KKBOX CHURN PREDICTION

OBJECTIVE & EVALUATION

- **Objective:** Predicting whether a user will churn after the subscription expires.
- **Evaluation:** The evaluation metric for this project is Log Loss

$$\textbf{\textit{log-loss}} = \frac{-1}{N}\sum_{i=1}^{N}(yi\ log(pi) + (1 - yi\ )log(1 - pi))$$

Where ***N*** is the number of observations, $\boldsymbol{y_i}$ is the binary target, and $\boldsymbol{p_i}$ is the predicted probability that y_i equals 1.

CLIENT & BUSINESS-IMPACT

- **Client: KKBOX** is Asia's leading music streaming service, holding the world's most comprehensive Asia-Pop music library with over 30 million tracks, supported by advertising and paid subscriptions.
- **Business Impact:** Since the music streaming service providers are becoming more competitive day by day one of the major problem these companies are facing is customer retention. Attracting new customers is much more expensive than retaining existing ones. Hence this predictive model helps the business by predicting the customers who are going to churn.

DATA-SET

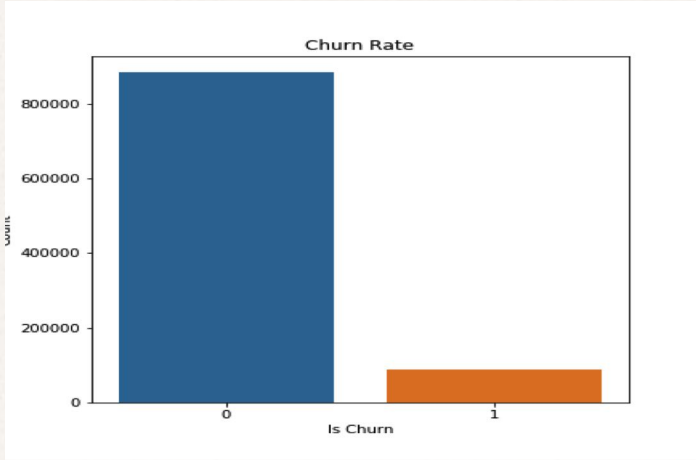
- For this analysis, I will be using the KKBox's churn prediction dataset which was publicly available on Kaggle. Data is distributed across 4 different csv files as follows:
  - **1. train_v2csv:** The train set, containing the user ids and whether they have churned.
  - **2. transactions.csv:** The transaction set contains all the payment details till feb-2017.
  - **3. transactions_v2csv:** The transaction set contains all the payment details of march-2017.
  - **4. members.csv:** The members set contains the user information.
  - **5. user_logs.csv:** The user logs set contains the daily user logs describing listening behaviors of a user for the month march-2017.

MEMORY REDUCTION

- Because all the data frames are very large in size and are using a lot of memory, we performed memory reduction to reduce the memory usage by changing the data types of some columns and splitting the date column into three different columns - year, month & day columns respectively.
- **Members** data set memory-usage has been reduced from 310 MB to 155 MB.
- **Transactions** data set memory-usage has been reduced from 1756 MB to 723 MB.
- **User-logs** data set memory-usage has been reduced from 1265 MB to 597 MB.
- **Train** data set memory-usage has been reduced from 15 MB to 8 MB.
- We have reduced the memory of each file to its half making it convenient for the later analysis.

DATA EXPLORATION IN TRAIN

- **Churn Percentage:** In the train data frame, we have the 'msno' – which are unique id no's for given to each customer & their churn detail are present.



- Only **9%** people have churned which looks so successful, making it a highly imbalanced classification problem.

DATA EXPLORATION IN MEMBERS

- **City:** There are total of 21 cities, there is no city '2'. We observe majority from city 1. Everything else looks similarly unpopular.
- **bd (age column):** In the bd (Age) column we observed it has lot of values set to 0 and there are some outliers ranging from '-7168' to '2016'.we find that younger users on average appear to be more likely to churn.
- **Gender:** Around 60% of the data is missing after the merge. With the data we have it seems both male and female are churning quite similar. We have to see how to deal with the missing values in future analysis.
- **Registered-via:** There are 5 classes ('3', '4', '7', '9', '13') listed as registration method. Method '7' appears to correlated with the most loyal users, while method '4' has slightly higher churn rate of all.
- **Registration & churning trends yearly & monthly:** we observed that popularity started rising slowly after 2009 and it started to increase strongly from 2012. Registrations are high during the year end and starting months.

DATA EXPLORATION IN TRANSACTIONS

- **Payment method-id:** There are 40 payment methods (method ‘9’ is missing) and the payment - method '41' is by far the most popular one.
- **Payment plan days:** The payment plan duration categories show strong differences in churn percentage. The lowest churn numbers (around 5%) are associated with the 30-days, 31-days, and the 0-day memberships (surprisingly). The churn percentage for next widely used plans 7-days is 35% & 410-days is 63%.
- **Plan list price & Actual amount paid:** The overall distributions of planned vs actual payment are very similar, even though there are some differences in price paid to the list price.
- Interestingly here, in most of the cases the users ended up paying more.
- But there isn't any surprising trend in churn rate of users who paid more.

DATA EXPLORATION IN TRANSACTIONS

- **Auto-Renew:** The vast majority of users have automatic renewal of their subscriptions enabled and users who did not choose to auto renew were clearly more likely to churn.
- **Is-Cancel:** Cancellations are high in the months of dec, jan, feb & mar. Rest all the months seem very similar.
- Not all users who cancelled their subscription are churning. Majority of the cancelled users are re-subscribing within a month.

DATA EXPLORATION IN USER-LOGS

- Here we observed that users with less life span are slightly churning more.
- It is also clear that Churn users are listening to less number of songs when compared to No-Churn users.
- But we observe that both the churn users & not churn users log-ins are very similar.

FEATURE ENGINEERING

- Feature Engineering is the process of creating additional relevant features from the existing raw features in the data, and to increase the predictive power of the learning algorithm. Here are the features that are created.
- **trans_count**: Count of number of transactions for each user.
- **transaction span**: Sum of payment plan days of all the transactions for each user.
- **total_list_price**: Sum of the listed price in all the transactions for each user.
- **total_amount_paid**: Sum of the amount paid in all the transactions by a user.
- **difference_in_price_paid**: Some of the users have paid more or less amount than the list price. So, this feature calculates the difference between the list price and price paid for each user.
- **amount_paid_perday**: This feature is created by dividing the total amount paid by transaction span of each user.
- **logs_count**: Count of number of entries for each user in the user-log dataset.

DATA PREPARATION

- we have dropped '**bd**' and '**gender**' features from our final data-frame because of having around 60% null-values.
- All the date columns are transformed into 'year', 'month', and 'day' columns respectively.
- Finally, all the data-frames are joined using left join on 'msno' to the train data-set.
- We have some user entries missing in the user-logs dataset. Because the entire row has missing values here, we are dropping all the rows with missing rows.
- **Label Encoding & Changing data-types**: We are changing the data types of the features making sure the categorical features and continuous features are converted to category, integer & float respectively.
- **Data Splitting:** Finally, before building the machine learning models the dataset is split into 'train' and 'test' sets. Here we have split the train and test sets in 80:20 ratio.

SMOTE

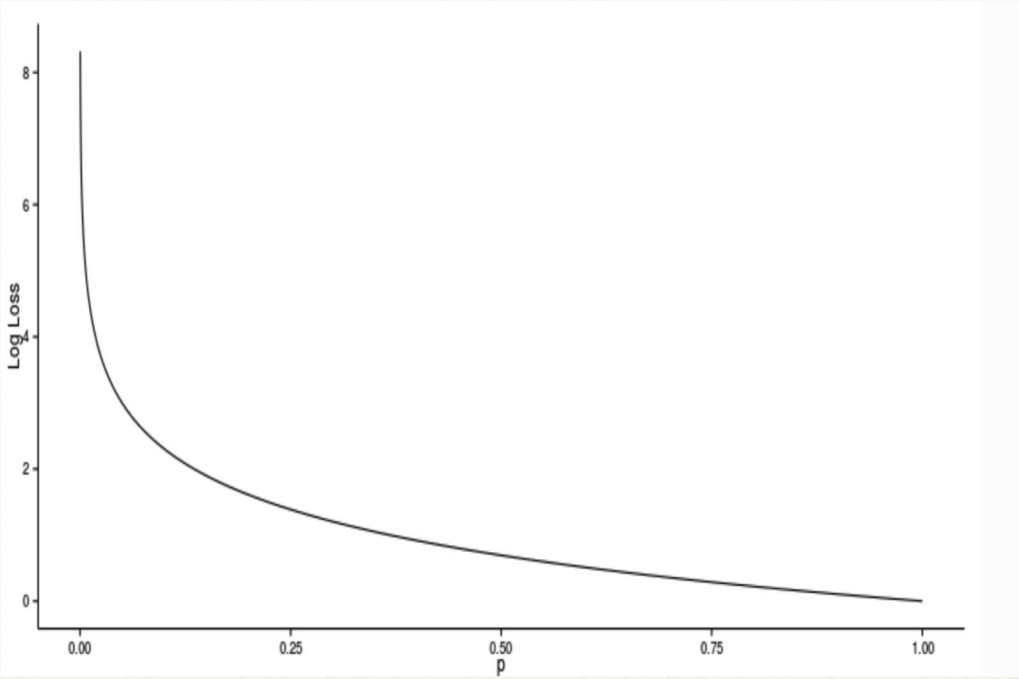
- Synthetic Minority Oversampling (SMOTE) works by creating synthetic observations based upon the existing minority observations.
- Since our dataset is highly imbalanced, we applied SMOTE on the training set.
- At a high level, SMOTE creates synthetic observations of the minority class (Churn-users) by:

1.Finding the k-nearest-neighbors for minority class observations (finding similar observations)

2.Randomly choosing one of the k-nearest-neighbors and using it to create a similar, but randomly tweaked, new observation.

LOG-LOSS

- Logarithmic Loss, is a classification loss function often used as an evaluation metric in classification problems.
- Log Loss quantifies the accuracy of a classifier by penalizing false classifications. Minimizing the Log Loss is basically equivalent to maximizing the accuracy of the classifier.
- A perfect classifier would have a Log Loss of precisely zero.
- The plot shows the Log Loss contribution from a single positive instance where the predicted probability ranges from 0 (the completely wrong prediction) to 1 (the correct prediction).



PREDICTIVE MODELLING

- Implementing machine learning algorithms to build a predictive model. Since our problem is a classification problem first implementing some of the simple classification algorithms & then ensemble methods and evaluate our metric 'log-loss' and see which method results lower 'log-loss' value.

Classifier	log-loss value
Logistic Regression	0.2187796499512514
Decision Tree Classifier	2.0344661505056827
Random Forest Classifier	0.4136045064093898
Gradient-Boosting Classifier	0.11795942846203314
Ada-Boost Classifier	0.653128643077741
XG-Boost Classifier	0.11729961185499269

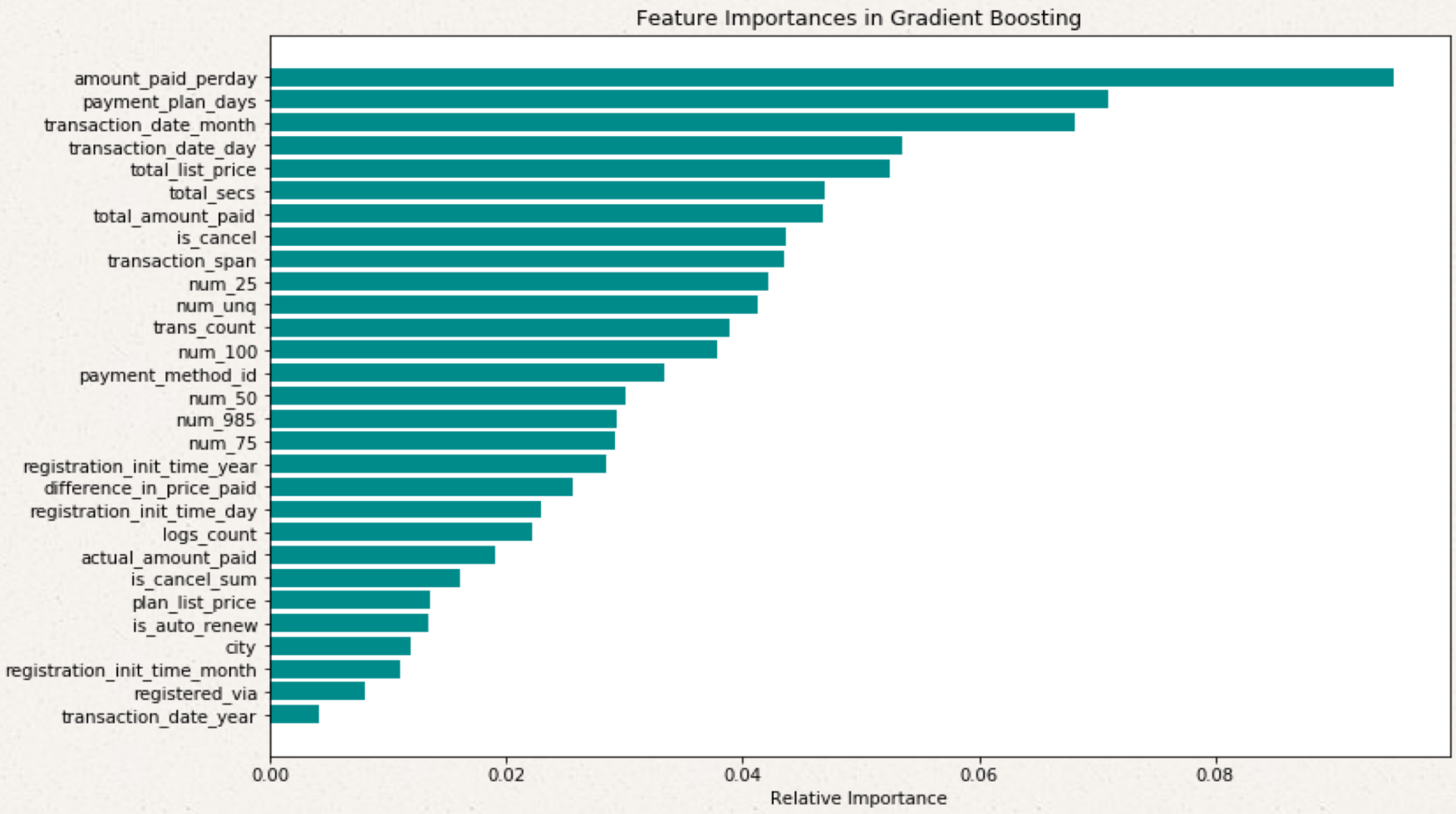
PREDICTIVE MODELLING

- Because, the Gradient-Boosting Classifier & XG-Boost Classifier are returning lower 'log-loss' values we are using them as baseline models and try to tune their parameters to optimize the loss-function and see if the 'log-loss' value gets any better.
- **Parameter Tuning in Gradient-Boosting Classifier:** After tuning some of the important parameters like ***learning_rate***, ***n_estimators***, ***max_depth***, ***min_samples_split***, ***min_samples_leaf***, ***max_features*** & ***subsample*** *we were able to lower the log-loss value.*

Log-Loss	0.1026813547203256

- Though it gets computationally very expensive finally now we can clearly see that this is a very important step as log-loss scored improved from ~0.1179 to ~0.1026 which is a significant jump.

GRADIENT-BOOSTING FEATURE IMPORTANCE PLOT



CONCLUSION

- Finally, we can say that the ‘amount paid per day by each user’, ‘payment plan days’, ‘transaction dates’ are top contributors for user churn.
- We can even try tuning XG-Boost classifier parameters and also include ‘user-log’ file which we couldn’t use in this analysis due to large file size (32 GB) to improve ‘log-loss’ score.