**Capstone 2 Project Final Report**

**"Customer Churn Prediction"**

by Yuriy V. Sereda

Feb. 2022

## 1. Problem Statement

### 1.1 Context

KKBOX is Asia's leading music streaming service, holding the world's most comprehensive Asia-Pop music library with over 30 million tracks. They offer a generous, unlimited version of their service to millions of people, supported by advertising and paid subscriptions. This delicate model is dependent on accurately predicting churn of their paid users.

When users sign up for the service, they can choose to either manual renew or auto-renew the service, and they can cancel their membership at any time. The **task** is to **build an algorithm that predicts whether a user will churn** after their subscription expires. Specifically, we want to forecast if a user will make a new service subscription transaction within 30 days after the current membership expiration date.

### 1.2 Criteria for success

Accuracy of prediction for a trained machine learning model should not deteriorate when applied to new, more recent data. Decision criteria are simple and interpretable. If using a decision tree, no more than 15 questions are asked before making a prediction.

### 1.3 Scope of solution space

Since the majority of KKBox's subscription length is 30 days, many users re-subscribe every month. The key fields to determine churn/renewal are transaction date, membership expiration date, and *is_cancel*. The *is_cancel* field indicates whether a user actively cancels a subscription. Subscription cancellation does not imply the user has churned. A user may cancel service subscription due to change of service plans or other reasons. The criterion of **churn** is no new valid service subscription within 30 days after the current membership expires.

Supervised learning methods will be used to handle this binary classification problem. Some of the records with missing and erroneous data will be placed in a separate category.

### 1.4 Constraints within solution space

- The data may reveal incorrect and missing values. Such problems will be treated individually for different features and depending on the number of problems of specific kind.
- The data is in a shape that is incompatible with machine learning algorithms that expect one record for each prediction label. In particular, there are usually multiple transactions and usage statistics records for each member, which must be combined into a single row.

- When engineering features, an overfitting problem may arise. This would require an information value analysis: features with too high or too low information value must be eliminated from modelling.
- Time cut-off needs to be applied to input data to avoid accessing future data.
- Logistic regression model requires all continuous-value features to be rescaled (standardized or normalized).

## 2. Data Collection and Organization

Data were compiled by Ann and Arden from KKBox, and are available at Kaggle https://www.kaggle.com/c/kkbox-churn-prediction-challenge. Data are provided in three CSV files containing basic information about KKBOX customers, transaction details, and listening statistics.

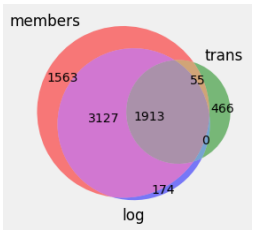
### 2.1 Data summary

- `members.csv`: city, age, gender, registration channel, initial registration time (6658 records).
  - `msno` - unique member ID, string: 6658 members.
  - `city` - categorical: 20 levels (1, 3-18, 20-22).
  - `bd` - age, categorical: 65 levels from 0 to 827. Has two outliers >117, use your judgement.
  - `gender` - categorical: male, female, and 64.9% missing.
  - `registered_via` - registration channel, categorical: 14 levels (1-9, 11, 13, 16, 17, 19).
  - `registration_init_time` - date of initial registation, timestamp: 2004-03-27 to 2017-04-29.
- `transactions.csv`: payment method, duration of subscription plan, listed price, paid price, auto renewal indicator, transaction date, membership expiration date, and the churn label (22940 records).
  - `msno`: unique member ID, string: 2434 members (transactions are available for 36.6% of registered members).
  - `payment_method_id` - categorical: 31 levels (8, 12-41).
  - `payment_plan_days` - plan duration, integer: 26 levels within 0 to 450.
  - `plan_list_price` - numeric: 31 levels from 0 to 2000.
  - `actual_amount_paid` - cost after discount, numeric: 31 levels within 0 to 2000.
  - `is_auto_renew` - binary 0/1: 85.1% of transactions are auto-renewed (1).
  - `transaction_date` - timestamp: from 2015-01-01 to 2017-03-31.
  - `membership_expire_date` timestamp: 1022 levels from 1970-01-01 (invalid) to 2018-06-10.
  - `is_cancel` - is subscription actively cancelled, binary 0/1: 0 - no, 1 - yes (3.95% of transactions are cancellations).
- `logs.csv`: date, number of songs listened below 25%, 50%, 75%, 98.5%, and 100% of their duration, number of unique songs listened, and total duration of listening. 424254 records.
  - `msno`: unique member ID, string: 5214 members (logs are available for 78.3% of registered members).
  - `date` - date of observation, timestamp: 821 levels from 2015-01-01 to 2017-03-31.
  - `num_25` - songs listened 0-25%, integer: 288 levels within 0 to 937.
  - `num_50` - songs listened 25-50%, integer: 137 levels within 0 to 257.
  - `num_75` - songs listened 50-75%, integer: 83 levels within 0 to 204.
  - `num_985` - songs listened 75-98.5%, integer: 109 levels within 0 to 201.
  - `num_100` - songs listened 98.5-100%, integer: 734 levels within 0 to 4376.
  - `num_unq` - total unique songs listened, integer: 386 levels within 1 to 2944.

- `total_secs` - total duration of listening in seconds, floating-point: 409744 levels - 89 missing values encoded as `-9.223372e+15`, 134 small values below 1s, maximum is 2.763295e+06.

All tables have customer ID column `msno`.

### 2.2 Missing and erroneous values




The main dataset contains transaction information that allows to determine churn status. Unfortunately, the transactions, membership, and usage logs have only partial overlaps, as illustrated by the Venn diagram. For instance, only 1968 members out of 6658 have transaction information, and 1913 members have a complete set of records. 3301 of members who have usage logs do not have transaction information, and thus their data could not participate in model training. Membership expiration dates have 610 errors: in 2 cases these dates are earlier than initial registration of the member, and in all other cases they are earlier than transaction dates.

Membership information was right-joined with transaction information, so that all transactions were preserved, and missing membership features were placed in a separate category: missing city, gender, and registration channel values were encoded as 0, and separate category 'unknown' was created for gender. Missing or obviously incorrect membership registration dates were replaced by the corresponding first transaction minus the average duration between initial registration and first transaction dates. This duration was one of the first engineered features used as a predictor.

### 2.3 Pre-processing and Training Data Development

Churn labels for training and testing were generated using transaction history: transactions followed by another transaction within 30 days from the current membership expiration were marked as non-churn (0), and were labelled as churn (1) otherwise.

Based on membership registration and first transaction dates, many useful binary features were engineered for machine learning indicating the weekend, day of the week, month, and season.

The main pre-processing challenge was to enforce compliance of usage log data with the requirement of machine learning methods that there is only one record per classification label. This was achieved by aggregating usage logs for a given customer over the last week, month, and 3 months. The resulting features were the averages of the listening duration per record and of the number of songs listened up to 25% of their duration, between 25% and 50%, 50-75%, 75-98.5%, and 98.5-100% for each of the above three time periods. These aggregated features were merged with the membership and transaction information.

Current transaction dates were used for filtering off future data that could not be used for predictions. The earliest available transaction was on January 7, 2015, the cut-off date was set at October 22, 2016, and the latest transaction was on March 7, 2017.

Upon extraction of useful information, all dates were dropped. One-hot encoding was performed for all categorical features by converting them to binary 0/1 dummy variables.

Features were filtered using Weight of Evidence and Information Value method: non-informative features with IV < 0.01 and too informative ones with IV > 0.8 were dropped. This reduced the number of predictors from 370 to 82. Further filtering was performed based on Variance Inflation Factor < 5: features with too high variance were dropped.

To improve performance of the Logistic Regression model, all predictors were standardized. This technique transforms the data to have a mean of 0 and a standard deviation of 1. Gaussian distribution of the features was used.

## 3. Modeling

### 3.1 Logistic Regression

Logistic regression model is based on a sigmoid function that has values very close to either 0 or 1 across most of its domain.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The interpretation is that the probability that the output for a given **x** is equal to 1 is

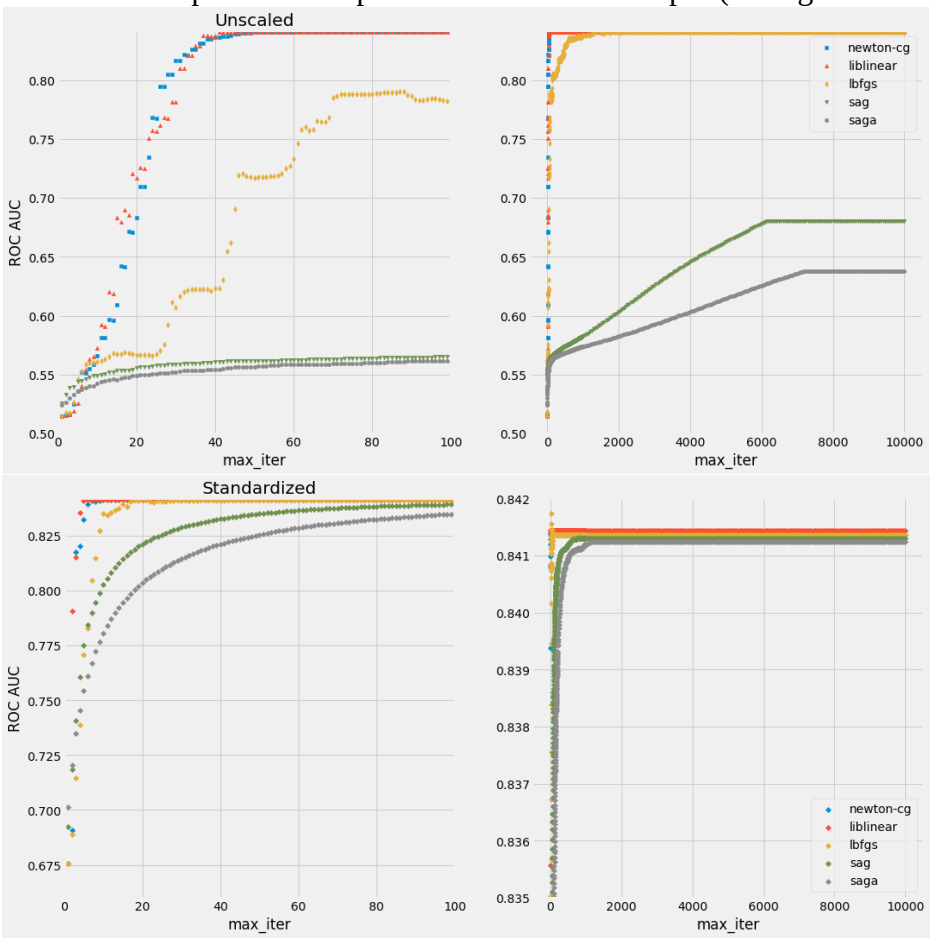
$$p(x) = \sigma\big(f(x)\big),$$

where $f(x)$ is a linear function logit

$$f(x) = b_0 + b_1 x_1 + \cdots + b_r x_r$$

and $b_i$ are the estimators of the regression coefficients, which are also called the predicted weights or just coefficients. For each observation $i = 1, \ldots, n$, the predicted output is 1 if $p(\mathbf{x}_i) > 0.5$ and 0 otherwise. The threshold doesn't have to be 0.5, but it usually is.

This model ranked 2nd from the bottom, exceeding only the decision tree model in Sec. 3.2. Convergence of the logistic regression model was shown to be much faster (less iterations) when using standardized input data compared to unscaled raw input (see figure below).



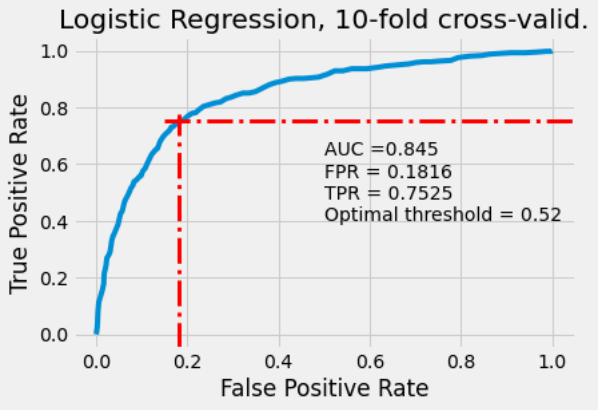
Comparison of the iteration steps needed for convergence of logistic regression solvers for raw and standardized input data.

| Solver | Unscaled | Standardized |
|---|---|---|
| newton-cg | 72 | 16 |
| **liblinear** | 44 | **7** |
| lbfgs | 2150 | 78 |
| sag | 6200 | 795 |
| saga | 7250 | 1200 |

Optimized hyperparameters included input scaling (***unscaled***) the solver (***'lbfgs'***), number of iterations (***1450***), regularization strength `C` (***2.6***), and `class_weight` (***11***). Optimization of `C` alone has led to only a tiny improvement in ROC-AUC score for `newton-cg`, `liblinear`, and `lbfgs` solvers (with maximal increase of 0.18% for `liblinear` solver), while the accuracy has dropped by -0.063% from 0.9233 to 0.9227. ROC AUC slightly increases from 0.84212 to 0.84586 (by 0.444%) with the increase of the `class_weight` for the less numerous churn lable 1, with the optimal setting being `class_weight = 'balanced'` (close to `class_weight = {0:1, 1:11}`). However, the accuracy strongly decreases from 0.92312 to 0.80381 (by 12.92%). Solvers `sag` and `saga` converge to a considerably smaller AUC for raw data

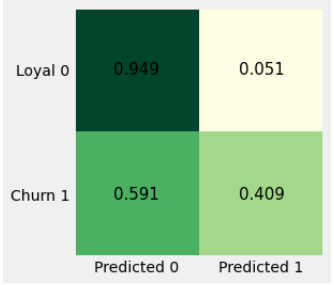
Area Under the Curve (AUC) of Receiver Characteristic Operator (ROC) helps us visualize how well our machine learning classifier is performing and choose the optimal classification threshold probability, which is important for our unbalanced data (only 7.94% of churn).

ROC curve




### 3.2 Decision Tree

This is the least powerful method, yielding to logistic regression in Sec. 3.1 and ensemble tree-based methods in Sects. 3.3-4. Decision tree is not sensitive to scaling of the input. It is quite deep for our problem: `max_depth` = ***36***. The only optimized hyperparameter was `criterion` = ***`gini`***. The `gini` index is $1 - p_0^2 - p_1^2$, where $p_i$ is the probability of churn status i. Other hyperparameters were not optimized, since the Random Forest and XGBoost models have all the



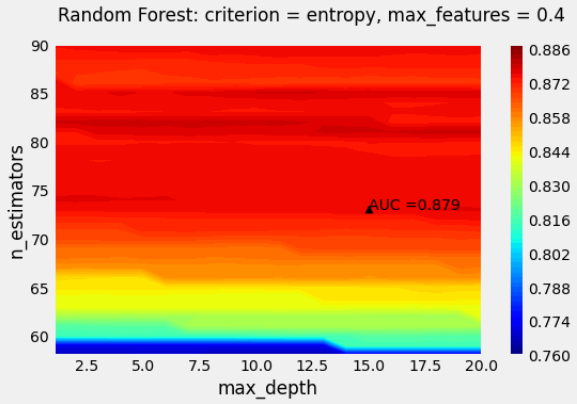
same and some extra hyperparameters. Confusion matrix plot below explains a relatively small accuracy due to a relatively large number of non-churn (label 0) with non-optimal TN/N = 0.949.

### 3.3 Random Forest

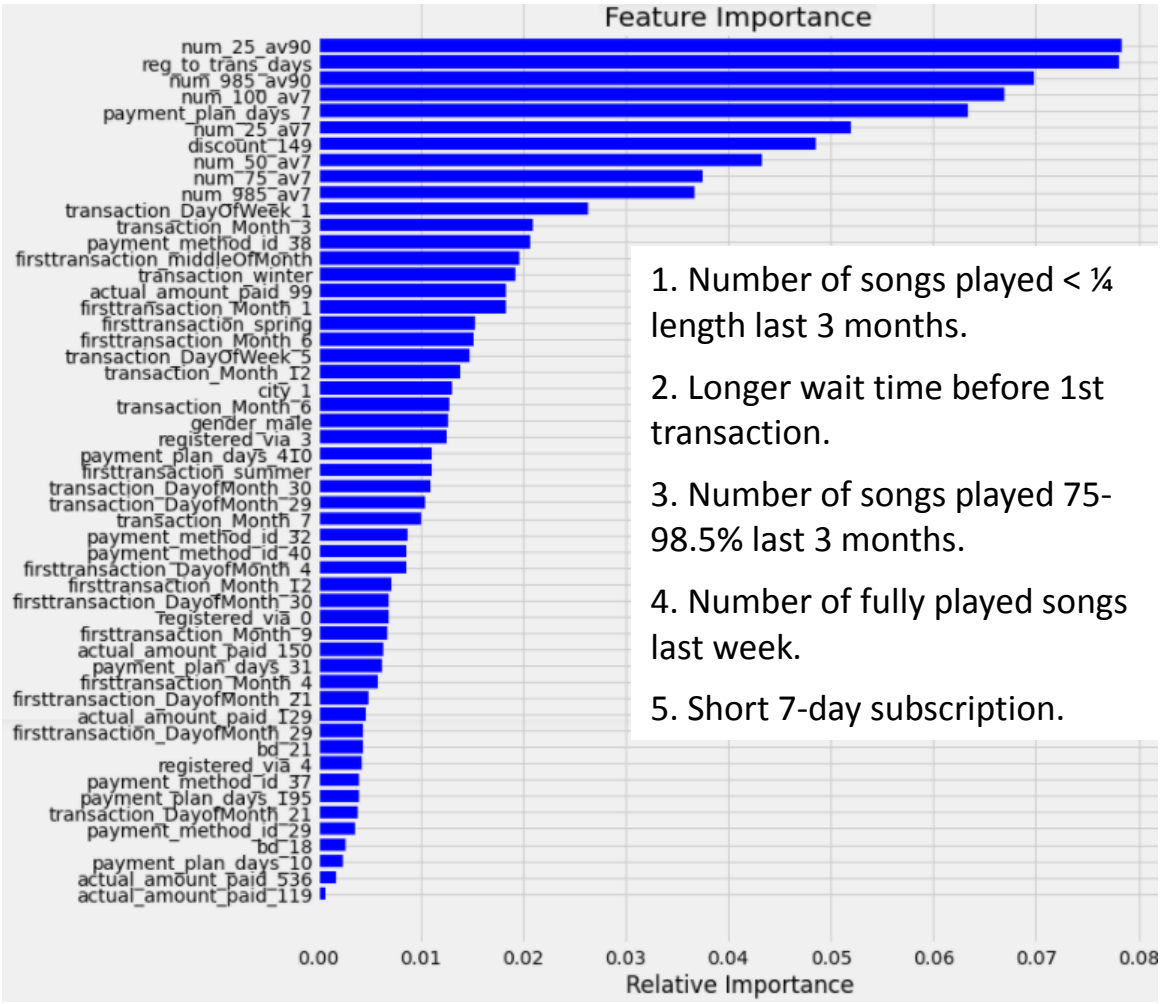
Based on 30% test dataset, this is the best model, closely followed by XGBoost model in Sec. 3.4. Hyperparameter optimization steps are shown in the table below.

| Optimization | criterion | max_features | max_depth | n_estimators | ROC AUC | Accuracy | TNR | TPR |
|---|---|---|---|---|---|---|---|---|
| Default | gini | auto | None | 100 | 0.853444 | 0.933632 | 0.990486 | 0.274510 |
| criterion | entropy | auto | None | 100 | 0.865571 | 0.933437 | 0.990698 | 0.269608 |
| max_features | entropy | 0.4 | None | 100 | 0.866910 | 0.934216 | 0.989641 | 0.291667 |
| max_depth | entropy | 0.4 | 15 | 100 | 0.878670 | 0.937524 | 0.992812 | 0.296569 |
| n_estimators | entropy | 0.4 | 15 | 73 | 0.879220 | 0.937330 | 0.993023 | 0.291667 |

The AUC landscape is pretty flat. The contour plot of AUC below shows that `n_estimators` hyperparameter is more important than `max_depth`. Due to the ensemble nature of this model, individual trees do not have to be as deep as in usual decision tree model.

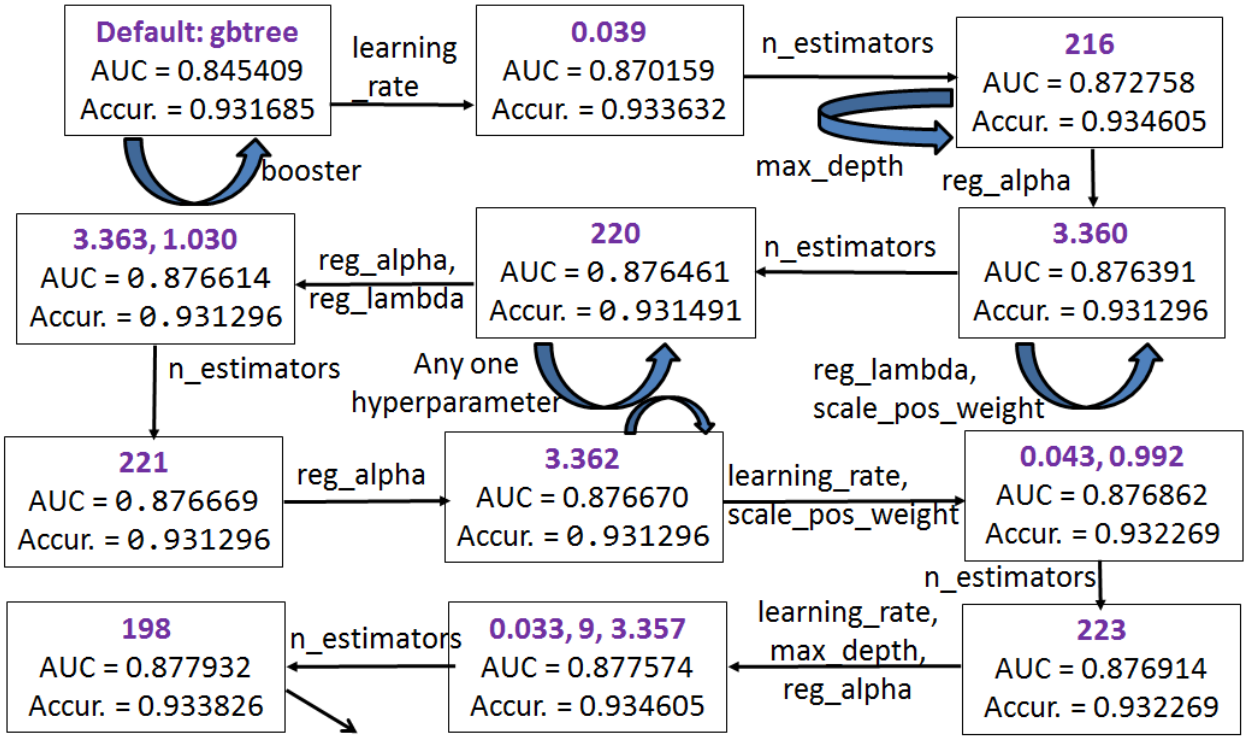



Random Forest model allows calculating the feature importance, shown below. The most influential predictors are related to the number of songs played and subscription duration.




### 3.4 XGBoost

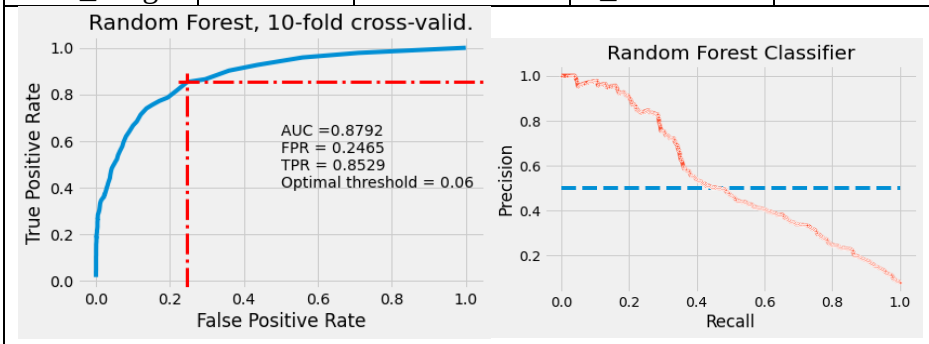
This model is 2nd best as judged by 30% test dataset, thus a detailed optimization was performed for it (see results in Sec. 3.5 table). However, this model is the best when using 10-fold cross-validation. The hyperparameters were optimized one-by-one and, when no improvement was achieved, the number of simultaneously optimized hyperparameters was increased by one (illustrated below).

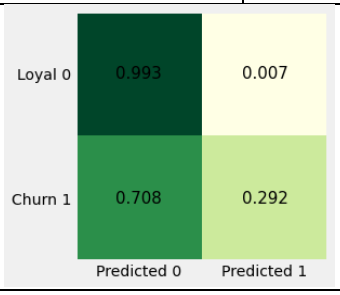



### 3.5 Performance and Hyperparameters

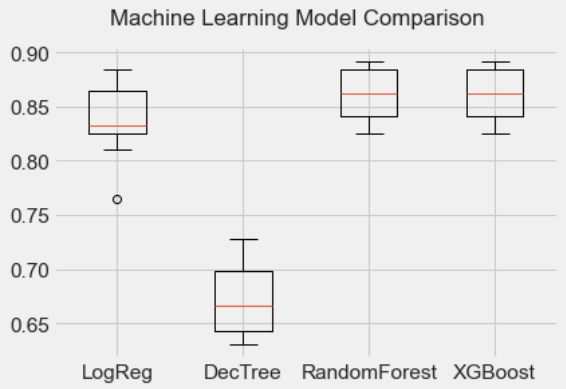
All models were optimized by maximizing the AUC score for 30% test split. Optimal hyperparameter values are shown in the table below, followed by ROC, precision-recall, and confusion matrix plots.

| **Logistic Regression** | | **Decision Tree** | | **Random Forest** | | **XGBoost** | |
|---|---|---|---|---|---|---|---|
| **AUC** | .846531 | .696687 | | **0.879220** | | 0.878893 | |
| Accuracy | .802452 | .905800 | | 0.937330 | | 0.934800 | |
| Data | Unscaled | Standardized | | Unscaled | | Unscaled | |
| solver | lbfgs | criterion | gini | criterion | entropy | booster | gbtree |
| max_iter | 1450 | | | max_features | 0.4 | learning_rate | 0.033 |
| C | 2.6 | | | max_depth | 15 | max_depth | 9 |
| class_weight | 11 | | | n_estimators | 73 | n_estimators | 206 |
| | | | | | | reg_alpha | 3.356 |
| | | | | | | reg_lambda | 0.707 |
| | | | | | | scale_pos_weight | 0.955 |



Performance comparison using 10-fold cross-validation.




| Model | AUC CV | SD AUC | AUC test | Accuracy | TNR | TPR |
|---|---|---|---|---|---|---|
| **XGBoost** | .861316 | .024332 | .879147 | .924484 | .970402 | .392157 |
| **Random Forest** | .857779 | .026130 | .876482 | .934216 | .995560 | .223039 |
| **Logistic Regression** | .836855 | .033439 | .845863 | .803815 | .807188 | .764706 |
| **Decision Tree** | .672499 | .033853 | | | | |

## 4. Suggestions and insights

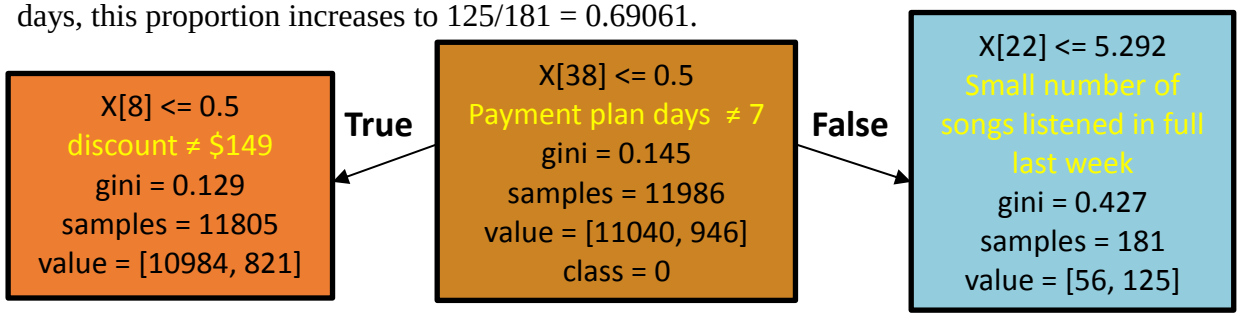
### 4.1 Improve data collection

- Collection of transaction and usage log information needs an enforcement of value completeness: 23.48% of members have missing transactions and usage logs, 21.69% of members do not have usage logs, 19.15% of members with transaction information have no membership information. For more details on missing records, see Venn diagram in Sec. 2.2.
- Membersip information suffers from high rate 66.93% of missing gender. This value should be made mandatory.
- Channels of data collection 3, 4, 7, and 9 need an improvement: 64.76% of all records acquired via these channels have missing gender.

### 4.2 Factors responsible for most churn

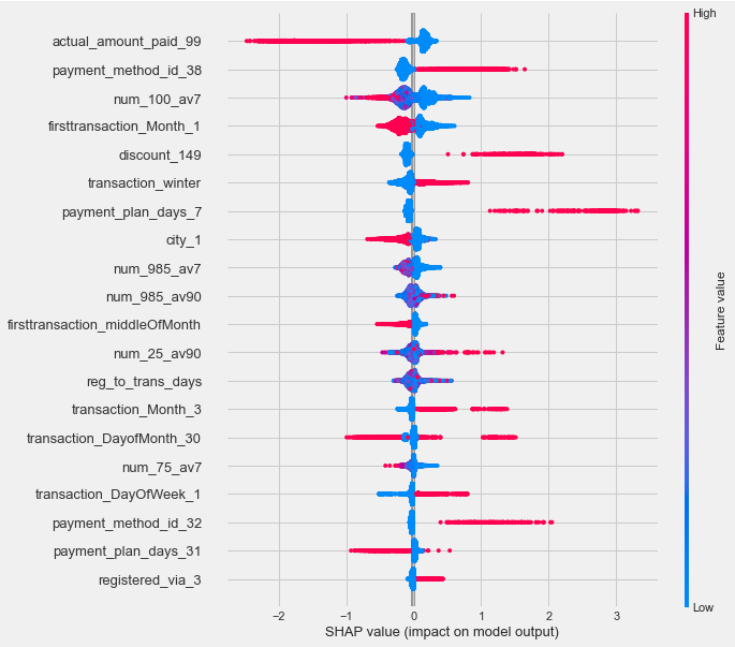
The decision tree model shows the following most predictive features.

- ***Payment plan is only for 7 days***: From the root node of the decision tree, we can see that the proportion of churn in all data till 2016/10/22 is 946/11040 = 0.078925. Following the 'False' branch to the right that corresponds to the group of transactions with payment plan for only 7 days, this proportion increases to 125/181 = 0.69061.



- ***Number of songs played in full last week***: The next important characteristic to consider is the number of songs played in full last week: following the 'True' branch downwards, we see that the churn rate increases to 0.848 in a group with standardized value $z \leq -0.407$, which corresponds to less than 5.28 songs (derived using mean $\mu = 18.749$ and standard deviation $\sigma = 33.093$, $x = \mu + z\sigma = 5.28$).
- ***Payment method id is 38***: Following the next 'True' branch downwards, the most decisive feature is payment method id being 38. If it is 38, then the proportion of churn is 103/117 = 0.88034.

Using SHapley Additive exPlanations (SHAP) for the best Random Forest model, we can rank additive predictive power of the features (more important on top in the SHAP summary plot below) and see if the higher (red) or lower (blue) values are responsible for the churn (right side).




- The most influential factor is paying \$99 for the plan: when not \$99 (value is 0, blue), the churn is more likely.
- Next in importance is payment method being 38: if it is (value is 1, red), then more churn.
- The smaller number of songs was played in full last week, the more likely churn.
- Winter and March have an increased churn.
- Short payment plan of 7 days.
- Small number of songs played 75-98.5% last week.
- Large number of songs played less than a quarter last 90 days.

## 5. Future Directions

- Add `is_cancel` feature to analysis, which was filtered out due to a probably too strict Information Value criterion: increase the upper allowed IV to its value of 1.327403 for `is_cancel`.
- Compare min-max scaling with standardization for logistic regression.
- The dependence of the number of features on the number of transactions presents a difficulty in preparing data for ML, which could be solved by grouping members with the same number of transactions `n_trans` and trying to predict churn for a customer who had exactly `n_trans` transactions, or simply by selecting a fixed number of transactions across some selection of customers. In the latter scenario, we would disregard all members with less than a chosen number `n_trans` of transactions, and take only the first or latest `n_trans` transactions for each of the remaining members. This will allow one to improve predictions due to using more detailed information on product usage, namely, all values vs averaged over the week, month, and 90 days.
- Try other models, such as SVM (`SVC()`), KNN (`KNeighborsClassifier()`), LDA (`LinearDiscriminantAnalysis()`), and Gaussian Naive Bayes (`GaussianNB()`).

**Attachments**

1. Finalized code in the Jupyter notebook <Capstone2_CustomerChurn.ipynb>.

2. Model Metrics file <Metrics.csv>, which contains the best model's features, hyperparameters, and performance metrics.