KKBox User Analysis
Aarya Suryavanshi & Emily You

Data Investigation and Exploration

Introduction

The data used for this analysis is given by KKBox, an online music platform based in Taiwan. The data includes characteristics (such as age, gender, and amount paid) of users whose memberships expire within the month of February 2017. KKBox uses BigQuery as a data warehouse, and it was assumed that the company used its own software to collect data as the dataset was obtained directly from Kaggle. Issues with the data include false self-reporting (users may incorrectly report their personal data) and glitches in the data collection software. For example, there were unreasonable age data ranging from -3152 to 2016, which is likely due to one of the problems. Each data point is a transaction, where one user can be represented multiple times in the data if he or she makes multiple transactions. Users are identified by their "MSNO" (ID) value, so users who make multiple transactions can be identified.

Response Variables

The continuous response variable is the total number of seconds a user plays in a day. The date at which this data was collected is included in the dataset. As KKBox pays artists based on the number of plays of their songs, users who listen to more music will on average cost KKBox more in terms of artist dividends. Therefore, it is vital to know how many seconds an individual streams music so that KKBox can more accurately determine its future cost.

The classification variable will be determining if the user churned (defined as 1) or did not churn (defined as 0). Churn is defined as the user failing to make a service transaction within 30 days of their plan expiring. For the average music streaming platform, user retention is vital as it ensures a steady supply of income and reduces the costs necessary to recruit new customers. Additionally, if customers present habits or characteristics that are associated with churning, KKBox can target its retention advertising on these customers rather than on the entire user base, which will reduce the cost of customer retention. Therefore, it is in KKBox's best interest to predict user churn rate and understand what factors are related to users who continue to stream music on their platform.

The analysis from this data set can be extended to other subscription-based services, such as HelloFresh and Netflix. Additionally, as more people switch to listening to music on streaming services, the amount of money artists are paid per song has become a more contentious topic. This analysis can be used to predict cost due to artist dividends in the future, as any changes in cost per song will simply be multiplied by total seconds listened. Furthermore, the user churn analysis can be used to understand the characteristics of users who tend to churn, which KKBox can use to determine its business strategy.

Data Processing

Because this analysis is interested in user-specific information rather than transactional data, duplicates in the data overrepresent users who make more than one transaction. Therefore, duplicate observations for users with more than one transaction were removed from the data set. Because there is a column indicating whether the user churned, the removal does not affect the analysis on user churn rate.

As the next step, several features that seemed reasonable, such as the difference between membership start date and when daily music user log was taken, and membership duration, were generated based on the data. Also, the data set was reduced from 725,000 data points to 100,000 through random selection of transactions to reduce computation time. All of the variables in the processed dataset are listed and described in the Appendix.

Next, data points with ages less than 1 and greater than 150 were removed, as this is outside the normal lifespan for a traditional user. Lastly, data points with missing gender value were also removed from the data set. It should be noted that removing data points without a gender will underestimate the churn rate due to the correlation between users with missing gender and those who churned. This can be seen in Figure 1 in the Appendix as most users with missing gender had the is_churn value of 0 (Note that Figure 1 was produced from a sample of data points to reduce plot generation time). As shown in Figure 2 and 3, most of the users with extreme ages had missing gender and missingness in gender is fairly evenly distributed within the accepted range of age.

For classification training set, 53397 observations with is_churn value of 0 and 46603 observations with is_churn of 1 were selected from the dataset. This process ensured that there are enough data points of churned users to train the models as the original dataset had an extremely large proportion of users who did not churn.

Prediction

The goals of this analysis are to predict if a user will churn and to predict the total seconds a user will listen to music on the platform. 80% of the data was randomly selected into the training set with the remainder of the data held out for the test set.

To predict how many seconds of music a user streams on KKBox, a linear regression model was created. The focus of the model was on predictive power rather than interpretability because the purpose of the analysis is to generate an accurate prediction of the total number of seconds users stream from the platform. KKBox's revenue strategy would depend on this number rather than what factors are associated with the total number of seconds.

The metric used for model selection was the cross-validated mean absolute error (CVMAE). 5-fold cross validation was used as this is a common practice used in the field of data science and has reasonable computation time. The model with the smallest MAE was chosen. MAE was used as opposed to root mean squared error (RMSE) as the metric because there is no need for the

model to penalize larger errors more heavily than smaller errors. In other words, the error of 10 seconds is equivalent to two times the error of 5 seconds as it is assumed that the cost per song stays constant (the total number of seconds users play does not affect the cost per song).
First, the baseline model, which predicts the mean of all total_secs in the dataset, was created. Then the correlations of the variables in the dataset were observed by creating a ggpairs plot. For variables with right-skewed distribution, log transformation was considered at first to make them more "normal." However, this was not appropriate as some of the data included 0s. Square root transformation was considered and transforming payment_plan_days improved the model.

Next, reasonable interaction terms were hand-picked from studying the correlation between the variables and choosing the ones that were highly correlated (>0.5 correlation coefficient). The variables that were tested are listed in the Appendix. Out of the covariates that were tested, the interaction term between is_auto_renew:memdur improved the model. In addition, higher order terms, also chosen based on the context of the analysis, were tested to see if they improved the model. The terms that were selected were: num_unq^2, num_100^2, actual_amount_paid^2, but none of them resulted in an improvement of CVMAE.

Another strategy that was used to obtain the best model was through regularization. Ridge and lasso regression were implemented for feature selection and lower variance. However, due to the nature of regularization, the bias of the models would be higher. The best model for the regression task was created through lasso regularization. The final model predicts total seconds using the covariates and coefficients listed under the Regression Model section in the Appendix.

The estimated test error of this model is 570.7867. Compared to the baseline model, the final model outperformed the baseline by 5727.0173. This was estimated by the 5-fold cross-validation error, which generates the estimated prediction error. However, cross-validation error was used for model selection and performance evaluation, so this number is an underestimate of the test error.

Model	**CV MAE**
Baseline	6297.804
OLS with all covariates	574.5504
OLS with sqrt(payment plan days)	574.5211
OLS with sqrt(payment plan days) & interaction term	574.5123
Ridge regression model	573.6239
Lasso regression model (Final)	570.7867

Table 1: Cross-validation Mean Absolute Error (CV MAE) for Regression Models

For the classification problem, the loss was defined as following:

$$Loss = 2 \times \#False\ Negative + 1 \times \#False\ Positive$$

The loss function for this analysis is based on the assumption that false negatives are considered worse than false positives. This is because false negatives will cause KKBox to overestimate its customer retention and therefore its revenue, whereas false positives only add additional cost spent for customer retention. Thus, it was assumed that false negatives are twice as worse as the false positives. Furthermore, true positives and true negatives were considered to have 0 loss since correct predictions do not lead to any additional monetary loss for KKBox (compared to the cost of actions currently taken by the company). For example, predicting a user will churn when they do churn will not cause the company to lose any more money than the amount they would be losing due to the churning customer.

The model that minimized the average loss was chosen to predict whether a user will churn. Each average loss was calculated through 5-fold cross validation to ensure that model selection is not based off of chance. This model needed be interpretable as it is vital for KKBox to determine which user behaviors are associated with churning. The company can benefit from knowing these features so that they can take appropriate measures to help prevent users from churning.

In the process, five models were created: the baseline model that predicts that all users did what the majority of users did, which is to not churn; a logistic regression model, a ridge model, a lasso model, and a classification and regression tree model. Each model was fitted on all covariates in the dataset. The average loss of each model is in Table 2 below.

Model	**CV Average Loss**
Baseline	0.5532461
Logistic Regression	0.2437412
Ridge	0.2644092
Lasso	0.2478048
CART	0.3051845

Table 2: Loss for Classification Models

Table 2 shows that the baseline model has the highest loss and the logistic regression model has the lowest. Therefore, logistic regression was chosen as the best model. The coefficients for this model can be found in the Classification Model section of the Appendix.

The estimated test error (loss) of the final model is 0.2437412. Compared to the baseline model, the final model outperformed the baseline by 0.3095049. However, cross-validation error was used for model selection and performance evaluation, so this number is an underestimate of the test error, as mentioned before.

Appendix

Variable Definition (Cleaned Dataset)

Variable Name	Definition
city	City of the user
bd	Age of the user
gender	Gender of the user
registered_via	Registration method
is_churn	If the user churned (1) or not (0)
payment_method_id	Payment method
payment_plan_days	Length of membership plan in days
plan_list_price	Plan price in New Taiwan Dollar (NTD)
actual_amount_paid	Actual amount paid in New Taiwan Dollar
is_auto_renew	If the user's plan is automatically renewed
is_cancel	Whether the user canceled the membership in this transaction (1) or not (0)
num_25	Number of songs played less than 25% of the song length
num_50	Number of songs played between 25% to 50% of the song length
num_75	Number of songs played between 50% to 75% of of the song length
num_985	Number of songs played between 75% to 98.5% of the song length
num_100	Number of songs played over 98.5% of the song length
num_unq	Number of unique songs listened to (daily)
total_secs	Total seconds of music listened to on platform (daily)
diffdate	Difference in days between when user first

	joined and the day on which data on daily music usage was taken
memdur	Duration of Membership

Interaction Terms Tested

- Payment_plan_days: plan_list_price, actual_amount_paid
- Num_100: num_unq
- Is_auto_renew:duration of membership

Classification Model

Covariate	**Coefficient**
intercept	6.513338e+00
city3	7.321332e-02
city4	4.591899e-02
city5	2.623933e-02
city6	8.267033e-03
city7	2.430892e-04
city8	-4.407508e-02
city9	-1.909361e-02
city10	-6.540726e-02
city11	-4.078861e-02
city12	-1.273656e-01
city13	-1.433601e-02
city14	-6.492941e-02
city15	-1.111084e-02
city16	-1.129068e-01
city17	-1.375922e-01

city18	-5.507905e-02
city19	-5.241178e-01
city20	-7.734110e-01
city21	-1.610933e-01
city22	-4.320248e-02
bd	3.536355e-03
gender(Female)	6.180988e-04
registered_via4	2.233667e-01
registered_via7	4.450947e-02
registered_via9	-2.730469e-01
registered_via13	1.027680e-01
payment_method_id6	-1.024815e+01
payment_method_id8	-7.460118e-01
payment_method_id10	-7.507448e+00
payment_method_id11	-8.814547e+00
payment_method_id12	3.414919e+00
payment_method_id13	-1.099653e+00
payment_method_id14	-8.579388e+00
payment_method_id15	-5.828535e+00
payment_method_id16	-8.325744e+00
payment_method_id17	-2.962865e+00
payment_method_id18	-8.389073e+00
payment_method_id19	-8.937883e+00
payment_method_id20	4.217278e+00
payment_method_id21	-7.803039e+00

payment_method_id22	-1.735820e+00
payment_method_id23	-8.953128e+00
payment_method_id26	-4.796592e+00
payment_method_id27	-8.993728e+00
payment_method_id28	-7.905663e+00
payment_method_id29	-8.323780e+00
payment_method_id30	-7.557796e+00
payment_method_id31	-9.808737e+00
payment_method_id32	-5.656308e+00
payment_method_id33	-9.038535e+00
payment_method_id34	-9.218703e+00
payment_method_id35	-3.621192e+00
payment_method_id36	-7.719294e+00
payment_method_id37	-8.723388e+00
payment_method_id38	-7.701840e+00
payment_method_id39	-7.853613e+00
payment_method_id40	-7.746347e+00
payment_method_id41	-7.788412e+00
payment_plan_days	1.946699e-02
plan_list_price	2.737897e-02
actual_amount_paid	-2.405303e-02
is_auto_renew1	-1.544053e+00
is_cancel1	4.836891e+00
num_25	6.217058e-04
num_50	-2.949601e-03

num_75	5.209281e-03
num_985	-1.037790e-02
num_100	-2.163460e-04
num_unq	1.012798e-03
total_secs	-5.296634e-06
diffdate	-1.655768e-02
memdur	1.670847e-02

Regression Model

Covariate	**Coefficient**
intercept	-7257.982894
city4	37.336249
city9	-54.385643
city13	21.342211
gender1	-32.367506
registered_via4	54.416138
registered_via7	-2.360411
is_churn1	-12.767832
payment_method_id11	1629.317807
payment_method_id19	43.159230
payment_method_id23	-76.171969
payment_method_id28	-1444.733759
payment_method_id37	27.529833
payment_method_id38	-36.670603
payment_method_id39	51.108237

payment_method_id40	4.882313
num_25	-12.201816
num_50	69.059558
num_75	146.699679
num_985	214.012591
num_100	217.058681
num_unq	27.754900

Graph

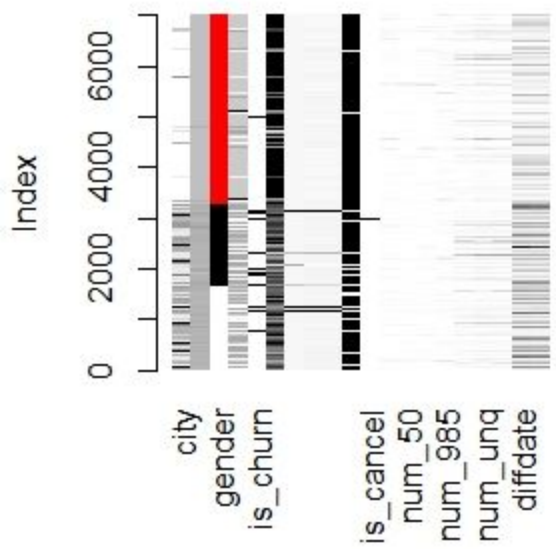

Figure 1: Sorted (and Missing) Gender Values and Missingness in Other Variables

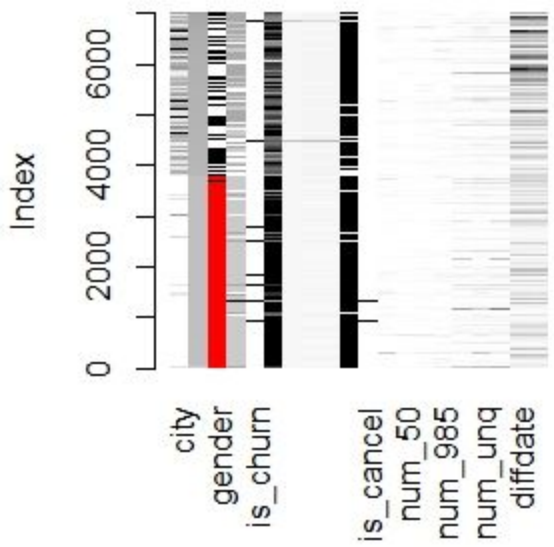

Figure 2: Sorted Age Values and Missingness in Other Variables (Variable to the Left of Gender is Age)

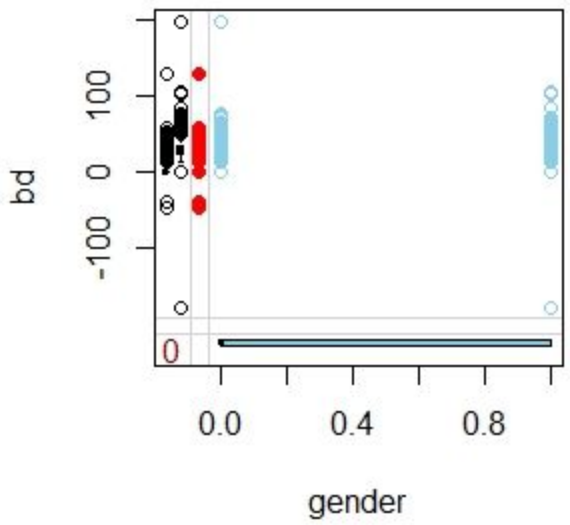

Figure 3: Margin Plot of Age and Gender

Code

```
library(tidyverse)
library(ggplot2)
library(GGally)
#install.packages("Hmisc")
#install.packages("VIM")
library(Hmisc)
library(VIM)
library(cvTools)

set.seed(123)

members = read.table(file = 'members_v3.csv', sep = ',', header = TRUE, stringsAsFactors=FALSE)
train = read.table(file = 'train_v2.csv', sep = ',', header = TRUE, stringsAsFactors=FALSE)
transactions = read.table(file = 'transactions_v2.csv', sep = ',', header = TRUE, stringsAsFactors=FALSE)
userlogs = read.table(file = 'user_logs_v2.csv', sep = ',', header = TRUE, stringsAsFactors=FALSE)
merged = merge(members, train, by='msno')
merged1 = merge(merged, transactions, by='msno')
# Remove duplicates before merging to reduce size
merged1 = merged1[!duplicated(merged1$msno),]
merged2 = merge(merged1, userlogs, by='msno')

# Don't include duplicates (disregard users who show up twice)
df = merged2[!duplicated(merged2$msno),]

# Filter out nan/NA values (still missing data in gender, etc.)
new_df <- df[rowSums(is.na(df)) == 0,]
is.nan.data.frame <- function(x)
  do.call(cbind, lapply(x, is.nan))
new_df <- new_df[rowSums(is.nan(new_df)) == 0,]
new_df = new_df[rowSums(new_df=="")!=ncol(new_df), ]
sapply(new_df, function(x) all(is.na(x) | x == " "))

# Make diff_price column (difference btwn plan list price and actual amt paid)
# new_df$diff = new_df$plan_list_price - new_df$actual_amount_paid
# filtered = transactions[transactions[,10]!=0,]

# Determine whether to delete bd (age) column
hist(new_df$bd)
hist(new_df[new_df$bd >0& new_df$bd <200,]$bd)
nrow(new_df[new_df$bd >0& new_df$bd <200,])
# We could delete all rows with age that doesn't make sense -> 342348 rows
# But if we do, possible selection bias if age is not missing at random
# Might be better to disregard this column in general?
# But then I saw someone else's analysis saying age is one of the biggest factors...
```

```
#
https://repositories.lib.utexas.edu/bitstream/handle/2152/67638/ZHENG-MASTERSREPORT-2018.pdf?sequence=1
&isAllowed=y


# Difference between membership start date and when daily music user log was taken
# (In days)
date = as.Date(as.character(new_df[["date"]]), format = "%Y%m%d")
MusicDate =  as.Date(date, "%Y%m%d")
MemberDate =  as.Date(as.character(new_df[["registration_init_time"]]), format = "%Y%m%d")
MemberDate =  as.Date(MemberDate, "%Y%m%d")
new_df$diffdate = MusicDate - MemberDate
new_df = new_df[sample(nrow(new_df), 100000), ]
#plot age against other variables to see if there is a bias in the bad values

# Membership Duration (in days)
ExpDate = as.Date(as.character(new_df[["membership_expire_date"]]), format = "%Y%m%d")
InitDate =  as.Date(as.character(new_df[["registration_init_time"]]), format = "%Y%m%d")
new_df$memdur = ExpDate - InitDate

#separate train and test
train.ind = sample(1:nrow(new_df), 0.80*nrow(new_df))
df.train = new_df[train.ind,]
df.test = new_df[-train.ind,]

# Data Exploration
# num_unique songs vs. total_secs
ggplot(df.train) + geom_point(aes(total_secs, num_unq)) + xlim(0,100000) + ylim(0,500)
# distribution of membership days
hist(df.train$payment_plan_days)


new_df2 = select(new_df, -registration_init_time, -transaction_date, -membership_expire_date, -date)
# matrixplot(new_df2, interactive = F, sortby = "num_100")

#with ages between 1 and 150 and removing rows without gender
goodage = new_df2[!(new_df2$gender == "" | is.na(new_df2$gender)), ]
  filter(between(bd, 1, 150))


#values with ages less than 1 or greater that 150
weirdage = new_df2 %>% filter(!between(bd, 1, 150))

#remove gender from weird age
weirdagegender = weirdage[!(weirdage$gender == "" | is.na(weirdage$gender)), ]
#removes a lot of values

#age vs. number of unique songs
```

```
#most people put 0s when they don't want to deal with anything else
#i dont think this is as helpful as figuring out how to get the matrix to work
ggplot(weirdagegender) + geom_point(aes(bd, num_unq))
ggplot(weirdagegender) + geom_point(aes(bd, is_churn))
ggplot(weirdagegender) + geom_point(aes(bd, total_secs))
ggplot(weirdagegender) + geom_point(aes(bd, num_100))
ggplot(weirdagegender) + geom_point(aes(bd, city))
#i wanted to see what percent of people who don't put in a proper age also don't put in their gender
#could comment on user input
badagevgender = (nrow(weirdage) - nrow(weirdagegender))/nrow(weirdage)

#good age data
ggplot(goodage) + geom_point(aes(bd, num_unq))
ggplot(goodage) + geom_point(aes(bd, is_churn))
ggplot(goodage) + geom_point(aes(bd, total_secs))
ggplot(goodage) + geom_point(aes(bd, is_cancel))
ggplot(goodage) + geom_point(aes(bd, num_100))
ggplot(goodage) + geom_point(aes(bd, city))

test_df = goodage[2:10]
#ggpairs never works :(

sampleGG = df.train[sample(nrow(df.train), 200), ]
sampleGG = select(sampleGG, -msno, -registration_init_time, -transaction_date, -membership_expire_date, -date)
ggpairs(sampleGG)

#find where NA in gender occurs to understand bias in data
#these plots look almost the exact same and i can't figure out how to fix this

# sample 5000 points
sample = new_df2[sample(nrow(new_df2), 5000), ]
sample[sample$gender == "female","gender"] <- 1
sample[sample$gender == "male","gender"] <- 0

matrixplot(sample, sortby = c('bd'))
matrixplot(sample, sortby = c('gender'))
marginplot(sample[,c("gender","bd")])
marginplot(goodageG[,c("gender","bd")])

# Sort by age and see if missingness has correlation with gender
goodageG = new_df2%>%filter(between(bd, 1, 150))
goodageG[goodageG$gender == "female","gender"] <- 1
goodageG[goodageG$gender == "male","gender"] <- 0
matrixplot(goodageG, sortby = c('gender'))
matrixplot(goodageG, sortby = "bd")

# Final Dataset
finaltrain = df.train[!(df.train$gender == "" | is.na(df.train$gender)), ] %>%
```

```
  filter(between(bd, 1, 150))
finaltest = df.test[!(df.test$gender == "" | is.na(df.test$gender)), ] %>%
  filter(between(bd, 1, 150))

# Drop msno and dates
drops <- c("msno","registration_init_time","transaction_date","membership_expire_date","date")
finaltrain = finaltrain[ , !(names(finaltrain) %in% drops)]

# Delete 1 user with payment plan day of 0
finaltrain = finaltrain[finaltrain$payment_plan_days != 0,]

# Modeling
# Baseline Models
base_reg = mean(finaltrain$total_secs)
base_error = mean(abs(finaltrain$total_secs - base_reg))


# Regression Model
# All covariates
finaltrain$city <- as.factor(finaltrain$city)
finaltrain$registered_via <- as.factor(finaltrain$registered_via)
finaltrain$is_churn  <- as.factor(finaltrain$is_churn)
finaltrain$payment_method_id  <- as.factor(finaltrain$payment_method_id)
finaltrain$is_auto_renew <- as.factor(finaltrain$is_auto_renew)
finaltrain$is_cancel <- as.factor(finaltrain$is_cancel)
finaltrain$diffdate = as.numeric(finaltrain$diffdate)
finaltrain$memdur = as.numeric(finaltrain$memdur)

model = lm(total_secs ~ ., data=finaltrain)
model.cv = cvFit(model, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model.cv)
summary(model)


# Square root transformation (inexact science anyways, so this is fine)
# Square rooting payment plan days improved model
model_sqrt = lm(total_secs ~ .+sqrt(plan_list_price) - plan_list_price, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)

model_sqrt = lm(total_secs ~ .+sqrt(actual_amount_paid) - actual_amount_paid, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)

model_sqrt = lm(total_secs ~ .+sqrt(num_25) - num_25, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)
```

```
model_sqrt = lm(total_secs ~ .+sqrt(num_100) - num_100, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)

model_sqrt = lm(total_secs ~ .+sqrt(num_unq) - num_unq, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)

model_sqrt = lm(total_secs ~ .+sqrt(diffdate) - diffdate, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)

model_sqrt = lm(total_secs ~ .+sqrt(memdur) - memdur, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)

# Selected model
model_sqrt = lm(total_secs ~ .+sqrt(payment_plan_days) - payment_plan_days, data=finaltrain)
model_sqrt.cv = cvFit(model_sqrt, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_sqrt.cv)


# Interaction terms
# is_auto_renew:memdur improved the model
model_int = lm(total_secs ~ .+sqrt(payment_plan_days) - payment_plan_days+
                is_auto_renew:memdur, data=finaltrain)
model_int.cv = cvFit(model_int, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_int.cv)

# Higher order terms
# Tried num_unq^2, num_100^2, actual_amount_paid^2
# No higher term improved the model
model_hi = lm(total_secs ~ .+sqrt(payment_plan_days) - payment_plan_days+
               is_auto_renew:memdur +
               actual_amount_paid^2, data=finaltrain)
model_hi.cv = cvFit(model_hi, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model_hi.cv)

finaltrain$payment_plan_days = sqrt(finaltrain$payment_plan_days)
finaltrain$int = (as.numeric(finaltrain$is_auto_renew)-1)*finaltrain$memdur

# Ridge regression
# Standardization -standardize both X and Y (already done by glmnet)
# How to compare models? Predict, transform back, and calculate MAE!
# Do variable selection thru LASSO and use those covariates to run OLS
# to make it more interpretable
# What if we're classifying a binary var - output can be in 0 and 1.
```

```
# How to standardize binary variables? - still do it, just not interpretable (regularization is just not interpretable anyways)

#install packages
# install.packages('glmnet')
library(MASS)
library(glmnet)
set.seed(1)
finaltrain[finaltrain$gender == "female","gender"] <- 1
finaltrain[finaltrain$gender == "male","gender"] <- 0
train_matrix = model.matrix(total_secs ~ .,finaltrain)[, -18]
# Center Y
mean_Y = mean(finaltrain[,18])
centered_Y = (finaltrain[,18]-mean_Y)

fit_ridge = glmnet(train_matrix, centered_Y, alpha = 0)
fit_ridge_cv = cv.glmnet(train_matrix, centered_Y, alpha = 0, type.measure="mae", K = 5, lambda = seq(0, 1000))
coef(fit_ridge_cv, s = "lambda.min")
cv_error = fit_ridge_cv$cvm[fit_ridge_cv$lambda == fit_ridge_cv$lambda.min]
print(cv_error)

# Lasso regression (subset selection)
# Lasso gave better results than ridge
fit_lasso = glmnet(train_matrix, centered_Y, alpha = 1)
fit_lasso_cv = cv.glmnet(train_matrix, centered_Y, alpha = 1, type.measure="mae", K = 5, lambda = seq(0, 1000))
coef(fit_lasso_cv, s = "lambda.min")
print(fit_lasso_cv$lambda.min)
cv_error = fit_lasso_cv$cvm[fit_lasso_cv$lambda == fit_lasso_cv$lambda.min]
print(cv_error)

# OLS with covariates selected from Lasso
# install.packages("coefplot")
library(coefplot)
covariates = extract.coef(fit_lasso_cv)
covariates
model = lm(total_secs ~ .-int, data=finaltrain)
model.cv = cvFit(model, data=finaltrain, y=finaltrain$total_secs, K=5, seed=1, cost=mape)
print(model.cv)

# Lasso model had the best performance!


# Classification Modeling
set.seed(123)
x.sub.1 = subset(df, is_churn == 1)
x.sub.0 = subset(df, is_churn == 0)
x.sub.0 = x.sub.0[sample(nrow(x.sub.0), (100000 - nrow(x.sub.1))), ]
```

```
x.sub = rbind(x.sub.0, x.sub.1)

train.ind.class = sample(1:nrow(x.sub), 0.80*nrow(x.sub))
x.sub.train = x.sub[train.ind.class,]
x.sub.test = x.sub[-train.ind.class,]
x.sub.train = x.sub.train[!(x.sub.train$gender == "" | is.na(x.sub.train$gender)), ] %>%
  filter(between(bd, 1, 150))
x.sub.test = x.sub.test[!(x.sub.test$gender == "" | is.na(x.sub.test$gender)), ] %>%
  filter(between(bd, 1, 150))

# Difference between membership start date and when daily music user log was taken
# (In days)
date = as.Date(as.character(x.sub.train[["date"]]), format = "%Y%m%d")
MusicDate =  as.Date(date, "%Y%m%d")
MemberDate =  as.Date(as.character(x.sub.train[["registration_init_time"]]), format = "%Y%m%d")
MemberDate =  as.Date(MemberDate, "%Y%m%d")
x.sub.train$diffdate = MusicDate - MemberDate

# Membership Duration (in days)
ExpDate = as.Date(as.character(x.sub.train[["membership_expire_date"]]), format = "%Y%m%d")
InitDate =  as.Date(as.character(x.sub.train[["registration_init_time"]]), format = "%Y%m%d")
x.sub.train$memdur = ExpDate - InitDate

# Drop msno and dates
drops <- c("msno","registration_init_time","transaction_date","membership_expire_date","date")
x.sub.train = x.sub.train[ , !(names(x.sub.train) %in% drops)]

# Delete 1 user with payment plan day of 0
x.sub.train = x.sub.train[x.sub.train$payment_plan_days != 0,]
x.sub.train$city <- as.factor(x.sub.train$city)
x.sub.train$registered_via <- as.factor(x.sub.train$registered_via)
x.sub.train$is_churn  <- as.factor(x.sub.train$is_churn)
x.sub.train$payment_method_id  <- as.factor(x.sub.train$payment_method_id)
x.sub.train$is_auto_renew <- as.factor(x.sub.train$is_auto_renew)
x.sub.train$is_cancel <- as.factor(x.sub.train$is_cancel)
x.sub.train[x.sub.train$gender == "female","gender"] <- 1
x.sub.train[x.sub.train$gender == "male","gender"] <- 0

#baseline
set.seed(123)
base.vec = as.factor(rep(base_class, nrow(x.sub.train)))
confusionMatrix(base.vec, as.factor(x.sub.train$is_churn))

confM = table(factor(base.vec,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)),
     factor(x.sub.train$is_churn,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)))
```

```
base.loss = confM[3]/nrow(x.sub.train)

#logistic regression
set.seed(123)
LogModel <- glm(is_churn ~., data=x.sub.train, family="binomial")
summary(LogModel)
#ROC
class.log.train = x.sub.train
class.log.train$is_churn <- as.numeric(class.log.train$is_churn)
class.log.train$is_churn = class.log.train$is_churn-1
class.mean = mean(class.log.train$is_churn)
class.log.train$scaled.is_churn = (class.log.train$is_churn-class.mean)
class.log.train$pred.is_churn = as.numeric(predict(LogModel, newdata = x.sub.train[-5]))

cost <- function(r, pi) {
  weight1 = 2 #cost for getting 1 wrong
  weight0 = 1 #cost for getting 0 wrong
  c1 = (r==1)&(pi<0.5) #logical vector - true if actual 1 but predict 0
  c0 = (r==0)&(pi>=0.5) #logical vector - true if actual 0 but predict 1
  return(mean(weight1*c1+weight0*c0))
}

cv.log.err <- cv.glm(x.sub.train, LogModel, cost, K = 5)$delta
predicts.log = predict(LogModel,type="response", newx=class.log.train)
log.vec = rep(NA, nrow(x.sub.train))
for (i in 1:nrow(x.sub.train)) {
  if (predicts.log[i] > 0.5) {
    log.vec[i] = 1}
  else {
    log.vec[i] = 0
  }
}

confM = table(factor(log.vec,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)),
      factor(x.sub.train$is_churn,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)))
log.err = (2*confM[2]+confM[3])/nrow(x.sub.train)

#setting up dataframe
class.ridge.train = x.sub.train
y <- ifelse(class.ridge.train$is_churn == "1", 1, 0)

# Ridge regression - need to add weights
#alpha = 0
set.seed(123)
library(glmnet)
class.ridge.matrix = model.matrix(is_churn ~ ., x.sub.train)[, -5]
```

```
class.ridge.cv = cv.glmnet(class.ridge.matrix, y, alpha = 0,
                           nfolds = 5, family = 'binomial')                  #find minimum lambda
class.ridge.model = glmnet(class.ridge.matrix, y, family = 'binomial',
                           alpha = 0, lambda = class.ridge.cv$lambda.1se, type.measure ="auc")    #find model with k = 5

predicts.ridge = predict(class.ridge.model,type="response", s=class.ridge.cv$lambda.1se, newx=class.ridge.matrix)
ridge.vec = rep(NA, nrow(x.sub.train))
for (i in 1:nrow(x.sub.train)) {
  if (predicts.ridge[i] > 0.5) {
    ridge.vec[i] = 1}
  else {
    ridge.vec[i] = 0
  }
}
confM = table(factor(ridge.vec,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)),
      factor(x.sub.train$is_churn,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)))
ridge.err = (2*confM[2]+confM[3])/nrow(x.sub.train)


# Lasso regression - need to add weights
#alpha = 1
set.seed(123)
class.lasso.train = x.sub.train
y <- ifelse(class.lasso.train$is_churn == "1", 1, 0)

class.lasso.matrix = model.matrix(is_churn ~ ., x.sub.train)[, -5]
class.lasso.cv = cv.glmnet(class.lasso.matrix[seq(1, 15000), ], y[seq(1, 15000)], alpha = 1,
                           nfolds = 5, family = 'binomial')                  #find minimum lambda
class.lasso.model = glmnet(class.lasso.matrix[seq(0, 15000), ], y[seq(0, 15000)], family = "binomial",
                           alpha = 1, k = 5, type.measure = 'auc', lambda = class.lasso.cv$lambda.min)    #find model with k =
5

#ROC
predicts.lasso = predict(class.lasso.model,type="response", s=class.lasso.cv$lambda.min, newx=class.lasso.matrix)
lasso.vec = rep(NA, nrow(x.sub.train))
for (i in 1:nrow(x.sub.train)) {
  if (predicts.lasso[i] > 0.5) {
    lasso.vec[i] = 1}
  else {
    lasso.vec[i] = 0
  }
}
confM = table(factor(lasso.vec,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)),
      factor(x.sub.train$is_churn,
levels=min(as.numeric(x.sub.train$is_churn)-1):max(as.numeric(x.sub.train$is_churn)-1)))
```

```
lasso.err = (2*confM[2]+confM[3])/nrow(x.sub.train)

betacoeff = class.lasso.model$beta


# CART
#install.packages('caret')
#install.packages('rpart.plot')
set.seed(123)
library(rpart)
library(caret)
library(rpart.plot)
covariates_c = extract.coef(class.lasso.cv)
covariates_c

cpVals = data.frame(cp = seq(0, .04, by=.002))
# Loss Fn
# small average loss
Loss = function(data, lev = NULL, model = NULL, ...) {
  c(AvgLoss = mean(data$weights * (data$obs != data$pred)),
    Accuracy = mean(data$obs == data$pred))
}
#5-fold CV
trControl = trainControl(method = "cv",
                         number = 5,
                         summaryFunction = Loss)

# Weights for loss fn
weights = ifelse(x.sub.train$is_churn == 1, 2, 1)
# Train
# Lasso selected all covariates
loss.cv <- train(is_churn ~ .,
                 data=x.sub.train,
                 method="rpart",
                 weights = weights,
                 trControl=trControl,
                 tuneGrid=cpVals,
                 metric="AvgLoss",
                 maximize=FALSE)
loss.cv$bestTune
mod3432 = loss.cv$finalModel
prp(mod3432, digits=2)
# See model with least average loss
print(loss.cv)
```