + ... + \beta_p z_{ip} + \sigma_i \epsilon_i$] performed better than predicting survival probabilities using the Cox proportional hazards model[1] . More advanced, non-parametric techniques have also been applied to churn prediction problems. Coussement and Poel (2008) assayed the effectiveness of support vector machine (SVM) classifiers for churn prediction in subscription services. They found SVMs to show good generalization performance but were sensitive to the parameter optimization procedure.[2] In 2009 Xie et al. proposed an improved balanced random forests method as a means to achieve high AUC when facing class imbalance with churn prediction problems[3] . Others have attempted to use various forms of artificial neural networks (ANN), such as hybrid ANN+ANN[4] and recurrent neural networks[5] to improve prediction outcomes. Given the wide range of methods applied to churn prediction in different sectors, we decided it would be best to apply several of the aforementioned methods to our dataset in order to see which method handled the intricacies of the KKBOX dataset the best.

2 Data

2.1 Description

The data KKBOX provided was separated into five main files: members, transactions, user logs, train, and test. The members data contains demographic information, the transactions file contains information about the users' payment plans, the user logs file contains information about how the users streamed music on KKBOX, and the train data contains whether or not the customer churned. The time frame over which these files span is January 1st, 2015 to March 31st, 2017. The train and the test data are selected from users whose membership expire within a certain month, namely February 2017 for the train data and March 2017 for the test data. This means we are looking at user churn or renewal roughly in the month of March 2017 for train set, and the user churn or renewal roughly in the month of April 2017. This encapsulates 992,932 users in the training set and 970,961 users in the test set respectively.

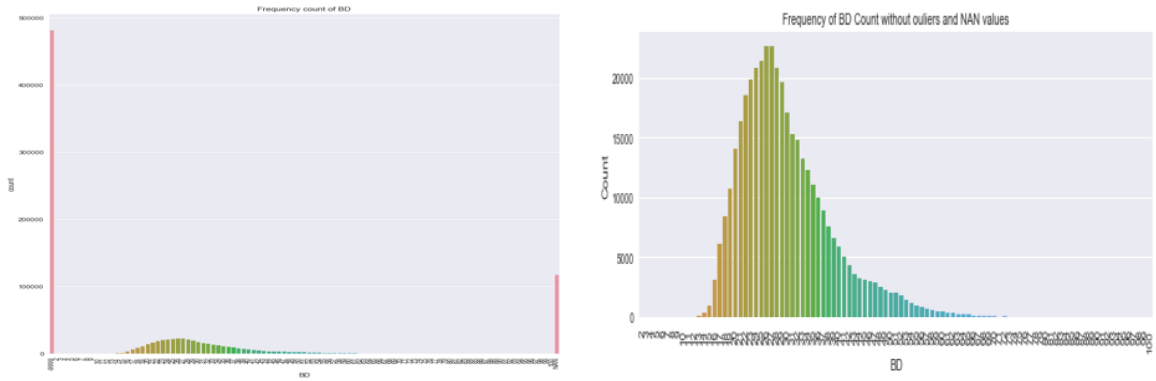
The membership data file contains information for 5,116,194 users. The fields contained within this file are the member ID (primary key) as well as each users' city, age, gender, the registration method, initial time of registration, and membership expiration date. The transactions table contains 21,547,746 total transactions covering 2,363,626 unique members. Within this data file are the member ID (foreign key), transaction date (composite primary key with member ID), membership expiration date, payment method, length of membership plan in days, the plan list price in New Taiwan Dollars (NTD), the actual amount paid in NTD, whether the user auto renews, and whether the user actively cancels their subscription. Lastly, the user logs table contains information on 392,106,543 sessions covering 5,234,111 unique users. For each user session the member ID (foreign key), date of session (composite primary key with member ID), number of songs played less than 25% of the song length, number of songs played between 25% to 50% of the song length, number of songs played between 50% to 75% of the song length, number of songs played between 75% to 98.5% of the song length, number of songs played over 98.5% of the song length, number of unique songs played, and total seconds played are logged.

2.2 Pre-Processing & Outliers

There were three main pre-processing steps involved with handling the KKBox churn data: handling the outliers, standardization, and one hot coding/dummy variables. The data were standardized by subtracting the mean and dividing by the standard deviation of the various attributes.

Since some of the models we attempted are influenced by the magnitude of the variables, this will allow for more robust churn predictions. Since the goal of the competition is simply to predict churn and not gain any inference from the attributes, there is no need to preserve the original units the variables.

In general, the datasets were fairly clean. However, there were some features that needed a bit of cleaning up. Namely, the features age, gender, city, registration method, payment method, and total secs needed processing before being suitable for modeling. As is common with many self-reported personal information, people's self-reported ages were questionable in most cases. Values for age ranged from a very youthful -6998 to an extremely old 2015. Furthermore, a preponderance (3,421,570) of ages were listed as 0. Therefore, we made the decision to limit the age range to 1-100 (N.B., the lowest reasonable age listed was 21). While there could be some centennials jamming out to Asian pop music, we figured this unlikely. All ages outside of the set range were set to -99999. This was, if there was a correlation between poor age input and churn rate we would be able to capture that information.



(a) Age (b) Age w/o outliers

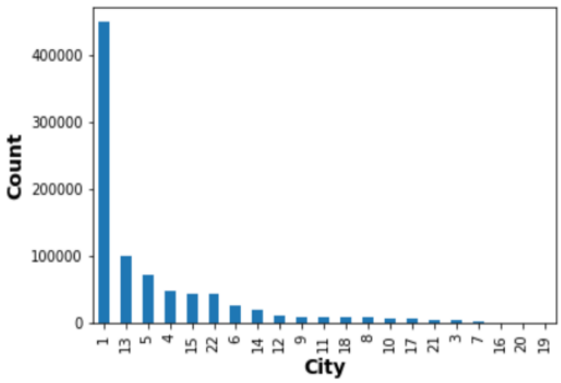
Figure 1: Age & Age w/o outliers

Gender was also problematic in the sense that 3,354,778 entries (65.572%) were blank. Since there could be a correlation between gender and churn rate, we replaced the NaNs with "unknown" instead of dropping the column altogether.

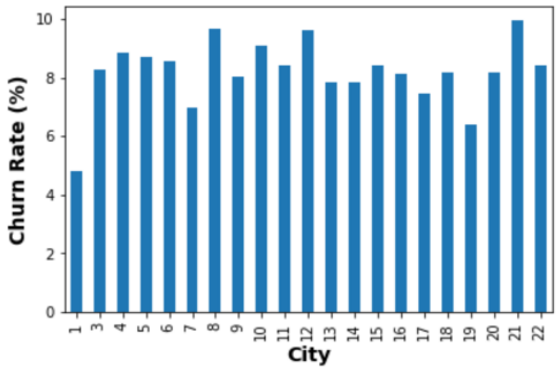
The city, registration method, and payment method were all encoded as integer values ranging from 1-22, 4-13, and 1-41 respectively. For modeling purposes, these variables were casted as categorical and one hot encoded. Lastly, some of the total secs values observed in the user logs were unfeasible. The minimum amount of time [in seconds] spent listening to music for a user

session, which one would expect to be 0, was -9,223,372,036,854,775,808 while the maximum time was 9,223,372,036,854,775,807 (i.e., the minimum and maximum values that can be stored within an int64). While these may represent some internal designation by KKBOX, these values had to be adjusted before feature engineering could take place. For negative values, we made the assumption that these numbers signified that the user logged on but did not listen to anything. Perhaps they were just exploring new artists or seeing what the top hits were for the week. Therefore, any negative values were set to zero. While there are very avid music listeners, the maximum value for listening time was unrealistic. Therefore, we set the maximum amount of time for a user session at 604,800 seconds (a week). While still extremely unlikely, it is possible that someone logged in and then streamed music on their device and left it active accidentally.

For each set of datafiles we wanted to understand the spread and distribution of each feature, how features interacted with each other, as well as how the features related to churn. We first explored patterns in the members dataset. Interestingly, many users are from the same city, (city 1) and that city has the lowest churn rate. Therefore, this may prove a useful predictor for the models.

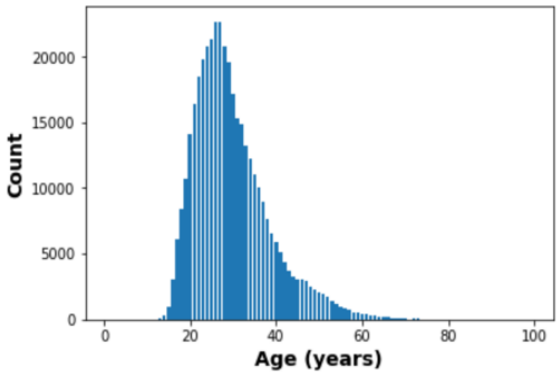


(a)

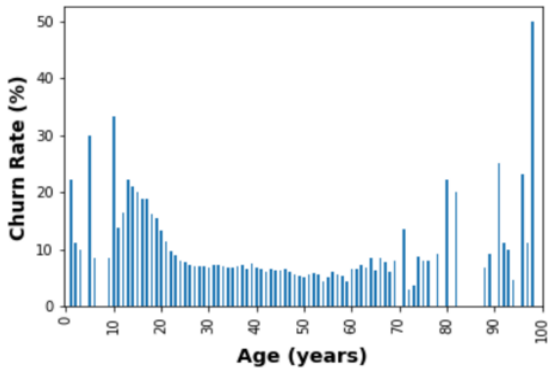


(b)

As shown earlier, when restricting oneself to a realistic age range it appears that age has a log-normal distribution. When examining what ages are most associated with churn one can see that both younger and very old people have higher churn rates. This could be because younger customers can no longer afford the subscription fee and older people don't find utility in the subscription.

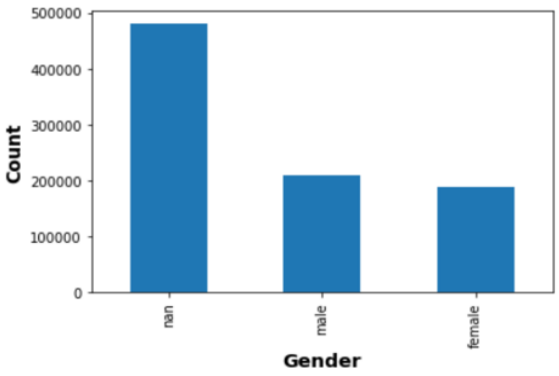


(a)

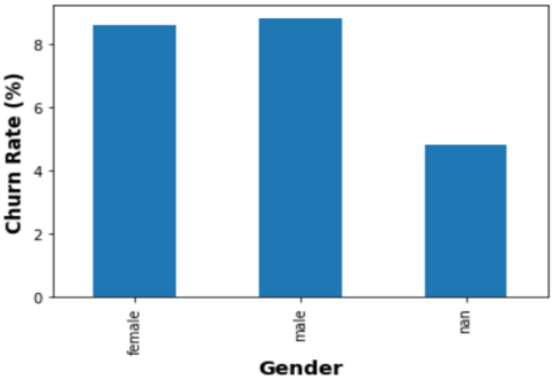


(b)

There is roughly an even ratio of males to females and they have similar churn rates. Interestingly, the unknown class has a lower churn rate. Therefore, it would be useful to keep this a feature for the models.

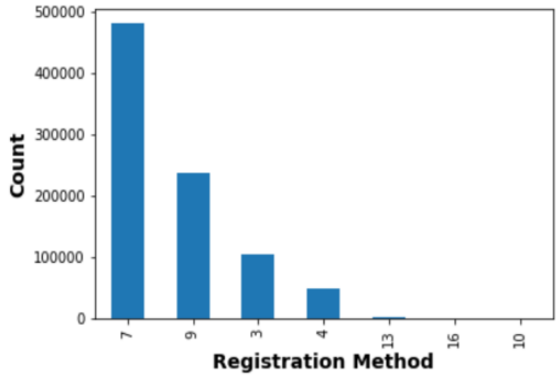


(a)

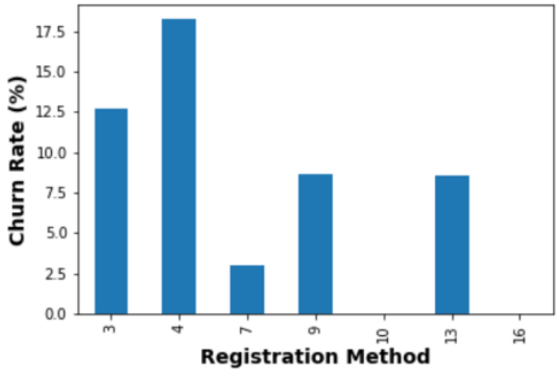


(b)

When looking at the manner in which users registered for KKBOX, most users registered with methods 7, 9, or 3. Like city, the most popular method also had the lowest churn rate. Therefore, whether a user registered with method 7 or not should be a good predictor for churn.

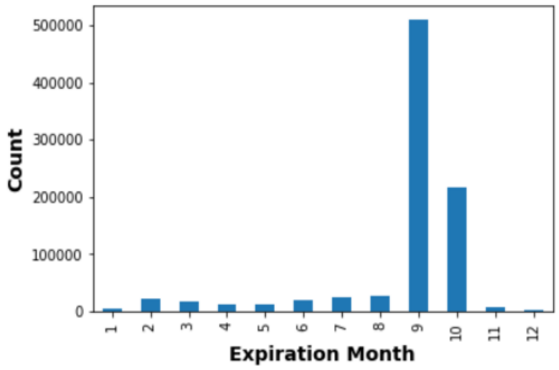


(a)

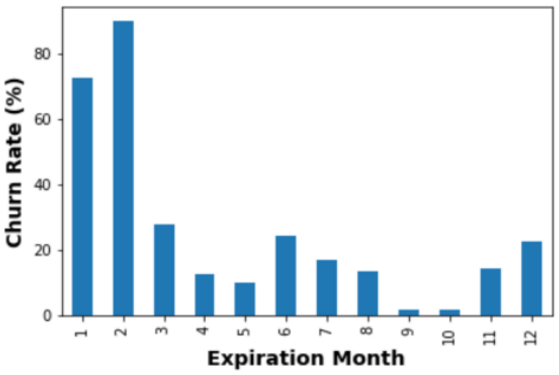


(b)

Most of the registration and expiration date information did not contain any suggestive information about whether a user will churn or not. However, the expiration month did seem to contain some information, as subscriptions expiring in January and February had much higher churn rates than other months

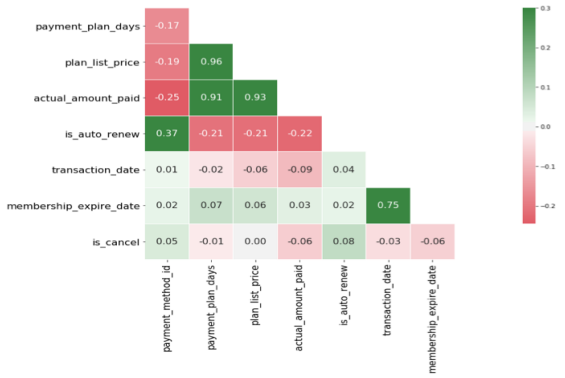


(a)

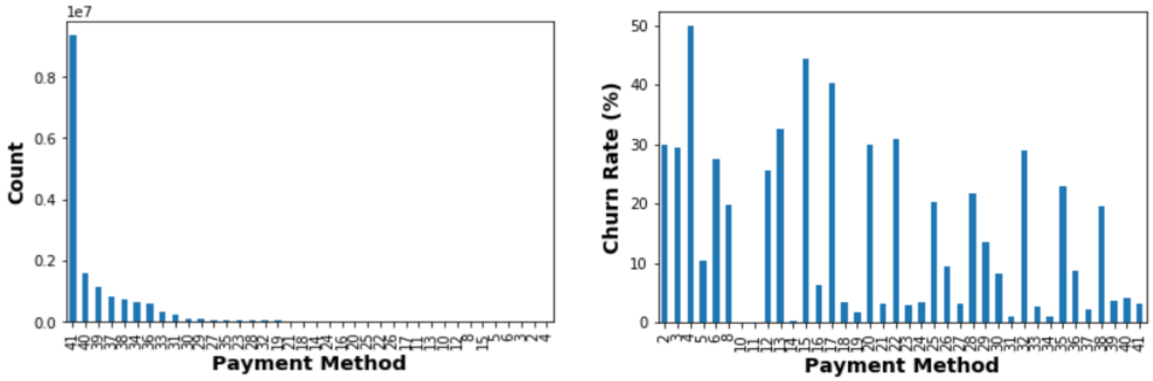


(b)

Since the transactions data file had much more quantitative data stored in it, we first wanted to see if there was any multicollinearity between the features by constructing a correlation matrix between the features. As can be seen below, the payment plan length is highly correlated with payment plan price and the actual amount paid, which are in turn almost perfectly correlated. This is not surprising as the subscription price is based off the amount of time a user subscribes for KKBOX's services. However, whether or not user has auto-renew setup is negatively correlated with the three aforementioned variables. This could be because the longer you subscribe for the less likely you need to auto-renew your subscription. All you have to do is remember perhaps once a year.

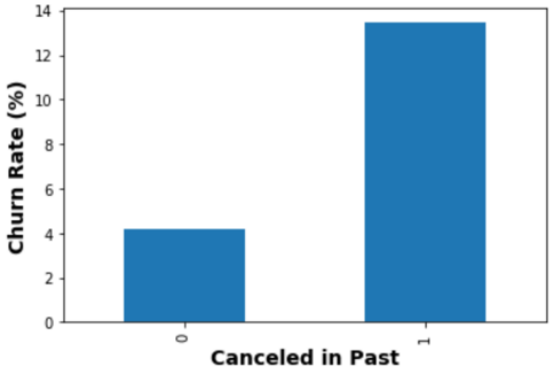


Because most people pay for things with their credit card, the method of payment feature was dominated by one method. Since this method was so preponderant, we naturally examined what users' churn rates were for this payment class. We found that those that pay with this method have a significantly lower churn rate than other methods. Furthermore, there are some methods that have extremely high churn rates. We thus posited that this feature would be a very good predictor for churn



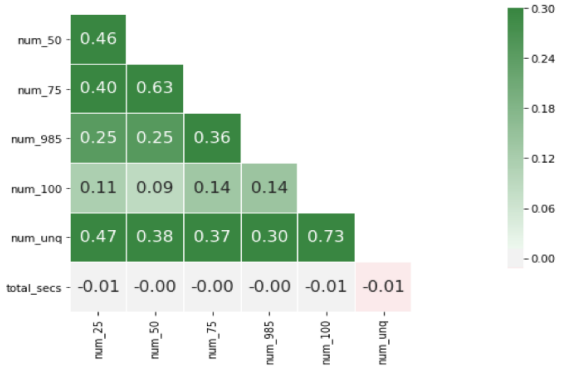
(a) (b)

Another obvious candidate feature to be a great predictor of churn was whether or not the user had canceled their subscription in the past (is_cancel). As seen below, this is in fact the case.

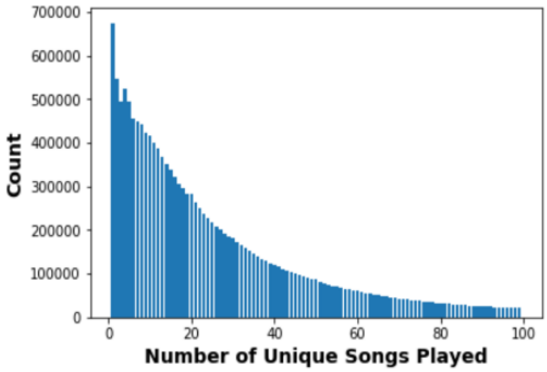


Other features, such as payment plan length and plan price, within this data file were very noisy with respect to churn rate. However, they were kept in the dataset because more sophisticated methods like ensembles may be able to capture the information inherent within them.

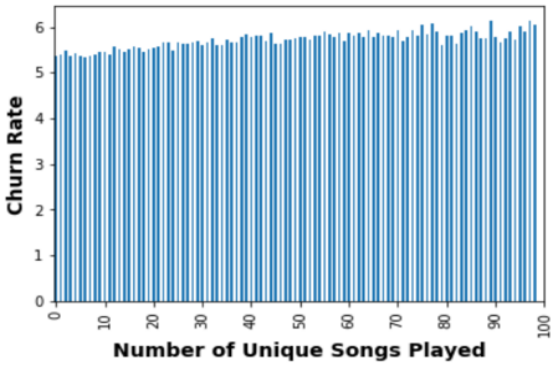
Lastly, we investigated the inherent distributions of the user log data as well as how it was associated with churn rate. However, because many of these features are highly related, we first checked constructed a correlation matrix to determine whether some variables could be dropped due to lack of new information.



The correlation matrix above confirms our suspicions. Essentially none of the variables are negatively correlated with one another. Rather, all of the listening features [except total listening time] have a moderate to strong correlation with each other. Interestingly, it appears that both those that don't listen to much of a song and those that listen through the entire song listen to a greater number of unique songs during each session. Despite the conventional wisdom that the most predictive information on whether a customer will churn reflected in how they use a service, this was not the case on the surface of the user logs data. Rather, most of the variables were strongly right skewed and had a uniform distribution for churn rate across their values. An example of this is shown below by the number of unique songs played during a user session.



(a)



(b)

This indicated to us that a significant amount of feature engineering would have to be performed in order to extract the relevant information from the user logs.

2.3 Sampling

Call	# Rows	%
Not Churn	81,8121	94%
Churn	51,696	6%
Total	869,817	100%

We recognized two potential problems in continuing with this dataset for building our models. First, the size of the dataset is fairly large when compared with the computational capabilities of our machines, and could mean long training time for the models along with an added risk of crashing of the machine. Second, given that positive class makes up only 6% of the data, there is huge class imbalance. Any model that we train will not see enough samples of the positive class and might result in poor performance.

We performed stratified sampling in order to solve for both of these issues. Below is the distribution of positive and negative classes in the sampled dataset

Call	# Rows	%
Not Churn	4000	80%
Churn	1000	20%
Total	5000	100%

This dataset was then broken down into Training and Validation datasets using 80-20 stratified split. Training dataset contained 4000 rows (3200 negative and 800 positive), and validation dataset contained 1000 rows (800 negative and 200 positive).

In this process, we have artificially changed churn rate from 6% to 20%. Therefore, we calculated class weights for both classes.

$$Weight\ for\ class_i = \frac{Number\ of\ rows\ in\ class_i\ in\ train}{Number\ of\ rows\ in\ class_i\ in\ original}$$

The class weights were provided as a parameter during model training. This approach ensured that the modeling algorithm saw enough rows for both classes while also maintaining the original positive class ratio of 6%. We do not expect the ROC curves to be affected due to this calibration, but we believe this will still affect the log loss metric.[6] For the purposes of this paper, most of the models were trained on the unbiased data without the sampling. This was to preserve the calibration of the probabilities for the KKBox competition, we did however run a XGBoost with the sampled data to see if we could improve some of the classification accuracy metrics.

3 Feature Engineering

Our data set had details of each transaction made by a single user and hence each user had multiple transaction rows. The challenge here was to combine these transaction rows into a single row of transaction for a user capturing all the relevant information for that user. We created a roll-up function in our code which takes multiple transaction of a single user id as input and gives out a single row of transaction. We created several new features within our roll-up function and each feature used a different logic to merge the transaction data.

1: Frequency of Transactions: Total number of transactions of a user

2: Total number of payment methods: Sum of different types of method used by a user in all the transaction

3: Most common method of payment: Payment method which had the maximum number of transactions for a user

4: Cancellation rate: Fraction of total number transactions where a plan was canceled after

subscription.

$$Cancellation\ rate = \frac{(Sum\ of\ transactions\ with\ is\_cancel = 1)}{Total\ number\ of\ transactions}$$

5: Discount availed: There were multiple transaction in our data where actual price paid was less than the list price of plan. This could be due the some kind of discount availed by the user. To capture the information related to discounts availed by the user, we created two new features.

   – Percentage of transactions where amount paid was less than the list price of plan
   – Whether or not a discount was availed in last subscription plan

6: Amount paid in last subscription: We hypothesized that the last transaction of the user will tell us a lot about his/ her potential churn. This feature took the amount paid in the most recent transaction of the user

7: Plan taken in last subscription: Counts the number of days in the last plan taken

8: Checking the trend in type of plan taken by the user: To see if there was a upgrade or downgrade in the last plan. To capture this, we created 2 features

   – Amount paid by user in their last transaction - Average amount paid
   – Number of days in last plan taken - Number of days in average plan

9: Auto renewal of the last plan: This feature checks whether or not auto renewal option was used in the most recent plan of the user

10: Number of days spent on KKBox: This feature gives the age of the customer on KKBox as we assumed that number of days spent till date could be a very important predictor of churn of a user. To calculate this: $Current\ date\ -\ Registration\ date\ of\ the\ user$

Below is the list of all the features created:

Table 1: Variables 1-10

Engineered Features	Description
V1_NumTransactions	Number of Transactions by a user
V2_NumUniquePmtMethods	Number of different payment methods used by a user
V3_MostFrequentPmtMethod	Most common payment method of a user
V4_PctTxCancel	Percentage of transactions where is_cancel was 1 for a user
V5_PctTxDiscountAvailed	Percentage of transactions where a discount was given to a user
V10_IsDiscountLastTx	Whether last subscribed plan of user had a discount

Table 2: Variables 7-13

Engineered Features	Description
V7_PlanAmountLastTx	Amount paid by the user in last subscribed plan
V6_PlanDaysLastTx	Number of days available in last plan taken by a user
V9_PlanAmountLastMinusAverage	Difference in amount paid by user in their last transaction and average plan
V8_PlanDaysLastMinusAverage	Difference in number of plan days in last transaction and average plan days of a user
V11_IsAutoRenewLastTx	Whether last plan subscribed was auto_renewed
V12_LastExpirationDate	Expiration date of last plan subscribed by a user
V13_DaysSinceFirstRegistration	Number of days passed after registration till February 2017

Table 3: Variables 14-20

Engineered Features	Description
V14_NumberSessions	Number of times user logs into KKBOX
V15_DaysBetweenFirstLastActive	The difference between the last user log date and first user log date
V16_LastSessionDate	Last time user logged into KKBOX
V17_DaysSinceActive	Difference between February 28th ,2017 and the last session date
V18_SumNum25	Sum of number of songs listened less than 25% of the song length for an individual user across all sessions
V19_SumNum50	Sum of number of songs listened less than 50% of the song length for an individual user across all sessions
V20_SumNum75	Sum of number of songs listened less than 75% of the song length for an individual user across all sessions

features capturing User Log information

Since the user logs is arguably the most important data available out of all the files, as it represents how the user actually used the service, we tried to extract as much information from the 392 million sessions. For each user, we counted the number of days the user logged into KKBOX, the number of days between the first time they used KKBOX and the last day they used KKBOX, their last session date, as well as the number of days since they were active (February 28th – last session date). We thought these variables would give a good indication of active a user was since their acquisition. We also rolled up all of the listening habits data into summed and averaged amounts, since it is important to know how much they have listened to music on KKBOX, how long they spend during each session, and the variety of what they listen to during each session. In an attempt to capture what each user's listening habits have been lately, we also grabbed both the number of unique songs played and time listening to music during their last active session. This resulted in the addition of 20 new features to our dataset.

Table 4: Variables 21-27

Engineered Features	Description
V21_SumNum985	Sum of number of songs listened less than 985% of the song length for an individual user across all sessions
V22_SumNum100	Sum of number of songs listened less than 100% of the song length for an individual user across all sessions
V23_SumNumUnq	Sum of number of unique songs played for an individual user across all sessions
V24_SumListenTime	Sum of listening time in seconds for an individual user across all sessions
V25_AvgNum25	Average number of songs listened to less than 25% of the song length for an individual user across all sessions
V26_AvgNum50	Average number of songs listened to less than 50% of the song length for an individual user across all sessions
V27_AvgNum75	Average number of songs listened to less than 75% of the song length for an individual user across all sessions

Table 5: Variables 28-33

Engineered Features	Description
V28_AvgNum985	Average number of songs listened to less than 98.5% of the song length for an individual user across all sessions
V29_AvgNum100	Average number of songs listened to less than 100% of the song length for an individual user across all sessions
V30_AvgNumUnq	Average number of unique songs played for an individual user across all sessions
V31_AvgListenTime	Average listening time in seconds for an individual user across all sessions
V32_LastNumUnq	Number of unique songs played during a user's last sessions
V33_LastListenTime	Listening time during a user's last sessions

4 Weight of Evidence

For two-class classification problems such as ours, weight of evidence can be a powerful metric to see the discriminative power of any independent variable in relation to the dependent variable. Trends in weight of evidence can also help us to understand how an independent variable affects dependent variable. For example, are recent customers more likely to churn or long-term customers.

Weight of evidence metric is commonly used in credit-scoring and is described as a measure of the separation of 'good' and 'bad' customers. Since we are trying to classify customers into churn and non-churn, we felt that weight of evidence can help us understand if our engineered features really have some discriminative power when it comes to this classification.

Below are the steps we followed to calculate WOE.

1: For a continuous variable, split data into bins

2: Calculate the number of events (churn) and non-events (non-churn) in each group (bin)

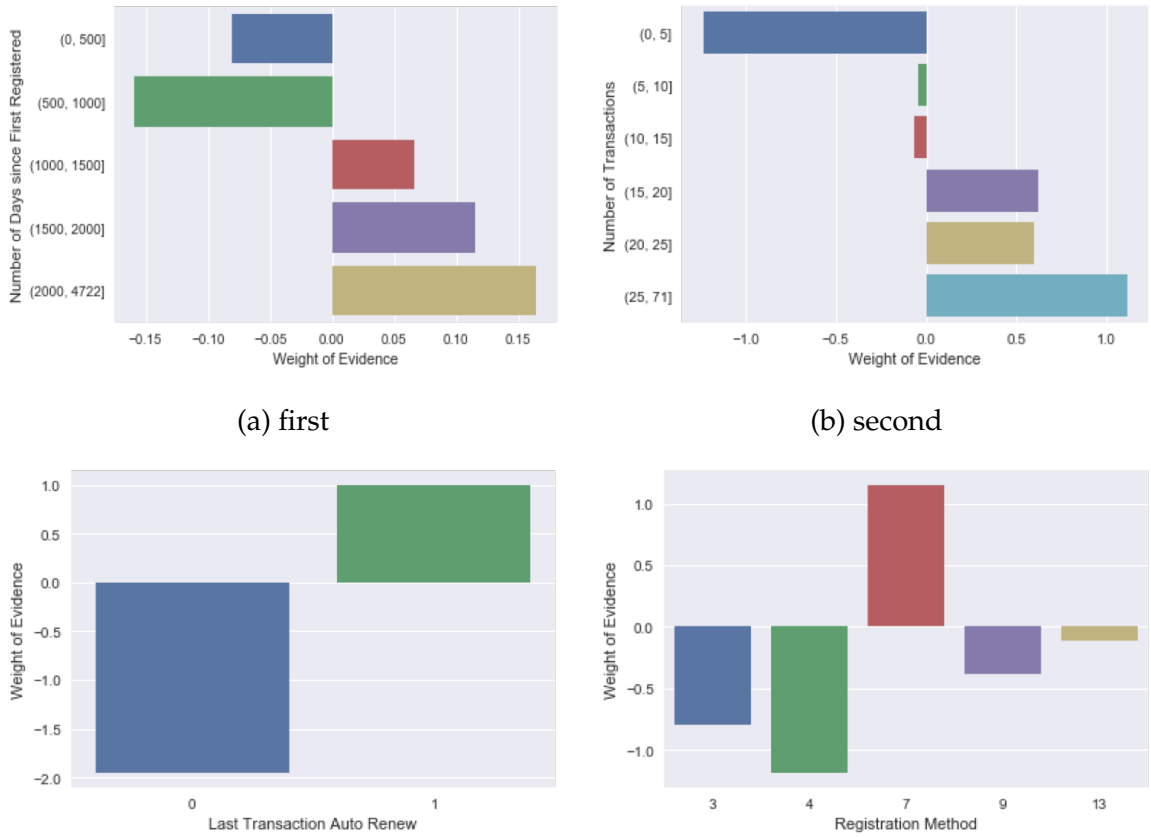
3: Calculate the % of events and % of non-events in each group.

4: Calculate WOE by taking natural log of division of % of non-events and % of events

For categorical variables, we do not have to split the data into bins as we can consider each category as a separate bin.

$$WOE = ln\left(\frac{\%\ of\ non-events}{\%\ of\ events}\right)$$

Below are our weight of evidence calculations for some of the features.



(a) first

(b) second

(c) third

(d) fourth

Figure 2: Weight of Evidence Graphs

Figure 3: WOE-number of days since registration

V13_DaysSinceFirstRegistration_bins	0	1	Total_Bin	Percent_NoChurn	Percent_Churn	WOE
(0, 500]	257924	17696	275620	31.527082	34.230888	-0.081048
(500, 1000]	144247	10732	154979	17.631887	20.759827	-0.159147
(1000, 1500]	143861	8489	152350	17.584705	16.421000	0.066511
(1500, 2000]	96920	5428	102348	11.846919	10.499845	0.115525
(2000, 4722]	175151	9351	184502	21.409407	18.088440	0.164376

Above table shows our weight of evidence calculation for number of days since registration. We divided the data into five bins and then calculated the percentage of no-churn vs percentage of churn customers in each bin. Weight of evidence was then calculated as the log ratio of these percentages as stated previously.

Fig.2(a) shows the weight of evidence numbers for number of days since registration by bins. We can see that the WOE is negative for the first two bins which means that percentage of churned users is higher in these bins whereas it's opposite for the bins where number of days since regis-

tered is more than 1000.

WOE calculations for number of days since registration suggests that this independent variable is capable of discriminating between churn and no-churn customers and from the WOE trends in the bins, we can conclude that long-term customers are less likely to churn as compared to relatively new customers.

Fig.2(b) shows the WOE for number of transactions. WOE is negative for low number of transactions and positive for high number of transactions, suggesting that users with a high number of transactions are less likely to churn.

Last transaction auto renewal status is a categorical variable. If the variable is set to 1, it indicates that in the last transaction, the user set the subscription to renewal and if not, it is set to 0. Since this is a categorical variable, we did not divide the data into further bins. Negative WOE for class '0' and positive for class '1' (Fig.2(c)) suggests that users who opted for auto renewal in their last transaction are less likely to churn.

Next, we calculated WOE for the registration method feature. This is also a categorical feature, where each class is a different method that a customer can use to register for Kkbox. The classes are encoded as digits and we do not have information about which digit corresponds to exactly what registration method.

We can see from Fig.2(d) that WOE is negative for all registration methods except method 7. Users who signed up using method 7 seem to be less likely to churn.

Weight of evidence calculations helped us to be more confident about the features we engineered before we proceeded for model building as they clearly seem to have discriminative power in relation to the question of whether or not a user churns from kkbox.

5 Modeling

5.1 Model Evaluation

For the Kaggle challenge, the quality of model is based on the log loss metric. But from a utility point of view for the business, other metrics such as Precision and Recall are critical in determining the benefits from the modeling exercise. For a churn prediction problem, the cost of not being able to identify true positives (churn) can be huge. Recall, the percentage of true positives captured by the model, is an important metric for churn prediction problem. Therefore, keeping business objectives in mind, our objective was to evaluate our models based on the Recall constrained to

acceptable Precision, while focusing on the main objective of the competition, minimizing log loss.

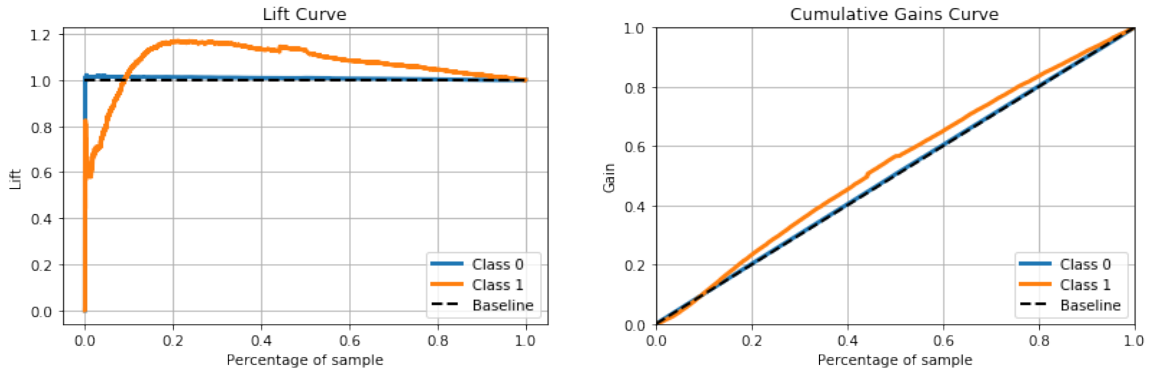
5.2 Logistic Regression

Logistic regression is part of the class of generalized linear models, while linear in the parameter space, it is put through a logistic transform to achieve well calibrated probabilities. This is the first model this paper considers to create baseline scores for the more complicated models.Logistic regression can be sensitive to the magnitude of its variables, so the data were scaled and pre-processed before hand. Since logistic regression is relatively fast and is very interpretable, it makes for a great starting point.

As with every model, logistic regression requires hyper parameter tuning. Sklearns grid-search was used to tune three main hyper parameters. The penalty term which controls parameter shrinkage, the cost penalty which controls the effect of outliers/errors, and finally the the solver type to see if there is any performance gain between solvers.

As expected, the logistic regression did not perform very well. This is likely because the data is very complex and has significant non-linear effects. Since logistic regression needs manual variable selection apart from the penalty shrinkage, it is unable to capture non-linear effects as well as other models. The model achieved an accuracy of 93.41 and a log loss of .691, its important to note that the accuracy is below a random selection of data. Interestingly, both recall and precision are 0 which means the logistic regression predicted only the not-churn class, this could be due to the low number of positive classes in the dataset. While logistic regression did not perform well, it is a good baseline metric for the other models in this paper.

Figure 4: Lift & Cumulative Gains



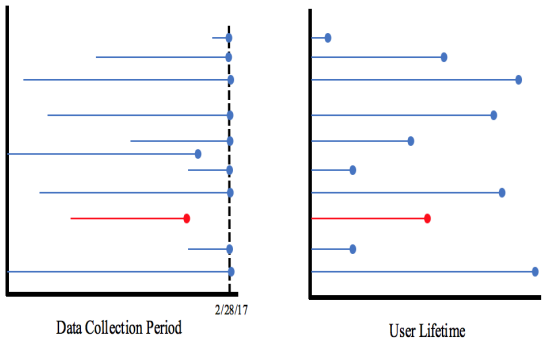
(a) Lift

(b) Cumulative Gains

5.3 Survival Analysis

Survival analysis is a set of methods for analyzing data where the outcome variable is the time until the occurrence of an event of interest. In this project, the event of interest would be a user churning and not renewing their subscription within 30 days. This class of models particularly apt for churn prediction because it can handle censored data. Censoring occurs when the exact time for the event of interest is unknown. This could be due to death, dropout, or termination of a study. For the KKBOX dataset, most subject are right-censored as they do not churn before the end of data collection. The diagram below depicts users who subscribed to KKBOX at varying dates (left) and their respective KKBOX lifetime (right). Users in blue are right-censored since they made it to the end of the data collection data and users in red churned.

Figure 5: Pictorial of right-censoring

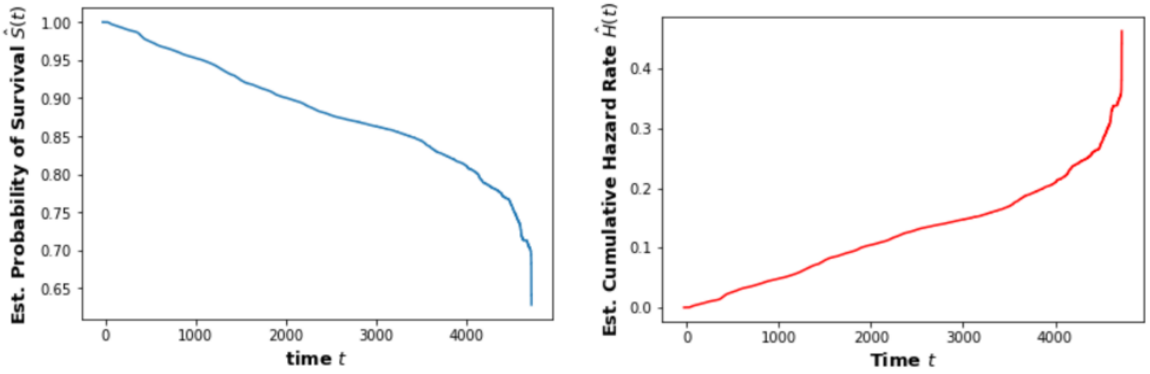


For right-censored data, each user record consists of a set of covariates and the time when an event occurred or the time of censoring. Since censoring and experiencing and event are mutually exclusive, the observable survival time for an individual user is can be represented as:

$$y = min(t, c) = \begin{cases} t & churn = 0 \\ c & churn = 1 \end{cases}$$

Given the knowledge of whether a user churned and their customer lifetime, the survival probability $[S(t) = P(T > t)]$ can be calculated at each point in time using the non-parametric Kaplan-Meier estimator. Furthermore, the cumulative hazard rate can be obtained using the Nelson-Aalen estimator. Below are sample curves from the aforementioned curves

Figure 6: Prob. of Survival & Cumulative Hazard Rate



(a) (b)

It is clear from the graphs above that KKBOX is facing a battle of attrition. There is only a steady decline in the probability of retainment, but once customers reach a around a decade there is a dramatic increase in risk of churning. This could be because Spotify has become available in Asia and older customer may be eager to head for greener (literally and figuratively) pastures with Spotify. While these estimators provide some insight into KKBOX's user retention, they do not provide a nice means for estimating the probability of an individual churning at a given point in time as they do not allow for time-dependent covariates. Therefore, a Cox Proportional Hazards Model was fit to the covariates 'V1_NumTransactions', 'V2_NumUniquePmtMethods', 'V4_PctTxCancel', 'V6_PlanDaysLastTx', 'V9_PlanAmountLastMinusAverage', 'V15_DaysBetween-FirstLastActive', 'V17_DaysSinceActive', 'V23_SumNumUnq', 'V24_SumListenTime', 'V31_Avg-ListenTime'. These covariates were selected from a larger subset by fitting each variable individually and recording Harrell's concordance index (c-index) on the training set. The model produced from this analysis had an accuracy of 89.16%, precision of 74.28, recall of 48.33, AUC of 66.85, and log loss of .1751. This represented a significant improvement over the logistic regression classifier.

5.4 Random Forest

Some ensemble decision tree methods including bagged trees have a significant amount of tree correlation, this increases the variance of the models and reduces the accuracy of the predictions. Random forests achieve more independent trees through randomly selecting from the feature space and building an individual tree from that selection. Each tree casts a vote for a new observation, and the class with the most votes wins. Since the feature space for this competition is

quite large, a random forest was fit to the data to see if there were any significant improvements from de-correlating the trees.

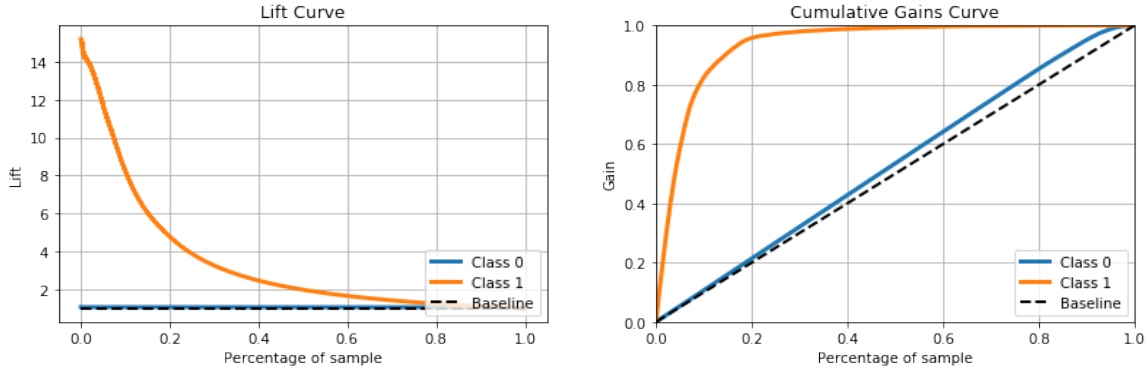
The main tuning parameters for a random forest are the number of trees to build, the number of features to include, and the maximum size of the tree. Each of these features attempt to find a balance between the bias/variance trade off. For the specific application of the KKBox competition, a grid search was performed and the optimal parameter were 500 estimators, a max features set to 7 and a max depth of 25. This is consistent with the concept of many strong learners that are averaged together in the end to decrease both bias and variance. with a depth of 25, trees are able to grow significantly to fit the data, but also need 500 trees to minimize over fitting.

The final model offered fairly good results, with an accuracy of 96.05 and a log loss of .1048, this indicates the data benefited from de-correlated trees, with improved accuracy from base line. While random forests is a more competitive learning approach with each tree competing against one another to cast a vote, the next section goes into more detail about collaborative approaches, where the trees work together to fit the data.

Table 6: Top Five most Important Variables

Features	**Importance**
V17_DaysSinceActive	0.122119
V4_PctTxCancel	0.0935567
V11_IsAutoRenewLastTx	0.0711686
V9_PlanAmountLastMinusAverage	0.0484664
V1_NumTransactions	0.0416764

Figure 7: Lift & Cumulative Gains



(a) Lift

(b) Cumulative Gains

5.5 XGBoost

Decision trees are notorious for being unstable methods for classification and regression applications due to their greedy algorithmic approach. Classification trees attempt to discretize the data in a way which makes more homogeneous groups as the tree gets larger. While this is an effective approach at classification, trees are prone to over fitting the training data due to making complex tress with maximum depth. Boosting provides a solution for this problem, by building a sequence of small trees which are weighted in a way to correct what the previous tree mis-classified, boosting models prevent over fitting and significantly improve test classification compared to a simple decision tree. This is accomplished through creating a series of weak learners which individually are poor classifiers but when combined out perform its individual components. XGBoost takes this a step further by optimizing the speed and performance of normal boosting models while taking advantage of the stochastic gradient boosting framework. By adding Parallelization to the algorithm and other performance enhancing techniques, XGBoost is able to outperform other boosting techniques in both speed and classification.

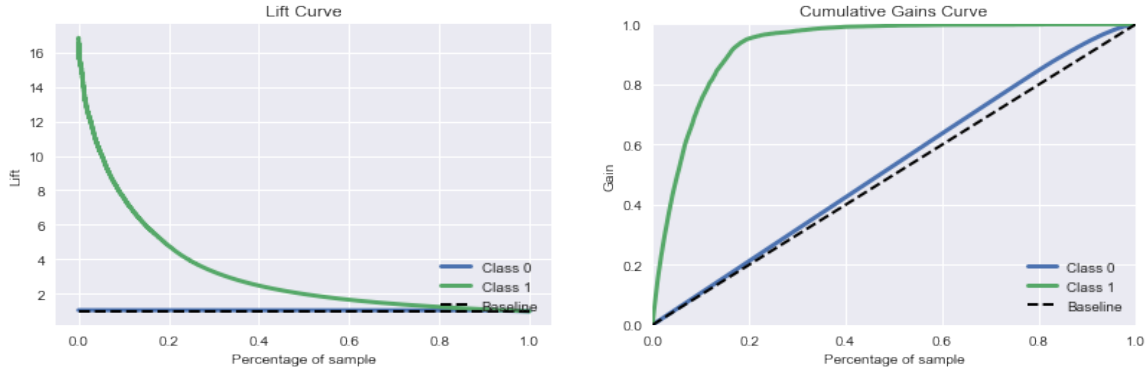
In regards to the KKBox competition, Sklearn's XGBoost classifier was fit to the data with the pre-processed data as before. To tune the classifier there were four main hyper-parameters considered: max depth, minimum child weight, gamma, and max delta step. The max depth of a tree controls how large the tree can grow, for boosting models this controls how weak the individual learners actually are. Due to the the large class imbalance in the data set, the minimum child weight will also be tuned. This parameter controls how many partitions the individual trees make by checking the sum of instance weight in the partition. This helps in modeling when the classes are imbalanced. Gamma controls the regularization of the data and hence controls overfitting the data. Finally, because of the extreme class imbalances, the max delta step will also be tuned which helps in the regularization process to prevent over fitting. To tune these parameters a cross validated grid search was performed on a sample of the data, due to computational constraints the tuning could not be performed on the full data set. The grid search also evaluated the parameters on the evaluation metric provided by the competition which was log loss. The final hyper parameters were a max depth of 9, a minimum child weight of 1, a gamma of 4, and a max delta step of 2. These parameters indicate that the data requires relatively large trees with preventative regularization to avoid over fitting. These parameters were then used to train a XGBoost model with the entire data set, and performed very well regarding log loss.

Once the model was trained on the full data set, it was then fit on the test set and achieved a log loss of .0908, an accuracy of 96.48% and a recall of 54.48%. The Lift curve also shows that the model can predict the positive class 16 times better than baseline accuracy. Since the recall is so low compared to the accuracy. indicates the class imbalance severely affects the accuracy of the positive class, even though the log loss is a competitive score. Due to this issue, we ran the XGBoost model on the sampled data set to see if it improved the recall. Doing this increased the recall to 88%, accuracy to 87.50%, precision to 87%, and AUC to 92.10%. While this approach to modeling is very beneficial to KKBox as a business, the sampling creates poorly calibrated probabilities and the models log loss score is only 0.261.

Table 7: Top Five most Important Variables

Features	**Importance**
V16_LastSessionDate	0.506374
V9_PlanAmountLastMinusAverage	0.0347575
V4_PctTxCancel	0.0324748
V1_NumTransactions	0.0311827
V13_DaysSinceFirstRegistration	0.0293738

Figure 8: Lift & Cumulative Gains



(a) Lift

(b) Cumulative Gains

6 Results

Table 8: Results

Model	Accuracy	Log Loss	Precision	Recall	AUC
XGBoost	96.48	.0908	80.03	54.48	76.81
Random Forest	96.05	.1048	82.05	51.06	75.14
Logistic Regression	93.41	.6931	0	0	49.99
Cox Proportional Hazard	89.16	.1751	74.28	66.85	66.85
Sampled XGBoost	87.5	.261	87	88	92.10

Ultimately the best model for the KKBox competition was the XGBoost model with a log loss of .0908. The data was incredibly complicated with confounding effects, which the XGBoost model was capable of capturing. Especially compared to the baseline model which was the logistic regression, every model trained was able to out perform it. XGBoost is a very scalable and adaptive model, it would be very useful to KKbox in predicting churn.

In terms of business use, the cox proportional hazard model from the survival analysis offered competitive recall and accuracy given the extreme class imbalance. Also the sampling technique used for the XGBoost model far out-performed the cox proportional hazard with respect to the classification statistics except for accuracy and log loss.

7 Discussion

Although all of the previous work resulted in the reporting of a table of performance metrics, a number of insight were gained at every stage of the process. For instance, from the exploratory data analysis phase it was found that city 1 had a substantially lower rate of churn amongst its denizens than other cities. Given that city 1 also had the most users, there could potentially be a social diffusion effect occurring amongst inhabitants. KKBOX may want to estimate the coefficients of innovation and imitation from this city 1's data. If there is a high coefficient of imitation, KKBOX could pay Asian celebrities to endorse their product and the adoption rate could be fast in other cities. Also gathered from the exploratory data analysis was the fact that users who have canceled their subscription in the past are at a much higher risk of churning than those that have not. For these customers, KKBOX could offer a promotional discount on their year-long subscription, thus locking in the customer for a longer period of time and reducing their tendency to cancel subscriptions. In a similar vein, they can offer discounts at the beginning of the year so

those with longer subscriptions do not churn at the beginning of the year (e.g. offer first month or two discounted).

On a personal level, we learned the importance of not just looking at slices of the data with respect to the target variable. It is not enough to know how each variable is associated with the target variable independent of the other features. Interactions amongst the covariates can filter through to your models and bias predictions, so it is necessary to address issues discovered during data exploration. Furthermore, exploratory data analysis provides invaluable information about issues with the data. As was the case with our dataset, there is always pre-processing that must be performed before proceeding to feature engineering and modeling. For example, before being able to calculate the summed and average listening times for a user session we had to address unrealistic listening times for an individual session. From an internal tracking standpoint, KKBOX may want to find a way to encourage users to report an accurate age if they believe it could have predictive power for churn. This was one of the least reliable variables found in the data set and proved to be not very useful in its current state. Thus, having access to reliable data is crucial if you hope to gain any insight into what drives your business.

Another valuable lesson was the massive importance of feature engineering. When trying to run XGBoost on the base variables (not reported), the model did not perform well in terms of log loss or recall. However, once additional features were extracted from the transactions and user logs data sets, these features had a large impact on model performance. This was evidenced by their high ranking in feature importance for both Random Forest and XGBoost. It was actually enjoyable trying to think of relevant features to engineer from the data and trying to implement them. Since the user logs file was so big feature engineering had to be done through Spark on AWS. It was a valuable experience figuring out how to translate seemingly trivial actions [in python] into Spark syntax.

While initial exploratory data analysis provided some interesting insights, the weights of evidence calculated for the engineered features proved just as valuable. Those engineered features that displayed a monotonic relationship with churn rate are especially valuable because they can be represented with simpler models such as logistic regression. It was nice to see that very intuitive predictors (e.g., Number of days since last active and customer tenure) of churn showed high weights of evidence. Therefore, in their own models KKBOX should spend a hefty portion of their time thinking of intuitive features they can collect from their base data. Otherwise they will be severely crippled by noisy data and lack of informative predictors.

After completing exploratory data analysis, pre-processing features and removing outliers, engineering new features, and getting a feel for how much information these predictors contain it seemed like we were in the clear. However, properly tuning the models turned out to be much harder than expected. Because of the significant class imbalance, we had to train some of the models on an over-sampled [with respect to the positive class] subset of the data. However, if class weights were not properly applied when obtaining class probabilities, the log loss performance metric turned out very poor. Thus, we quickly realized the importance of applying class weights when forming predictions in order to avoid biasing your model from your sampling method. Furthermore, merging the datasets resulted in missing values for some users. Thus, we had to perform imputation just to run certain models (e.g. random forest classifier). The tradeoff between using fancier imputation methods versus class conditional medians or means extended to hyperparameter tuning. The method for which we chose to impute methods was a "knob of complexity we had to take into account when building the models.

As of now, KKBOX implements survivor analysis to predict user churn within their business. However, as can be seen by our model performance, there is still much to be desired beyond basic survivor analysis. Therefore, we suggest that KKBOX looks into using mixed models to predict user churn. More specifically, we would like to look into how scikit learn's scikit-survival more advanced models. These include gradient boosting survival analysis, fast survival SVM, hinge loss survival SVM, and their ensemble selection regressor that chooses base learners for regression that accounts for the accuracy and correlation of errors. We believe that combining the robustness of gradient boosting with the time-sensitivity of survival analysis would prove very useful in predicting churn. Going further, one could model each user as a time-series and feed these lower level models into rolled up aggregate models.

In conclusion, there was a lot of insight to be gained at every stage of the model building process. We learned the importance of doing the best possible job at each stage of development because decisions propagate through the pipeline. Our overall suggestion to KKBOX is to focus their time and efforts on feature engineering and maintaining the quality of their data. If they improve in these areas then their model performance will follow.

References

[1] Lu, J. & Ph, D. Predicting Customer Churn in the Telecommunications Industry — An Application of Survival Analysis Modeling Using SAS. Techniques **114–27**, (2002)

[2] Coussement, K. & Van den Poel, D. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. Expert Syst. Appl.**34**, 313–327 (2008).

[3] Xie, Y., Li, X., Ngai, E. W. T. & Ying, W. Customer churn prediction using improved balanced random forests. Expert Syst. Appl. **36**, 5445–5449 (2009).

[4] Tsai, C. F. & Lu, Y. H. Customer churn prediction by hybrid neural networks. Expert Syst. Appl. **36**, 12547–12553 (2009).

[5] Martins, H. Predicting user churn on streaming services using recurrent neural networks Predicting user churn on streaming services using recurrent neural networks. (2017).

[6] Burez, J. & Van den Poel, D. Handling class imbalance in customer churn prediction. Expert Syst. Appl. **36**, 4626–4636 (2009).