Capstone Project 1

Introduction

The Problem:
The modern-day lifestyle has turned into one that revolves around the subscription business model. Nowadays, you can get fresh ingredients delivered to your door, clothes to update your wardrobe, movies and TV shows to watch across different online devices, and even access to a variety of music to listen to during long commutes; all for a reasonable recurring monthly fee. The critical business question is no longer just “How will I find customers for my product?” but will also include “How will I retain my customers?”

KKBOX, Taiwanese subscription based music streaming service, shared their customer data on Kaggle to learn more about predicting churning. When users signs up for our service, users can choose to either manually renew or auto-renew the service. Users can actively cancel their membership at any time. For this project, the criteria of "churn" is no new valid service subscription within 30 days after the current membership expires.

The Inquiry:
Can we predict if a user will make a new service subscription transaction within 30 days after the current membership expiration date based on their behaviors and interactions with the product?

The Client:
The client is KKBOX. KKBOX will be able to use this information to offer special deals to customers who are likely to churn or on the verge of churning.

The Data:
KKBOX has provided 4 datasets:

1. Train Set, contains the user ids and whether they have churned for March 2017
    a. msno: user id
    b. is_churn: This is the target variable. Churn is defined as whether the user did not continue the subscription within 30 days of expiration. is_churn = 1 means churn, is_churn = 0 means renewal.
2. Transactions Data, contains the transactions data until 3/31/2017
    a. msno: user id
    b. payment_method_id: payment method
    c. payment_plan_days: length of membership plan in days
    d. plan_list_price: in New Taiwan Dollar (NTD)
    e. actual_amount_paid: in New Taiwan Dollar (NTD)
    f. is_auto_renew
    g. transaction_date: format %Y%m%d
    h. membership_expire_date: format %Y%m%d
    i. is_cancel: whether or not the user canceled the membership in this transaction.
3. User Logs Data, contains listening behaviors of a user until 3/31/2017
    a. msno: user id
    b. date: format %Y%m%d

    c. num_25: # of songs played less than 25% of the song length
    d. num_50: # of songs played between 25% to 50% of the song length
    e. num_75: # of songs played between 50% to 75% of of the song length
    f. num_985: # of songs played between 75% to 98.5% of the song length
    g. num_100: # of songs played over 98.5% of the song length
    h. num_unq: # of unique songs played
    i. total_secs: total seconds played
4. Members Data, contains user information
    a. msno
    b. city
    c. bd: age
    d. gender
    e. registered_via: registration method
    f. registration_init_time: format %Y%m%d

Data Wrangling

The data provided were all very large files.

- Train set contained 992,931 rows
- Members data contained 6,769,473 rows
- Transactions data contained 21,547,746 rows
- User log data contained 400,000,000+ rows

The train set is extremely biased with only a 6.4% churn rate. Having an extremely biased train set means that there's a high chance of not churn and it would be pretty accurate to label all users as not churn. Therefore, the majority group needs to be downsampled. There are 63,471 users marked as churned who are all kept in the new train set. I have randomly sampled 75,000 users from the not churned group to be included in the new train set as well. The resulting churn rate of the new train set is now 45.8%, which is a lot more balanced than before. All the other datasets will also only reflect those in this new train set.

All columns with date data were converted into datetime objects for easier manipulation and feature engineering relative to time.

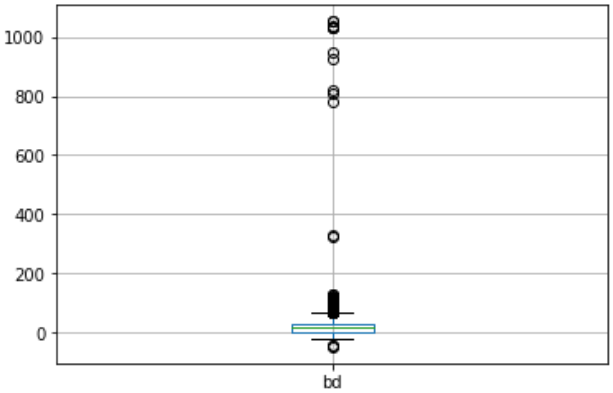
Members Data

The resulting members data has 123,815 rows meaning that we do not have member information for all of those in the train set.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123815 entries, 0 to 123814
Data columns (total 6 columns):
msno                    123815 non-null object
city                    123815 non-null int64
bd                      123815 non-null int64
gender                  63206 non-null object
registered_via          123815 non-null int64
registration_init_time  123815 non-null int64
```

The only column with missing values is gender. I will hold off filling in missing values until I build the model because the exploratory data analysis should be based off the data provided.

Another problematic column is age (bd). The age column contains outliers that have negative values and positive values larger than 100.



Outliers under 1 year old and over 100 years old were removed, leaving only 63,188 data points. Although more than half of the data points were removed, those outliers heavily skewed the mean. It is more accurate to only keep data that makes logical sense.

Transactions Data

The resulting transactions data has 1,909,676 rows, at least 1 per user. The transaction records in the data spans over 3 years (2015/01/01 to 2017/02/28). Oddly, 'payment_plan_days', 'plan_list_price', and 'actual_amount_paid' had minimum as 0. However, after further investigation, it seems like transactions with 0 as 'payment_plan_days' are all back in 2015. They might have not had payment plan day options back then. As for the plan price and actual amount paid with 0 values, these transactions does not seem to be a temporal issue since the most recent occurrence happened 2017-02-02. This might just mean that the user received a free trial.

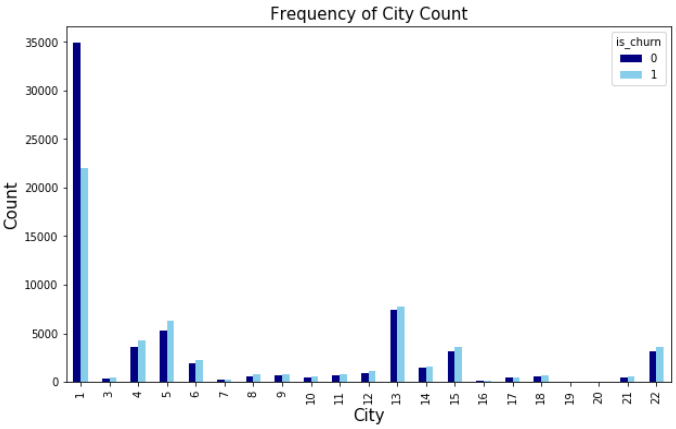
User Logs Data

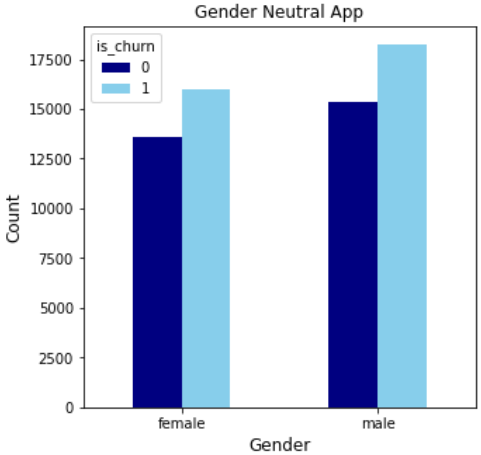
The resulting user logs data has 10,927,697 rows, and it does not have activity logged for all users in the train data, only 115,730 users. The activity log also spans over the same time period as the transaction data (2015/01/01 to 2017/02/28). For some reason, there are 1,564 rows in which total seconds played had negative values. This occurred for data between 2015/04/22 and 2016/04/18.

Exploratory Data Analysis

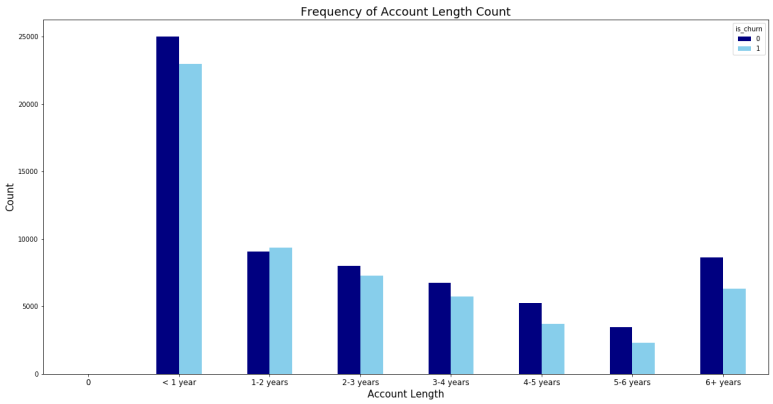
The exploratory data analysis focuses on comparing different variable distributions between the churn group and the group that did not churn to see if certain distributions differ between the 2 groups.

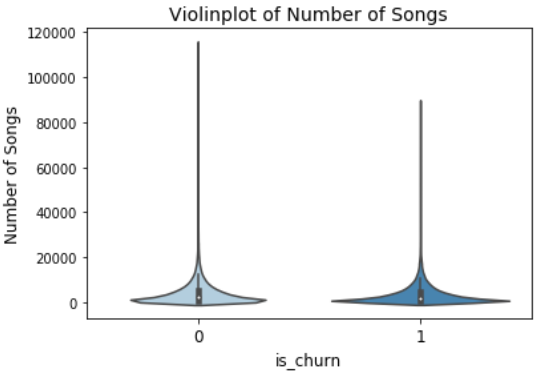
The two groups are actually pretty similar in demographics. Users in both groups are mostly in city labelled as "1". I would think that it represents Taipei. Also, there's an even ratio between males and females who use the app.

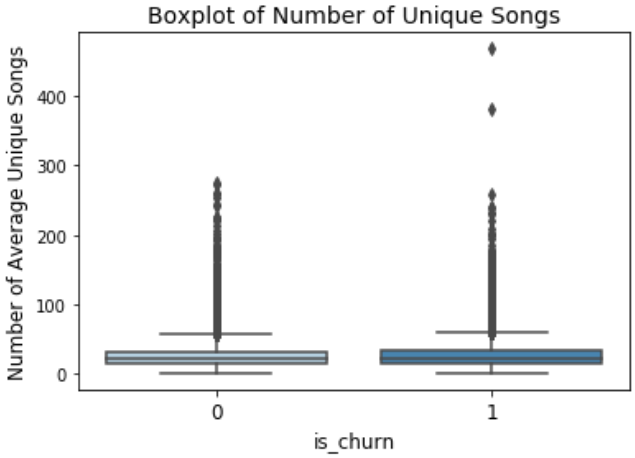


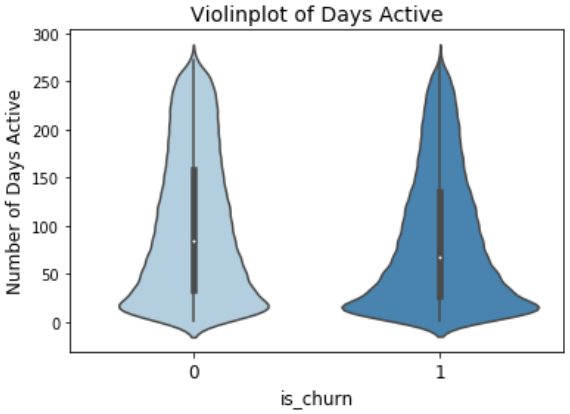


Most users are new and have had accounts for less than a year. Also, on average, they seem to be active on similar amount of days and listened to the same amount of songs.

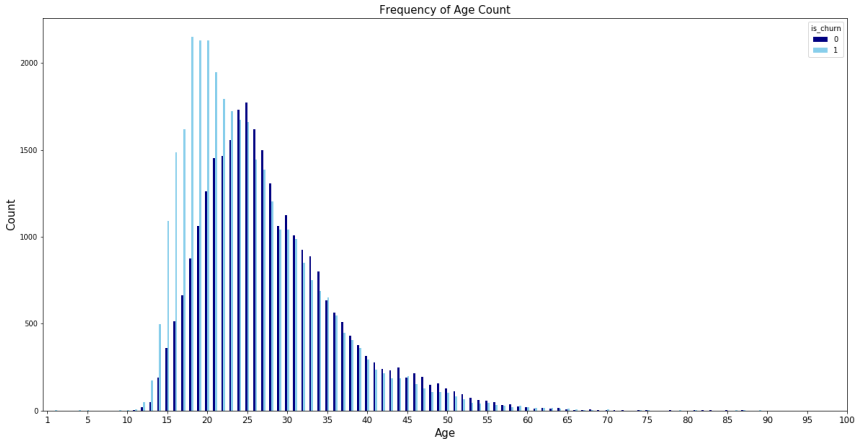




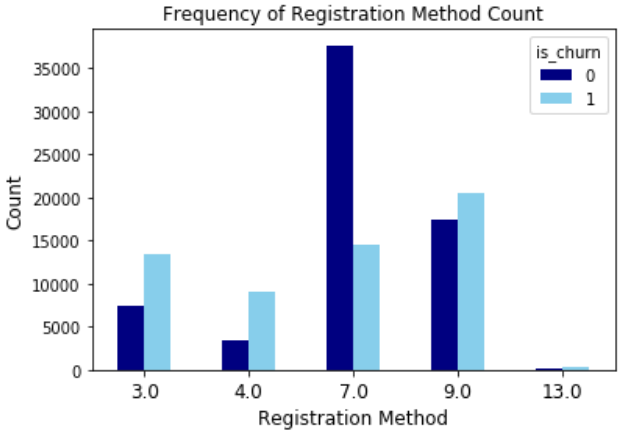




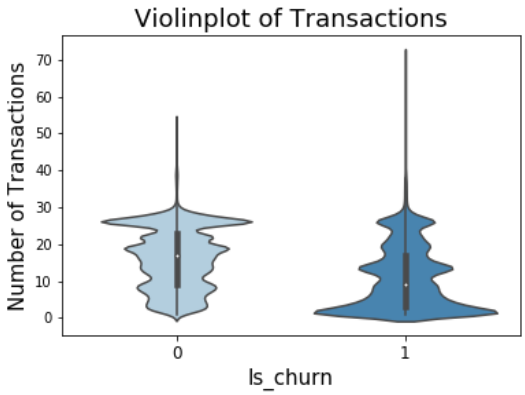
However, there are a few variables with distributions that seem to differ between the churn group and did not churn group. First off, the churn group seems to be comprised of younger users. Most of churned users are in their early 20s while users who did not churn are mostly in their late 20s.

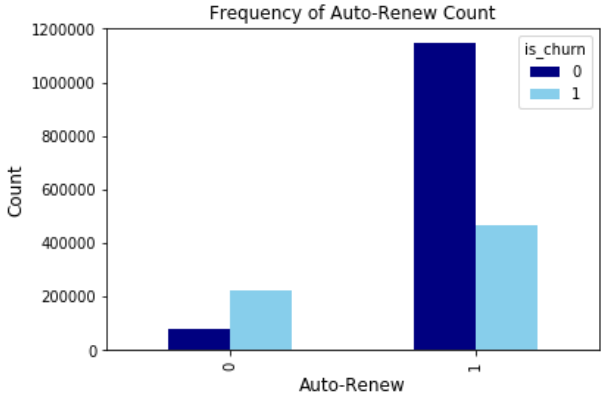


The method they used to register also differed. Users who did not churn mostly signed up via registration method 7 while most churned users registered via method 9. Unfortunately, we do not know what these numbers actual represent.



The number of transactions made also seemed to differ. Those who did not churn seemed to have made more transactions than those that churned. Also, from those transactions, the users who used the auto-renew feature tend to be less likely to churn.





Hypothesis Testing

Although the distributions may seem to look like they differ visually, these hypotheses should be tested to see if the difference is statistically significant. I have tested 5 different variables to see if these differences happened by chance or if they can actually explain chances of churning.

The first hypothesis is regarding the age distribution.

H_0: There's no difference in the age between user who churned and did not churn.

H_A: Users who did not churn are older than users who churned.

```
Mean Difference: 2.5739036785245375
Margin of error: 0.13734545113421098
95% Confidence Interval: [2.4365582273903263, 2.7112491296587486]
```

There is a mean difference of 2.574 in age between the group that churn and did not churn. The confidence interval does not contain 0; therefore, we are 95% confident that there is a difference. After conducting a bootstrap test, the null hypothesis was rejected due to a p-value

of 0. This means that there is an age difference between the two groups, and that younger users are more likely to churn.

The next hypothesis is regarding the distribution of users who use the auto-renew feature.

H_0**:** There's no difference in auto-renewal rate between user who churned and did not churn.

H_A**:** Users who did not churn have a higher auto-renewal rate.

```
Mean Difference: 0.44100533658159546
Margin of error: 0.004210435379622165
95% Confidence Interval: [0.4367949012019733, 0.44521577196121764]
```

There was a difference of 0.441 in mean auto renewal rate between those that churned and did not churn. The confidence interval is very narrow [0.437, 0.445], meaning it is not likely that there is no difference in auto renewal rate. The bootstrap test had a p-value of 0; therefore, the null hypothesis was rejected. There is statistically significant difference in auto renewal rate between the churn and not churn group. On average, those who did not churn were 44% more likely to have their subscription auto-renewed.

I also tested the hypothesis regarding the number of transactions made.

H_0: There's no difference in the number of transactions between user who churned and did not churn.

H_A: Those who did not churn made more transactions than those who churned.

```
Mean Difference: 5.5200867432370675
Margin of error: 0.08865758164061524
95% Confidence Interval: [5.431429161596452, 5.608744324877683]
```

On average, those who did not churn made 5.52 more transactions than those that churned. The 95% confidence interval is [5.431, 5.609]. The bootstrap test resulted in a p-value of 0.0, meaning that the null hypothesis can be rejected. Therefore, we can say that there were more transactions made by those who did not churn.

Although the boxplots showed that the distributions for listening behavior seemed to be similar in the two groups, I still wanted to perform hypothesis testing to see if a difference in the mean was significant.

One hypothesis I tested was regarding the total number of seconds listened.

H_0**:** There's no difference in the total number of seconds listened between user who churned and did not churn.

H_A**:** There's a difference between the groups.

```
Mean Difference: 21891093981546.016
Margin of error: 18958175174543.344
95% Confidence Interval: [2932918807002.672, 40849269156089.36]
```

** Note: Absolute value was used to compute difference*

There seems to be a difference in the total number of seconds listened. After conducting the bootstrap test, we fail to reject the null hypothesis at 0.01 significance level. This means that number of total seconds listened may not differ between those that churned and did not churn.

The last hypothesis tested was regarding average number of unique songs listened per session.

H_0: There's no difference in the number of unique songs listened per session between user who churned and did not churn.

H_A: There's a difference between the groups.

```
Mean Difference: -1.093376972354637
Margin of error: 0.2055256639861163
95% Confidence Interval: [-1.2989026363407532, -0.8878513083685207]
```

On average, those that churned listened to one more unique song per session than those that did not churn. The confidence interval does not contain 0, so we are 95% confident this finding is significant. After conducting the bootstrap test, we are able to reject the null hypothesis since we received a p-value of 0.0.

Out of the 5 variables we've tested, 4 of the variables were proven to have statisically significant differences in distribution. These variables should be included in the predictor model:

- Age
- Auto-Renew
- Number of Transactions
- Number of Unique Songs

Classification Model

To take advantage of the temporal aspect of the data, we will split the dataset into 2 separate time periods for training and testing data:

- Training Data: February (2017/02/01 - 2017/02/28)
- Testing Data: March (2017/03/01 - 2017/03/31)

Just like the training data, the testing data is very biased (~9% churn rate). I have downsampled the majority group so that the testing data has about a 50% churn rate.

Feature Engineering

Categorical features need to be numerical in order to be incorporated into the model. The following features were transformed into dummy variables. The first categorical class of every category was dropped due to the issue of perfect collinearity.

- Members Data Set
    - City
    - Gender
    - Registration Method
- Transactions Data Set
    - Payment Plan (7 or 30)

Temporal features needed to be transformed so that it is relative to the time period being modelled. I will use the relative refactoring method,which involves mapping a date-driven feature into a new feature space which is not anchored in any particular time-series, but is instead relative to a selected point in time. This ensures the feature is relatively comparable across different time periods. I have

chosen to compute the difference in days between the date element and the end of the time period. For example, the training data has information for those that churned in February; therefore, the end of the period is 2/28/17. I have created 2 features using this method:

- Days since registered
- Days since last logged in

For the transactions data and the user log data, there are multiple lines per user according to how much they interact with the service. I found the average of these values for each user. For the transactions data, I am interested in the ones 3 month prior to the month of interest. As for the user log data, I am only interested in the activity for the month prior.

In the end, I have a total of 48 features to be used in the model.

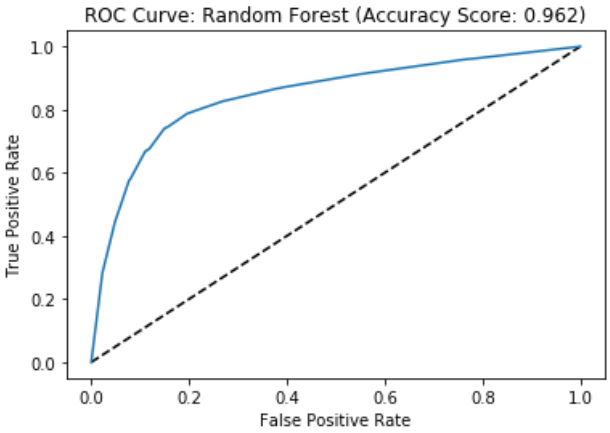
Data Preprocessing

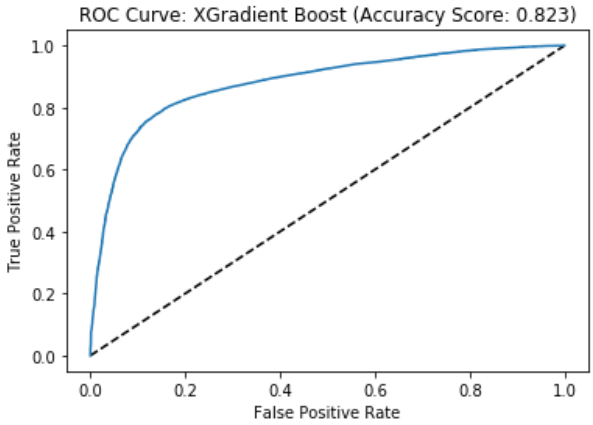
Once I began merging all the data sets together, I realized that we did not have all the information for those we had the churn data for. I had kept all the users that had a record of their account information. If the user did not have any transactions data, the transactions count feature was filled with 0. If the user did not have any user log data, the days since log in was filled with 31. All the other features were just the average of of all records on file; therefore, the users without any transactions or user log data was filled the sample mean for the column. Since the values for each feature vary in range, all the data was normalized prior to fitting the model.

Building the Model

I had fit 5 different classification models on the data using the default parameters to see which classifier was the most effective. Log loss was used as the model metric because not only do I care if the predictions are accurate, but I also care about the confidence in the labels produced. Log loss penalizes errors that are correct but not confident as well as incorrect but confident. The following are the results from the 5 fold cross validation:

Classification Model	**Average Log Loss**
Logistic Regression	0.463
Random Forest	1.464
Adaptive Boosting	0.681
Gradient Boosting	0.428
Extreme Gradient Boosting	0.427





As shown by the graph and results, Random Forest had an extremely high log loss, but performed the best in terms of accuracy (log loss: 1.464, accuracy score: 0.962). Extreme Gradient Boosting, on the other hand, had the lowest log loss of 0.427 and an accuracy score of 0.823. Even the area under the ROC Curve is higher for the model that uses the Extreme Gradient Boosting algorithm.

I will continue to refine the model that applies the Extreme Gradient Boosting algorithm as well as the Logistic Regression. The model with Logistic Regression resulted in a log loss score of 0.463, which is very close to the I have fine-tuned the hyperparameters of the Extreme Gradient Boosting to finalize the model. After performing a grid-search cross validation on a range of values, the following had proven to be the best parameter: colsample_bytree = 0.8, learning_rate = 0.05, and max_depth = 8. This model results in 83.9% accuracy and a log loss of 0.388, which is a slight improvement from the default model. As for the model with the Logistic Regression algorithm, the grid search cross validation selected the following as the best parameter: penalty = 'l2', dual = False, C=1, and max_iter = 100.

It is interesting to see which features have the most effect on the churn rate. Here are the coefficients for the features with the highest effect on the churn rate:

Feature	Coefficient
avg_is_cancel	0.905624
avg_payment_plan_7	0.362903
avg_discount_received	0.165612
avg_is_auto_renew	-0.947160
trans_count	-0.848782
log_count	-0.192631
registered_via_7	-0.145023

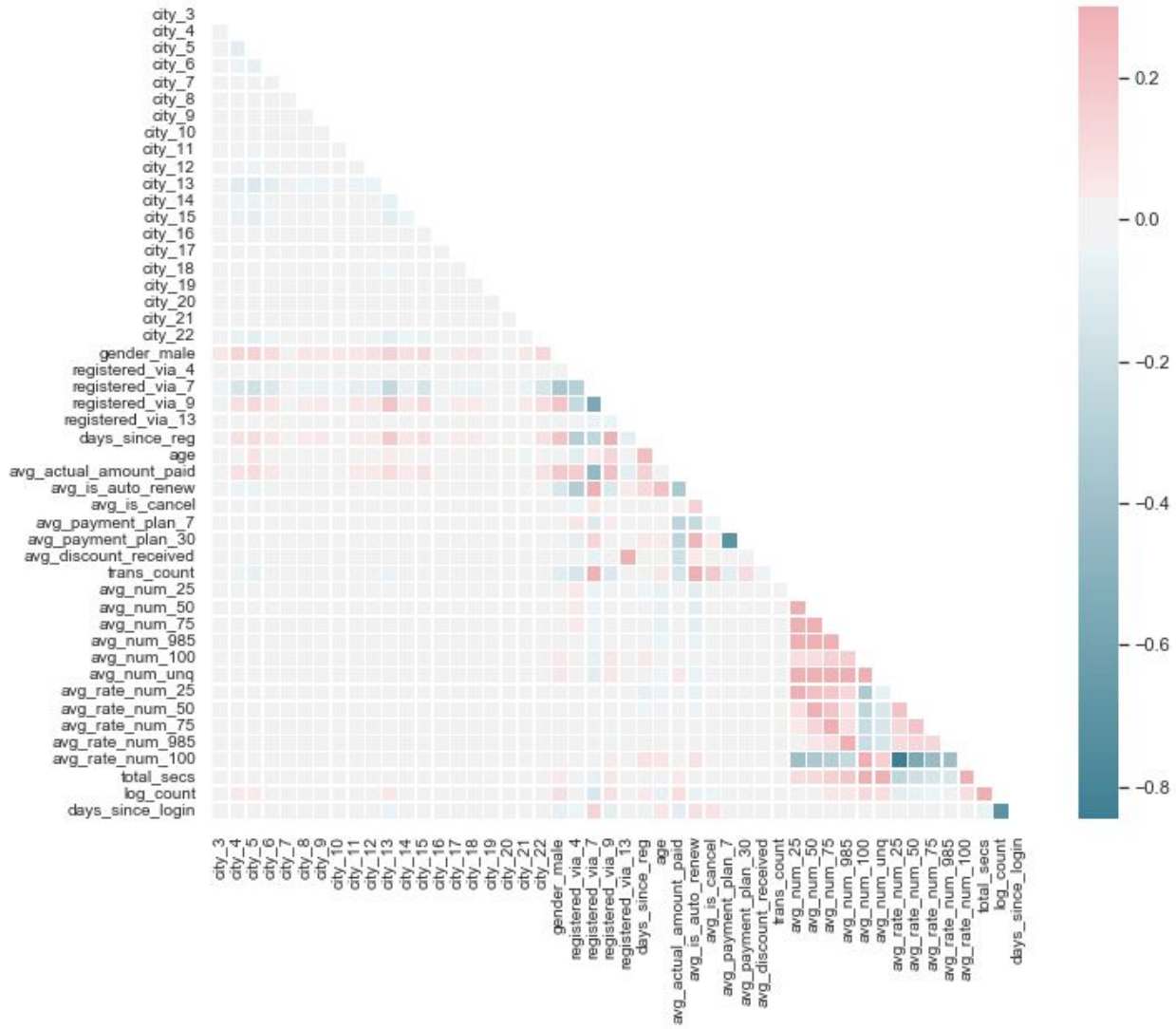
The features with the highest effect mostly seem to relate to the transactions dataset. The average rate of auto-renewal seems to explain the likelihood of churning the most with a coefficient of -0.947. Those that opted to use the auto-renewal feature are less likely to churn. The average cancellation rate also has a high effect on the churn rate since the coefficient of 0.9056 suggests that those who made a cancellation transaction are more likely to churn. The number of transactions also has a high effect on the churn rate; it seems as if the more number of transactions made, the less likely the user would churn (coefficient of -0.849). Interestingly, the effect of having a payment plan of 7 days also had a decent effect on the churn rate. those who had a payment plan of 7 days were more likely to churn. The registration method also seemed to matter since those that registered via method 7 were less likely to churn.

Feature Selection

Feature selection is the method of selecting the most important features to be used in the model. There are multiple ways to perform feature selection:

1. Drop one of the variables that is highly correlated to another
2. Use the recursive feature elimination method to see dropping which features would affect the model the least
3. Perform feature importance to see which features are the most important for the model

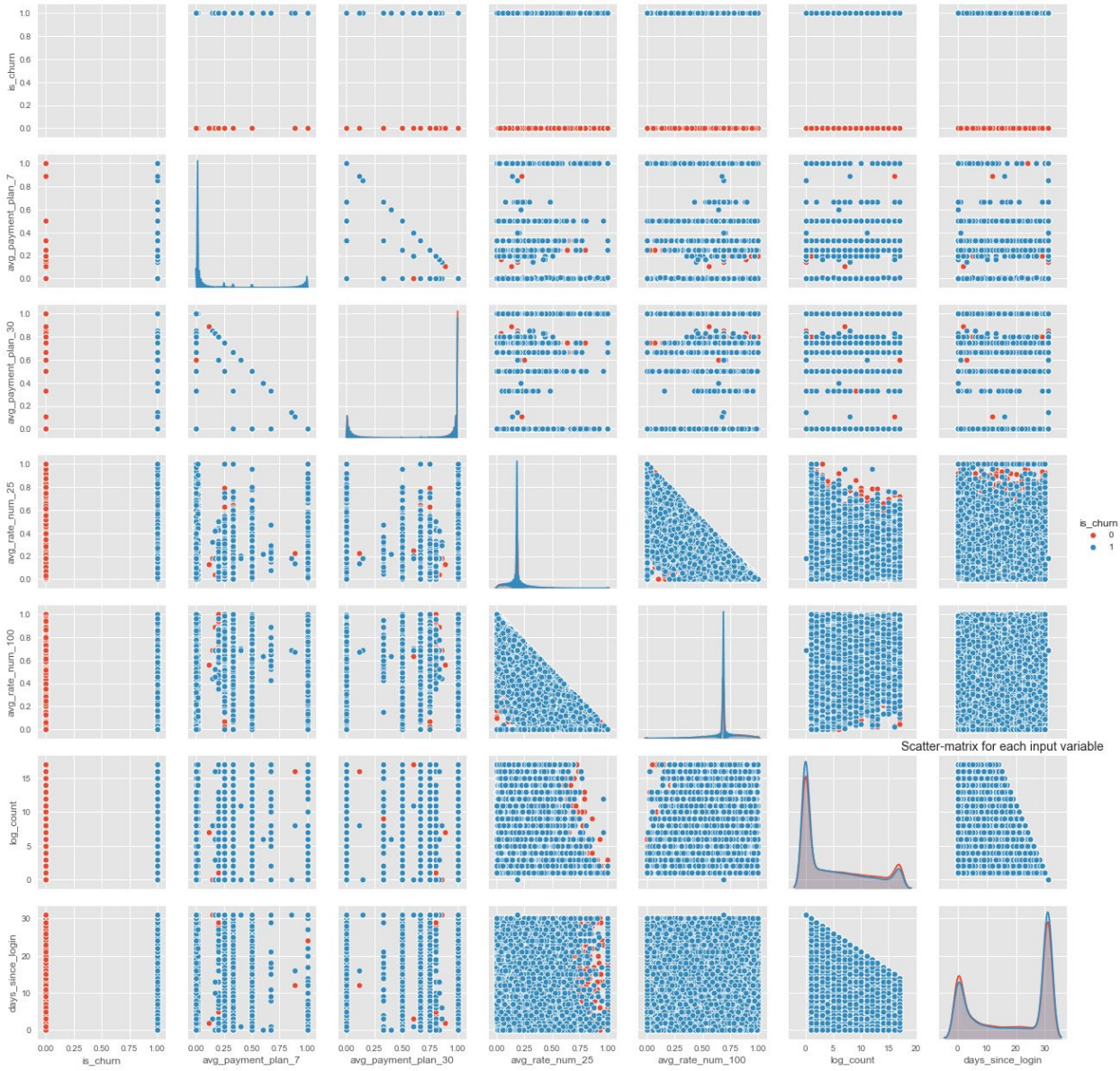
The first method I have performed is dropping variables highly correlated to one another (multicollinearity). These variables supply the same information; therefore it is redundant to keep both. Because they supply redundant information, removing one of the correlated factors usually doesn't drastically improve the model.



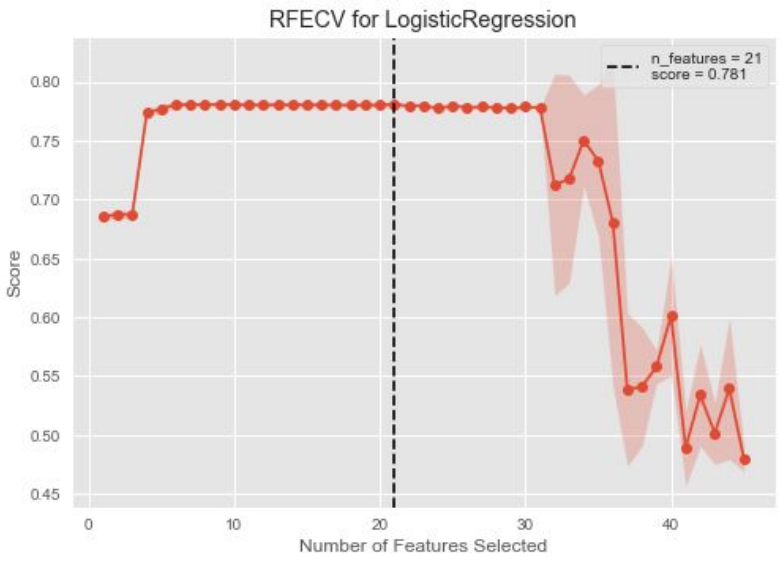
As shown by this pairwise scatter plot, there are 3 pairs of variables that are highly negatively correlated.

- avg_payment_plan_30 and avg_payment_plan_7
- avg_rate_num_25 and avg_rate_num_100
- log_count and days_since_login.

The pairplot graph below illustrates this negative correlation. While there is no pattern between the other variables, the downward triangles can be seen for these 3 pairs of variables. To select which one of the variables in the pairs to remove from the data, I tested and dropped each variable to see which model would have a better log loss score. As a result, dropping 'avg_payment_plan_30', 'avg_rate_num_25', and 'days_since_login' produced a log loss score of 0.463304.



The second method is the Recursive Feature Elimination (RFE), which recursively removes features, builds a model using the remaining attributes and calculates model accuracy. RFE is able to work out the combination of attributes that contribute to the prediction on the target variable (or class). Below, I have graphed the effect on the model when keeping only a certain number of features.



As the graph suggests, the ideal number of features to keep is 21. The following are the 21 features to be kept:

Column name	Keep
city_7	True
city_8	True
city_9	True
city_10	True
city_16	True
city_18	True
city_19	True
city_20	True
registered_via_7	True
registered_via_9	True
registered_via_13	True
avg_is_auto_renew	True
avg_is_cancel	True
avg_payment_plan_7	True
avg_payment_plan_30	True
trans_count	True
avg_rate_num_25	True
avg_rate_num_50	True
avg_rate_num_75	True
avg_rate_num_985	True
avg_rate_num_100	True

A majority of the features kept are regarding the demographics of the user. The city the user lives in as well as their registration method mattered since a good chunk of these dummy variables were kept. The other half of the features is composed of the the nature of the user's transactions (type of payment plan, number of transactions, auto-renewal, cancellation) and the average rate of how much of the song the user actually listens to. The number of active days did not seem to make it onto the list. The log loss score after dropping these variables was 0.468988.

The last method used was measuring feature importance. The scoring method used to rank each feature is Mutual Information. Mutual information measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency. Only features with low mutual information should be included in the model because if knowing one of these variables reduces uncertainty about the other, there is no need to include both variables. Only the variable that explains more of the target variable should be kept.

Number of Top Features Kept	Log Loss Score
18	0.4674
20	0.4642
24	0.4634
28	0.4634
30	0.4634

According to the results, we would only keep the top 24 features to be used in the model. It is where the log loss score is at the lowest and starts to plateau. These 24 features and their importance scores are as listed below:

Feature	MI Score
avg_is_auto_renew	0.190177
trans_count	0.145051
avg_actual_amount_paid	0.123797
avg_is_cancel	0.088513
avg_payment_plan_7	0.075063
avg_discount_received	0.064310
registered_via_7	0.061058
age	0.023075
registered_via_4	0.020251
avg_payment_plan_30	0.015818
days_since_reg	0.013782
registered_via_9	0.011532
gender_male	0.010578
city_14	0.005798
city_5	0.005577
city_4	0.005511
log_count	0.004673
city_15	0.004220
city_13	0.004218
city_12	0.004160
avg_num_985	0.004034
city_17	0.003831
avg_rate_num_25	0.003822
city_6	0.003816

The top 6 features all seem to relate to the details of the transactions the users have made. The most important feature is the average auto renewal rate. As shown in the exploratory data analysis, most users who did not churn opted for the auto-renewal feature. The features kept due to feature importance is very similar to the features kept due to recursive feature elimination. They are comprised of features related to transactions details and user demographic information. The user activity does not seem to be that important as only 3 of the features are from the user activity dataset. The log loss score is slightly better through this method with a log loss score of 0.46393.

Principal Component Analysis

Another way to improve the model is through adding more variance into the model. Principal Component Analysis (PCA) linearly transforms the features into a lower dimension while keeping the most important components of the feature. The larger the variance the larger the amount of information the variable contains. Since the Recursive Feature Elimination method and the Feature Importance both removed many features from the dataset, the variance in the dataset was reduced. I have added the top 2 principal component into each of the features of the dataset to see if they would result in a better log loss score. The results are as followed:

Feature Reduction Method	Log Loss Before PCA	Log Loss After PCA
Recursive Feature Elimination	0.4690	0.4677
Feature Importance	0.4639	0.4637

Although the adding the principal components did improve the log loss score for both feature reduction methods, the feature importance method still beat the recursive feature elimination method; therefore, the final model will utilize the features kept through that method.

Results of the Final Model

The final model uses the top 24 most important features as well as the 2 principal components to train on. The following are the results on the test data and train data:

Algorithm Used	Train Data Results	Test Data Results
XG Boosting	Accuracy score: 0.8367 Log Loss: 0.3953	Accuracy score: 0.8027 Log Loss: 0.4559
Logistic Regression	Accuracy score: 0.8129 Log Loss: 0.4637	Accuracy score: 0.8016 Log Loss: 0.4718

As seen, the score for the test data is slightly worse than the train data. However, this difference is expected when applying models to test data. It is good to know that the model did not overfit to the training data.

Limitations and Further Opportunities

There are a few limitations in the model that could be improved if someone else were to replicate this project. The model could be vastly improved if the analysis was done on a computer with higher CPU. An alternative to balancing out the biased data would be keeping all the data points and putting higher weights on the minority group (in this case, the churn group). Also, fine tuning more hyperparameters of the classifiers might better fit the model and result in lower log loss. More feature engineering could be done given more guidance on how the values of certain columns were derived. There were lots of outliers that did not seem to make sense. For example, extreme negative values when the value should strictly be positive such as age and total seconds listened.

A model with a low log loss allows accurate prediction of the likelihood of churning. Therefore, as user behaviors start to yield a high probability of churning, KKBox can offer promotions proven to stir behaviors back to user behaviors with low likelihood of churning. Clustering similar users together in terms of demographics and activity can pave the opportunity of a recommender system of which promotion would be most effective.