**Background**

The problem to be solved in this project is how to accurately predict user churn for the KKBox subscription service. Churn refers to when a customer has discontinued a subscription service. For a subscription based business to be successful its churn rate has to lower that its rate of new customer subscriptions. KKBox is a music streaming app in Taiwan and surrounding regions. KKBox presented this as Kaggle competition and provided user data to do so. Specifically, they want to forecast if a user make a new service subscription transaction within 30 days after the current membership expiration date. Predicting the future is never easy but with such a large sample of user data it should be possible to use a machine learning algorithm to classify churners and non-churners using insights from their user data. The evaluation metric for this challenge is the log loss of the predicted probability of churn for all the users in the test set. An accurate model will help the business identify customers who are likely to churn. This is valuable to the marketing team for applying targeted customer retention efforts and to analyze what factors are causing customers to churn.

**Description Of Data**

The data provided by KKBox consists of 5 csv files labeled: train, members, transactions, users logs and sample submission. The train file has two columns, the anonymized user id, and a binary ‘is churn’ value. There are about 993 thousand rows/users in the train file and they were all selected as having memberships that expired in February 2017. All other information about the users is jumbled somewhere in the members, transactions and user logs files. The sample submission file (evaluation set) is 970,000 rows/user who have been selected for having subscriptions that expire in March 2017. So correctly stated, the train set is for user churn in March 2017, and the test set if for user churn in April 2017.

The members file has almost 6.8 million rows/users and 5 columns of information about each anonymized user id. These columns are city, age, gender, registration method and date of initial registration. About 4.5 million of these users have NaN values in the gender column. There are also about 4.5 million users with a value of zero in the age column and well as about 6,000 outliers that make for nonsensical ages such as -5 or 873.

The transactions file does not have one row per unique user. It is a 21.5 million row by 9 column csv file that has about 2.4 unique users. Each row is a specific transaction that one user made and since KKbox subscription typically renew monthly there is a transaction row for each month the user has been a member. Columns of the transaction file include: payment method id, payment plan days, plan list price, actual amount paid, ‘is auto renew’ binary values, transaction date, membership expire date, and binary ‘is cancel’ values.

Each row of the user logs file contains data about one day of one user’s usage of the KKBox app, so it turns out to be a massive file that is difficult to work with. Even the transactions file can present memory problems at times at a size of 1.6 GB, so the user logs file at size of 28.4 GB certainly requires special treatment. The simplest work-around is to just take the first X million rows of the user logs file and work with that data. The information in the columns of the users log file includes counts of songs listened to grouped by percentage of song played, number of unique songs played, and total seconds played for that day.

So the scope of the user information available both seems somewhat limited but also overwhelming in quantity and shape. Perhaps KKbox has other data available about its users such as what variety of music users listened to, but they did not present that and it would be proprietary so my analysis will be limited to these datasets. Instead of looking for other data I focused on organizing what I have and subsetting it into a usable size dataframe.

An approach that I took was to focus the user log data instead of the transaction data. The rationale being that it should be easier to discern differences in the listening behavior of churners vs non-churners and that listening behavior would have more predictive power. I tried cleaning and wrangling the data in a variety of ways and the latest iteration went as follows in the next paragraph.

First, I read the members csv and apply a boolean filter to exclude members with nonsensical ages. Then merged the members and train set with an inner join on the user id column to create a dataframe called df. Then I read the first 40 million rows of the user logs csv and selected only the entries for users that existed in df. I changed the string values in the date column of the user log into datetime objects so I could filter out specific date ranges of logs. Then as a shorthand way to limit memory usage I saved that abridged version of the user logs as a csv that could be opened in a new notebook with a fresh kernel.

In order to convert the daily user logs into a form that can be used with some of the sklearn machine learning algorithms, ie. one row for each user with multitude of feature columns, I used the group-by method on each data column, grouping by the user-id and using the mean as the aggregation function. So this turned the daily user behavior into the average daily user behavior for whatever time period I filter for. I then took the ‘mean user logs’ then did another inner merge on user id with df. Since this resulting dataframe a few column of categorical data I used the pandas ‘get dummies’ function to convert then into additional columns with binary values.

**Initial Findings**

The data was then ready use to try build a predictive model. Since the goal was to predict a binary outcome I thought from the beginning that this would call for logistic regression. But first I tried using k nearest-neighbors to see how it would do and used the train-test-split function on this data frame for evaluation. T KKN model was unpromising as it resulted in ROC AUC scores of about 0.52-0.55 and a log loss close to 1.

Next I tried a logistic regression model and it seemed to do a bit better with ROC AUC scores around 0.62-.064 and a log loss around 0.22. I should be able to improve upon just trying a few different approaches in data wrangling and feature engineering. I also tried a random forest model and got ROC AUC of about 0.57 and a log loss around 1. My mentor advised me that one of the most important things to look out for in building machine learning predictive models is to avoid multicollinearity between features. To try to do this a variance inflation factor (VIF) can be calculated for the features using available software in python. Eliminating features with high VIF did not seem to improve ROC AUC score or log loss but should help to ensure robustness of the model.