KKBox User Churn Prediction Project

By Marshall Lee

Springboard

February 2nd, 2023

Spotify has reigned supreme in the music streaming industry for many years now, boasting over 80 million tracks and 456 million active users in 2023. Despite its large selection of music, however, it has only a small selection of Chinese songs. As a spotify user based in the US, I was frustrated since I couldn't find many of the Chinese songs my parents used to blast in the car.

Luckily, KKBox has filled this void for my family and I. The Taiwan-based company features an impressive 45 million tracks to listen from and is extremely popular in countries like Japan, Taiwan, Hong Kong, and Malaysia. For this project, my goal is to build a model to predict user churn* and help KKBox compete with titans like Spotify. I will also explore the top features associated with churn using a dataset from Kaggle.

Data Wrangling

The dataset comprises four tables. The train table provides churn labels for users whose expiration dates fall in March 2017. The transactions table, the largest of the four, contained over 1.4 million entries of data. Cleaning this was challenging since some of the transaction data was purposely labeled with an inaccurate date by the poster. Take a look at user '5ty4nZkq54z93wQtBN7RHVYj8rNghBDCVBH+3xmxf0I=' - he clearly renewed his subscription on a weekly basis but the transaction date doesn't reflect this.

payment_method_id	payment_plan_days	plan_list_price	actual_amount_paid	is_auto_renew	transaction_date	membership_expire_date
38	7	0	0	0	20151215	20170402
38	7	0	0	0	20151215	20170409
38	7	0	0	0	20151215	20170416
38	7	0	0	0	20151215	20170423
38	7	0	0	0	20151215	20170430
38	7	0	0	0	20151215	20170507
38	7	0	0	0	20151215	20170514
38	7	0	0	0	20151215	20170521
38	7	0	0	0	20151215	20170528
38	7	0	0	0	20151215	20170604
38	7	0	0	0	20151215	20170611

***Figure 1**: Dealing with dirty data in the transactions table. Around 20% of the data looked like this.*

---

* KKBox defines “churn” as no new valid service subscription within 30 days after the current membership expires. Source: https://www.kaggle.com/competitions/kkbox-churn-prediction-challenge/data

To solve this problem, I divided the data into duplicate entries and non-duplicate entries first. Then, for the duplicate entries, I took the sum of the membership duration and the latest transaction date to avoid removing any relevant data for my analysis. I also removed rows/columns with null values. After more aggregation and cleaning on the remaining tables, I combined them into one dataframe. No other outliers were found in the dataset.

Out of our sample set of 725,723 users, 6.4% of them churned in March 2017. Below is the list of the 20 features for our analysis:

- *Msno* - the user ID (unique)
- *Is_churn* - our target variable and what we are trying to predict. Takes only Boolean values ("0" as False and "1" as True)
- *Payment_method_id* - method of payment
- *Payment_plan_days* - number of days of the payment plan
- *Plan_list_price* - plan listed price
- *Actual_amount_paid* - the actual amount the customer paid. Can be different from plan listed price
- *Is_auto_renew* - Whether the customer chose to auto-renew during that specific transaction. Takes only Boolean values
- *Is_cancel* - Whether the customer chose to actively cancel their subscription during that specific transaction. Note that a customer who cancels their subscription does not necessarily mean that they churned (i.e. they might have changed their subscription plan)
- *Sum_membership_duration* - Created by taking the total membership duration of each user
- *Avg_time_between_transactions* - Created by taking the average time between each transaction
- *Num_unq* - the number of unique songs a user has listened to
- *Percent_25, 50, 100* - the % of songs a user listens to until x% completion. For example: a value of .15 in the Percent_25 column indicates that the user listened to 15% of songs until they were only 25% completed before changing the track.
- *City* - A user's city of residence in numerical format
- *Registered Via* - Registration method in numerical format
- *Latest_transaction* - the date of the user's latest transaction
- *Latest_membership_expiration* - the date of a user's latest membership expiration. Can take place in the future
- *Latest_listen_date* - the date of the user's latest listen with the service
- *Registration_init_time* - the date of when the user registered for that particular transaction

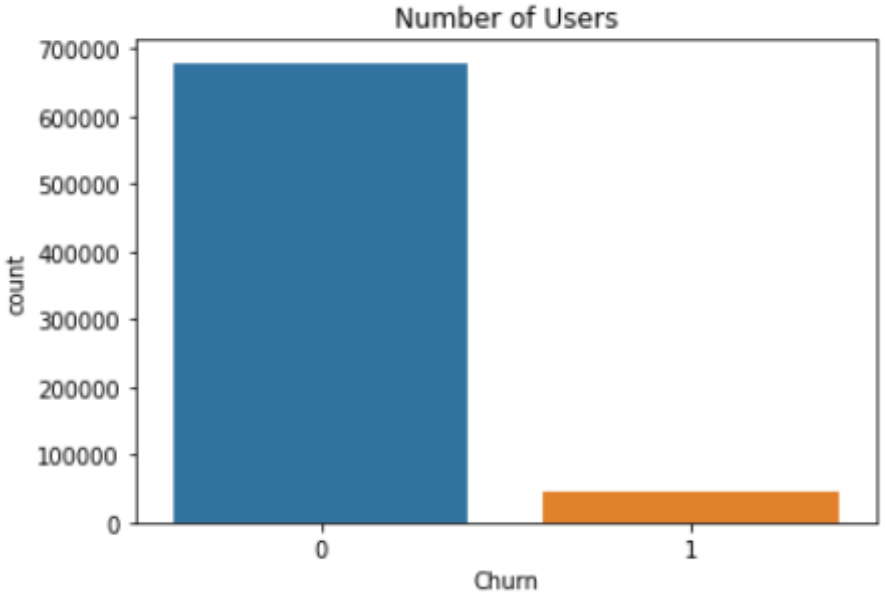


Figure 2: *Countplot of churned users shown in orange.*

Total membership duration

One of the first features that showed a high correlation with churn rate was actually one that we created during the wrangling step.

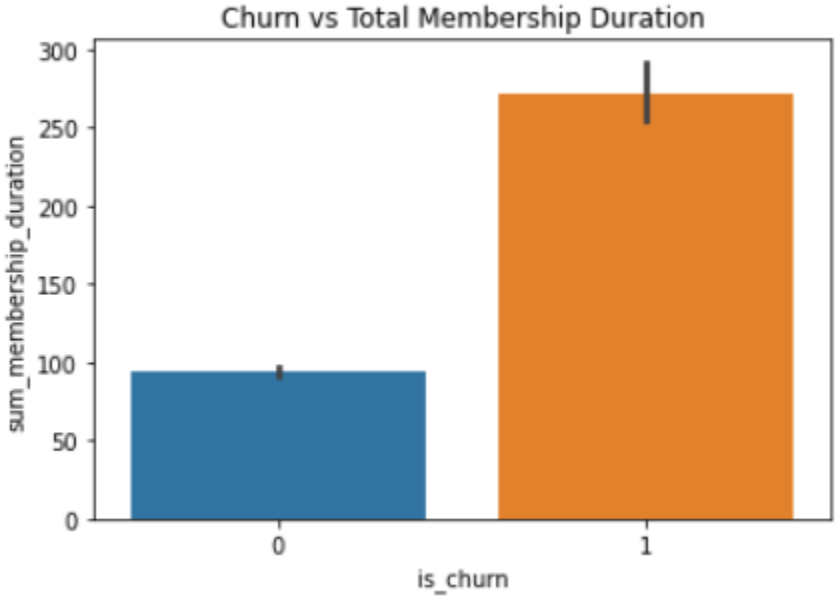


Figure 3: *Churned users seem to have longer memberships.*

On average, we see that churned users are subscribed for about 3 months longer than non-churned users. This is a very interesting find, considering KKBox has three main subscription options: a 30-day plan, a 90-day plan, and a 365-day plan.

Figuring out the exact values that were associated with churn was tricky because over 76% of users opt for monthly memberships. In other words, plotting a histogram of membership duration didn't tell me much since most of them fell in the 30-day bucket. Luckily I was able to use panda's qcut function to get evenly-distributed buckets based on frequency. Then I plotted these buckets against churn rate on a bar chart. We get a clear bucket as we see churn jump up to 5% for users whose membership is longer than 32 days.

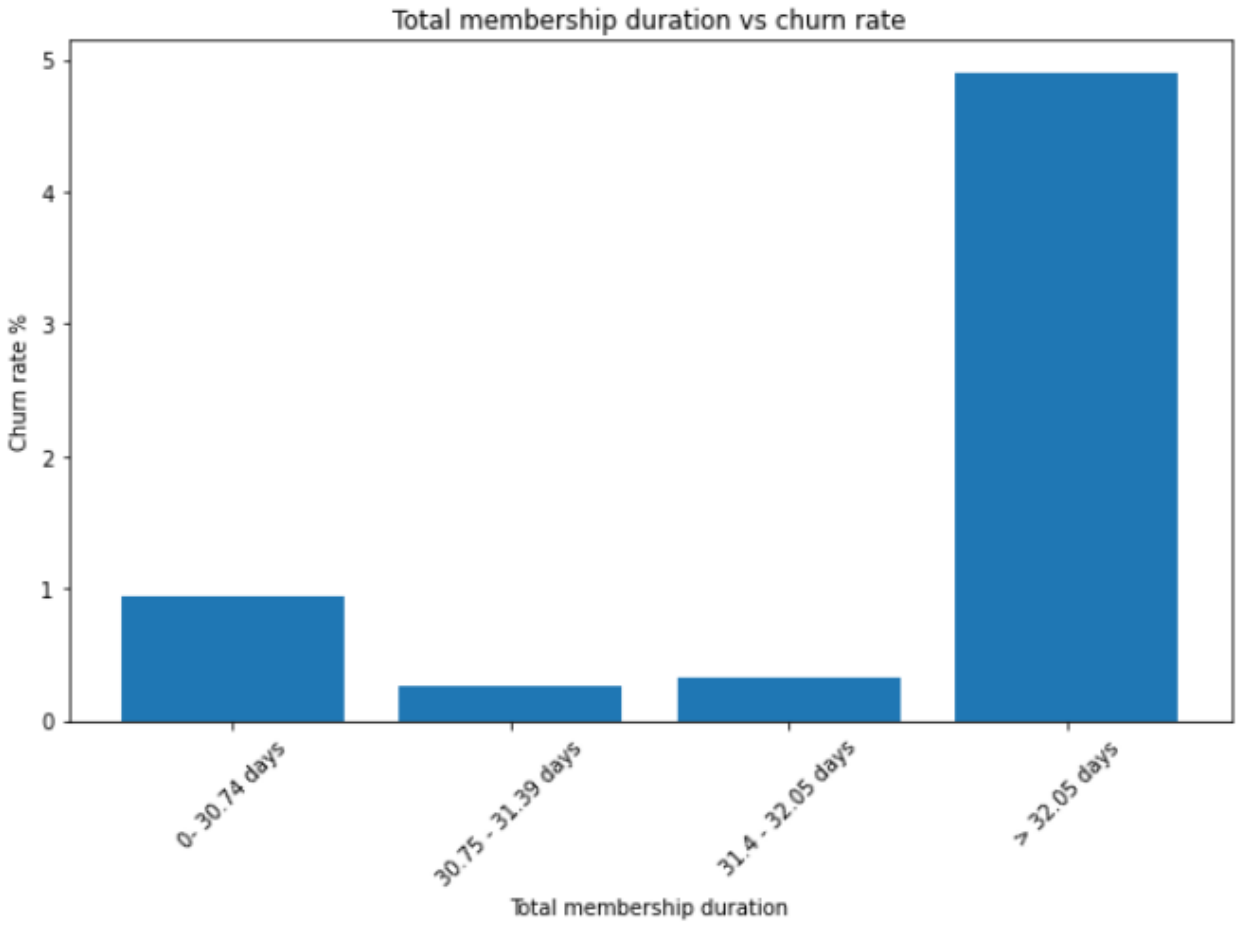


Figure 4: *Users with longer memberships longer than 32.05 days show a large difference in churn rate.*

Payment details

When looking at payment plan days and payment amounts, I faced the same challenge as I did during total membership analysis: since a large number of users chose 30-day payment plans, the dataset was heavily imbalanced. Pandas qcut helps again here and divides churn users into four distinct buckets.

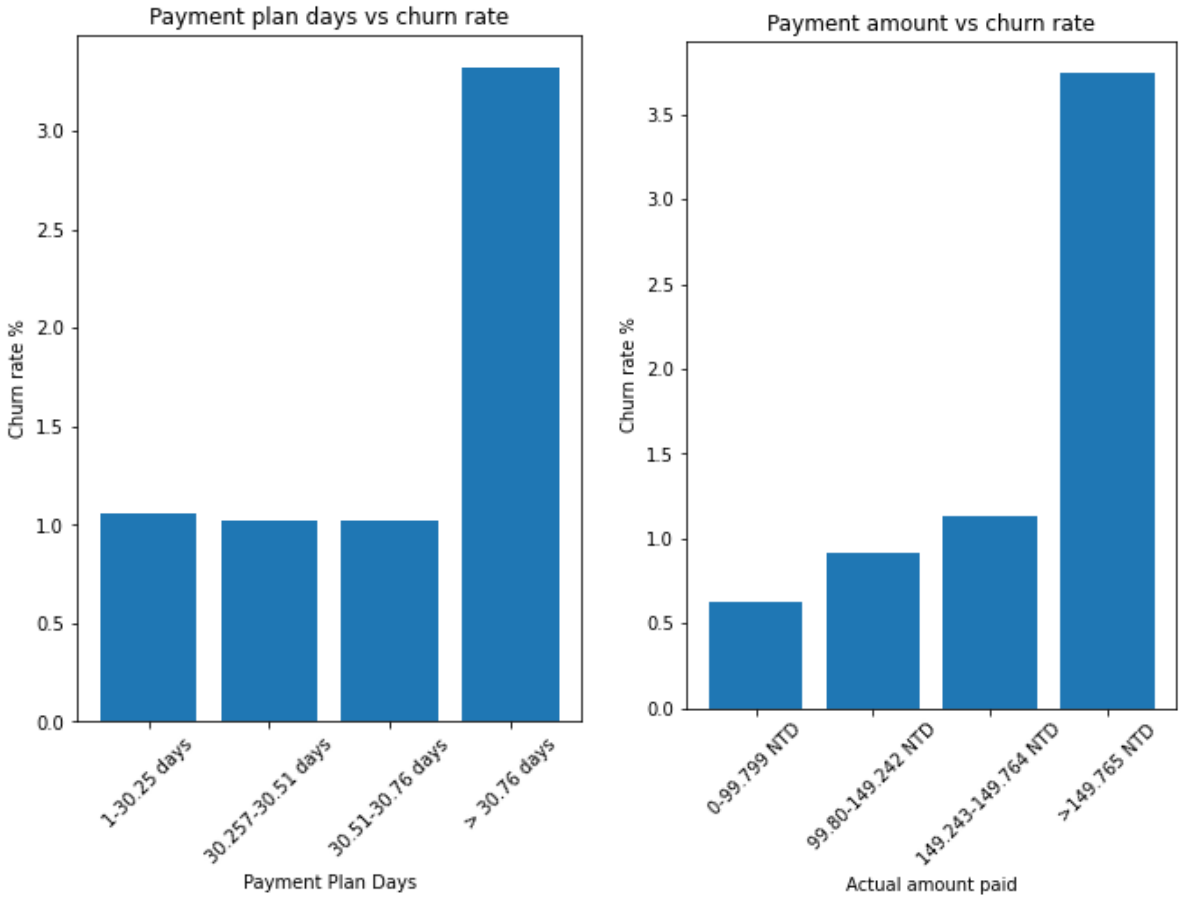


Figure 5*: Churn rate bar graphs after bucketing variables using the qcut function. We see differences in churn rate after a certain threshold.*

On average, churned users paid 300 Taiwan dollars more (equivalent to ~$10 USD) than users who did not churn, and opted for 90-day payment plans more often. This suggests that focusing on retaining long-term users may be the most effective strategy at reducing user churn.*

After plotting a heatmap for all numerical variables, we find several cases of

---

* Other feature correlations can be found in the Jupyter notebook.

multicollinearity - namely, all the variables related to price. Out of plan_list_price, actual_amount_paid, and payment_plan_days, I chose to remove plan_list_price since it was practically the same as actual amount paid (showed a Pearson correlation coefficient of 0.99). I chose to keep both actual_amount_paid and payment_plan_days.

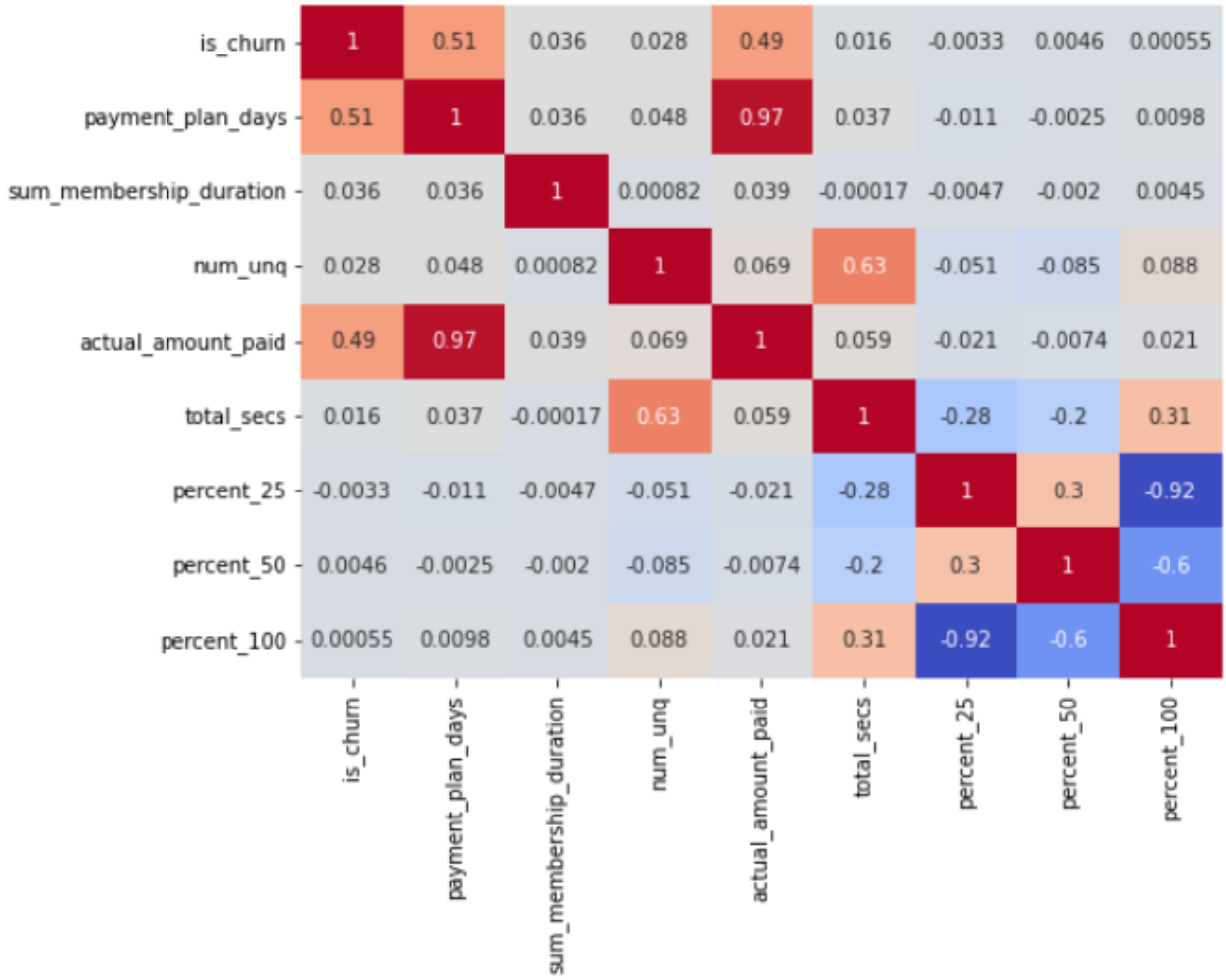


Figure 6: *Pearson Heatmap showing correlation amongst all our numerical variables. The three price-related features are correlated to churn.*

Auto-renewal

Auto-renewal was another feature heavily associated with churn rate. On average, users that chose to auto-renew their subscription rather than manually renew it were 25% less likely to churn. This variable showed the highest correlation coefficient of negative 0.34, indicating that it might be another possible solution to our problem.

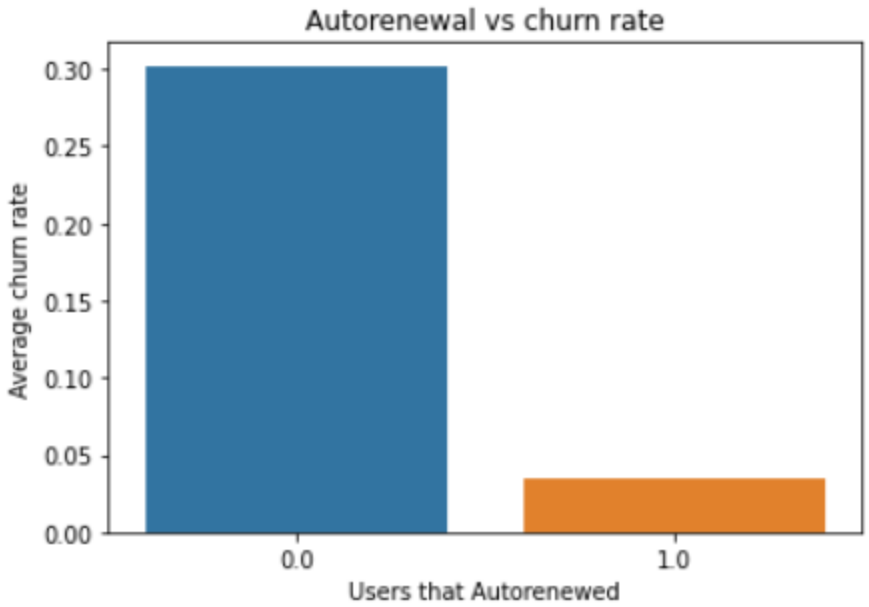


Figure 10: *Autorenewal against average churn rate.*

Feature Engineering

Now that we have our top features, let's check the correlation coefficients for each feature against churn rate to determine the most effective solutions for reducing it. I divided the numerical features from categorical features and used Point-Biserial/Cramer's-V respectively to get these numbers. The resulting table is shown below.

Feature	Correlation Coefficient
payment_method_id	0.515816
payment_plan_days	0.506000
actual_amount_paid	0.492414
is_auto_renew	0.345880
is_cancel	0.254632
registered_via	0.156701
avg_time_between_transactions	0.049787
sum_membership_duration	0.035820

num_unq	0.027861
total_secs	0.015547

It seems that payment details and auto-renewal are both heavily associated with churn, which is no surprise given our preliminary analysis.

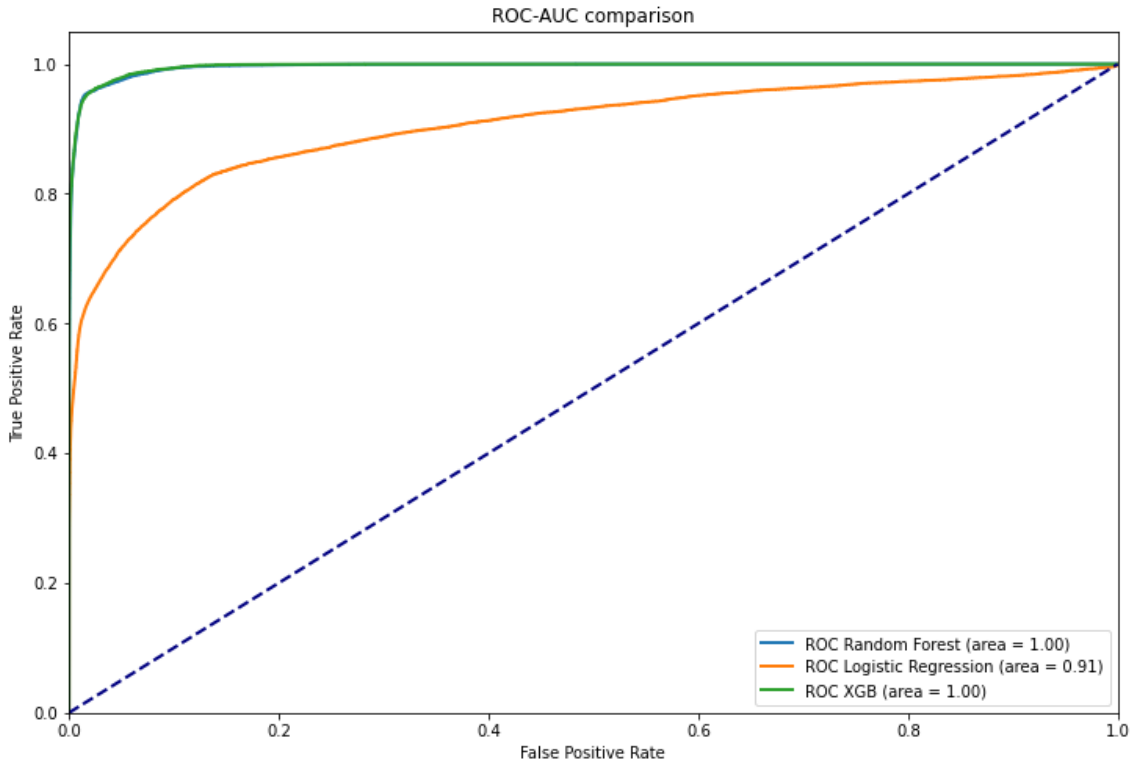
Model Building and Evaluation

I chose to compare three models: Random Forest, Logistic Regression, and Gradient Boosting. Since we are concerned with correct positive labels, I judged the performance of each model based on its F1-score and ability to minimize Type-II (false negative) errors.

After one-hot encoding the date-time objects and scaling the data, I started with the Random Forest. I used RandomizedSearchCV, which found that the best hyperparameters were 100 n-estimators with a max-depth of 20. Out of the 181,431 users in our test set, the Random Forest correctly identified 9794 churned users, and mislabeled 1857 users as non-churned. Its F1-score after tuning was 0.89.

Similarly, I used RandomizedSearchCV with the Logistic Regression model and determined the best C value to be 10 and solver to be Newton-CG. This time, the model performed poorly, only identifying 6229 churned users correctly. In addition, it mislabeled 5422 churned users! It drops to the bottom of the selection list with a F1-score of 0.65.

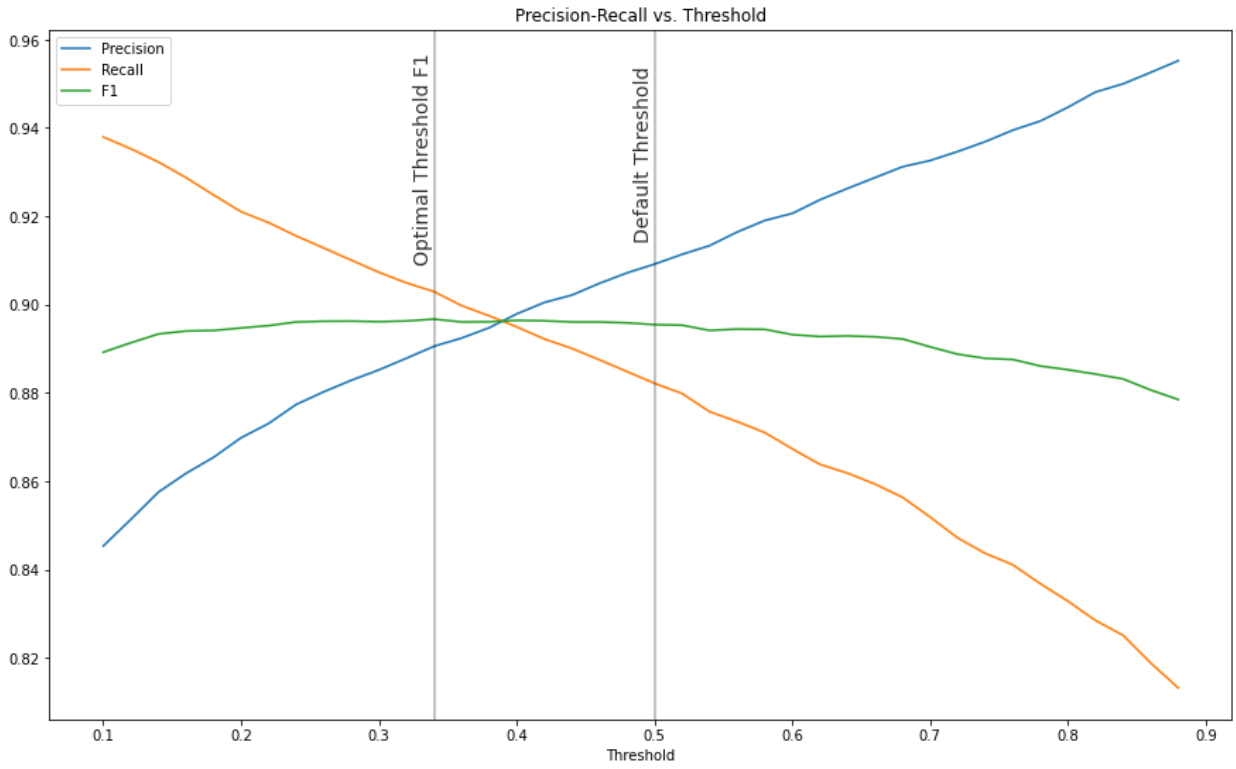
Extreme Gradient Boosted model outperformed the Random Forest by a slight margin with an F1-score of 0.90 after tuning. It correctly identified 10107 churned users, and only mislabeled 1544 users as non-churned. Looking at the score comparisons below, we could go with either the Random Forest or the XGB model. I'll choose the XGB model due to its slightly higher Recall and F1 scores.

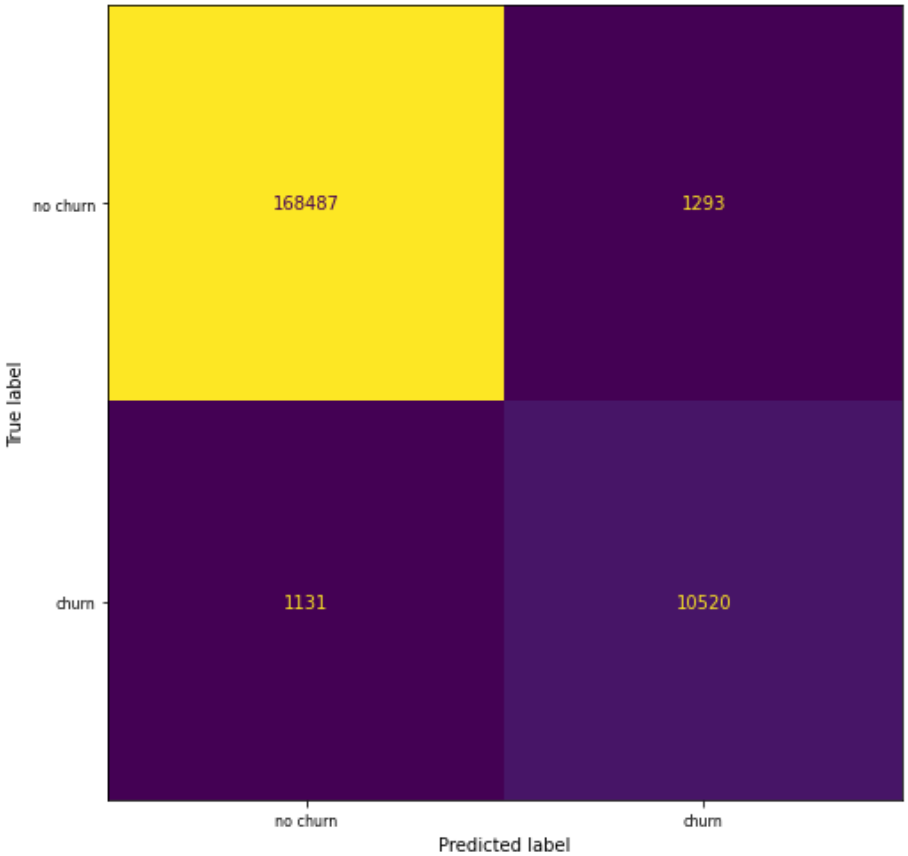


Model	Precision	Recall	F1-score
Random Forest	0.94	0.84	0.89
Logistic Regression	0.84	0.53	0.65
XGB	0.91	0.88	0.90

***Figures 8 and 9**: ROC/AUC graph and PR/F1 scores for each model. Though XGB performs slightly better than the Random Forest, both are viable.*

To further adjust for the imbalance in our data set, I also chose to change the default 0.50 threshold to 0.34, as it yielded the highest F1-score. The PR vs threshold graph and the resulting confusion matrix are shown below.


Precision-Recall vs. Threshold
Precision
Recall
F1
Optimal Threshold F1
Default Threshold
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Threshold


no churn
churn
True label
168487
1293
1131
10520
no churn
churn
Predicted label

Figures 10 and 11: *Threshold graph* and final confusion matrix of XGB using the 0.34 threshold.*

We have effectively improved the performance of our model after adjusting the threshold. It now correctly labels 10520 churned users, and limits the number of false positive labels to 1131.

Conclusion

It is clear long-term users churn more often than short-term users. Any strategies that focus on retaining long-term users will effectively reduce the churn rate by a significant amount. Before implementing any solution, I would first suggest gathering additional information in the form of customer surveys, if time allows. The survey would cover why a user is choosing to cancel their subscription, how satisfied they were with the service, etc.

We could then dive deeper into the data and brainstorm possible solutions based on customer feedback. For example, if a customer is choosing to leave because of the high costs associated with continued use, we could build a regression model to determine a new pricing model, or offer family/friend bundles in which groups can use the service at a slightly discounted price. Another solution could be to start a customer loyalty program that rewards users for their continued subscription - for example, an additional free week after three months of use.

Users that chose to automatically renew their subscription were also far less likely to churn than users that manually renewed it. This makes sense given the ease of access of automatic payments. KKBox could also explore this further by determining why some subscribers opt for manual payments instead, and developing strategies that specifically target these customers.

Lastly, since our dataset only focuses on users that have churned in March 2017, we can further supplement our data by gathering churn data for the rest of the year. This way, we can identify monthly/yearly trends we may be missing with our current dataset.