KKBox User Analysis

Aarya Suryavanshi & Emily You

Introduction

The dataset used in this paper is from KKBox, a subscription-based music streaming service in Taiwan. The data consists of user characteristics (age, whether the user churned, etc.) and subscription plan characteristics (length of membership plan, etc). Linear regression was used to predict the total number of seconds a user streams in a day and classification to determine if a user will churn. Lasso regression model was chosen as the final model for regression and logistic regression model was chosen for the classification task. Names and definitions of the covariates used in this analysis can be found in the Appendix.

Prediction on the Test Set

The final models for both regression and classification were evaluated by cross-validation error (CV error); the model with the lowest error was chosen. For regression, MAE (mean absolute error) was used and for classification, the loss was defined as 2*# false negatives + 1*false positives. The estimates of test errors (CV errors) and the actual values of test errors are given in Table 1 in the Appendix.

The regression test error was 92% higher than the estimate of the test error. The classification model test error was 27% greater than the estimated test error. This was expected because CV error was used for both model selection and test error estimation, meaning the estimate tends to be lower than the true test error.

Inference

The final logistic regression model from the previous section was chosen for further analysis. First, statistically significant coefficients for the model were found at the 95% confidence level. The values are in Table 2 in the Appendix.

Only 10 of the 73 coefficients were statistically significant at the 95% level. This means that it is unlikely for the coefficients to have been observed if the coefficients were, in fact, 0. In other words, they are likely to be nonzero and therefore are likely to be associated with changes in the response variable, is_churn. If KKBox wants to determine if a user is likely to churn, it should focus on the statistically significant covariates. However, because the confidence level is 95%, we would expect 5% of the coefficients to be significant due to random chance alone. Therefore, the user of this analysis should keep in mind that a small proportion of the significant coefficients determined by the model may not be significant in the real population.

Next, the same model was fit on the test set and statistically significant coefficients were determined. The values are listed in Table 3 of the Appendix.

There were some covariates that were statistically significant on the test set but not on the training set: city_14, city_17, and city_20 (city of users). There was a variable that was significant on the training set but not on the test set, bd (age). Again, this could be a sign that multiple hypothesis testing could be affecting the analysis. Some covariates that were considered

significant in one dataset and not in another could have been determined from chance alone. For example, age could have happened to explain the variation in the response variable well in the training set, but not in the population model; it may not be truly significant. On the other hand, there were many covariates that were significant in both datasets, meaning they are more likely to be associated with the user churning. This also suggests that the model was not too overfit to the training set.

In order to estimate confidence intervals for regression coefficients, bootstrap was performed on the training set. This may have resulted in vulnerability to post-selection inference. The final model was chosen because it did well on the training set and the confidence intervals were created based on the same dataset - this means the coefficients would seem to be more significant than what the true population model suggests. This may have resulted in incorrect estimates of the confidence interval.

Estimation of confidence interval by the normal interval approach was reasonable because the distribution of the coefficients roughly follow a normal distribution. The confidence intervals for the coefficients are different from the standard regression output from R - for the coefficients on payment_plan_methods, R gave very large confidence intervals. This may be a result of an incorrect assumption that the population model is linear normal with homoscedastic error. On the other hand, bootstrap does not require any assumptions. Each coefficient's bootstrapped confidence intervals and estimations from R can be found in Table 5 in the Appendix.

The chosen model already included all covariates given in the dataset. Therefore, to determine if the same covariates are statistically significant in a different model trained on the same data, a new model was created using forward stepwise regression and AIC. The statistically significant coefficients based on the 95% confidence level are in Table 4 in the Appendix.

Of the new model, total_secs was the only new statistically significant covariate compared to the model built with all covariates. Some changes in the statistical significance of each coefficient is expected, as this comes from fitting a specific model to a specific data set. However, this could also imply collinearity or multicollinearity in the model as a model that includes multiple collinear covariates could assign less significance to the covariates. This would affect the significance of the coefficients.

Collinearity in covariates can potentially cause problems in the analysis. In fact, collinearity was observed from the pairwise correlation plot. Covariates that had high correlations (>0.5) can be found in the Appendix. Also, standard errors for some of the regression coefficients were very high, which makes it reasonable to suspect collinearity. To determine whether multicollinearity is affecting the results, the variance inflation factor (VIF) was computed for the model. Through this analysis, it was seen that memdur, diffdate, num_100, plan_list_price, payment_method_id, registered_via were multicollinear with other variables.

Another potential problem is that because a significance level of 5% was used, the null is falsely rejected about 5% of the time (false positive). To correct for the high occurrence of false positives, the Bonferroni correction can be applied. With the correction, a coefficient is

considered significant only if its p-value is less than 0.05/p, or 0.0007. With this threshold, only 8 out of 73 covariates would be statistically significant, compared to 13 out of 73 without the correction. This ensures that the probability of declaring even one false positive is no more than 5%.

Because the models were trained and selected based on the training data, there is potential bias in determining which covariates are significant. Even though cross-validation error was used as the metric to choose the final model, the fact that both model training and selection were done on the same dataset can result in regression coefficients with higher significance than what the population model suggests. Therefore, the significance of the coefficients may be overestimated and some covariates that are determined to be significant may not be significant in reality.

For relationships that are statistically significant, causal effects can be assigned if there is no sampling bias. This is ensured if the assignment to treatment is uncorrelated with the outcome through randomized assignment. For the KKBox dataset, no values are randomly assigned and most user attributes are self-selected. Therefore, there are no relationships that can be declared as causal with either outcome. Only age is not self-selected; however, because it is not an assigned treatment or control, no statements of causal effects can be made. Even if there were no other confounding covariates, it would not be possible to establish causal relationships.

Discussion

The final regression model can be used to predict the daily total number of seconds a user will listen to music on KKBox. Because KKBox pays its artists based on the streams, the more seconds of music a user listens to, the more money he/she will cost the company. It can use the predictions to more accurately determine its cost and make decisions about its revenue strategy. For example, it can determine how much to charge for different subscription plans or how much to pay the artists. It is not as important for KKBox to know what specific user attributes are associated with listening to more music because it is not trying to make changes to such characteristics to control the outcome. Therefore, the regression model is a predictive model.

In terms of potential pitfalls, it may not be feasible for KKBox to change the pricing for different plans very frequently, even if the output from the model suggests so. Additionally, decreasing the amount paid to artists may impact the company's image negatively, as payment to artists is a social issue in the music realm. Lastly, there may be errors in prediction the company will need to take into account when considering such huge changes.

The classification model will be used to understand what factors are closely related to a user churning. Therefore, this model will be used for inference. KKBox can then use this information to target its user retention campaigns to those who are in danger of churning rather than spending money on targeting users who are not in danger of leaving KKBox.

A potential pitfall of the classification model is that there is room for false inference. As mentioned, collinearity, multiple hypothesis testing, post-selection inference, and other things may have affected the value or significance of the regression coefficients. Additionally, different users may require different user-retention efforts, so tailoring campaigns to specific attributes

may be a better strategy than solely identifying which groups to focus retention efforts on with the model.

The linear regression model does not need to be refit often, as KKBox does not need to understand how much music users are listening to in real time. However, the model should be refit every season or so to take into account the effects of seasonality on user usage (to monitor usage, calculate costs, and determine if it is necessary to implement new business strategies). Additionally, it should be refitted if KKBox or its competitors have promotions or if a new business strategy is implemented, such as a change in the subscription plan. Therefore, the model should be refitted every few months or whenever there is a change in the business strategies of KKBox or its competitor.

The refitting process for the classification model would be similar to the regression one. The refitting does not need to be done in real-time as it would incur high costs; the company just needs to detect sudden changes. Therefore, it would be reasonable for KKBox to refit the model every few months to monitor the churn rate, in addition to when considering customer retention strategy changes, subscription/cost changes, or any other major company-wide changes. As it is vital for the company to know what is causing its users to churn with more up-to-date information, the refitting for monitoring purposes may be done more frequently (e.g. every 3 months for classification and every 5 months for regression).

It would be necessary for those using the models in this analysis to know about certain choices made in the process. First, only a subset of the given data was considered after the data cleaning process. Duplicate observations for users with more than 1 subscription transaction (sign ups for subscription plan) were removed so that each user contributes to one data point at maximum. Also, 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 removed. It should be noted that removing data points without gender will overestimate the churn rate because most users with missing gender did not churn, as shown in Figure 1 in the Appendix. Also, the data for the classification model was specifically chosen so that the number of users who churned and who did not churn were roughly equally represented. This is not representative of the population model in which there was a significantly higher proportion of users who did not churn. Furthermore, because artificially sampled data, rather than the randomly sampled data, was used to train the model, the quality of the classifier could have worsened.

In terms of multiple hypothesis testing, no correction was made to reduce the number of false positives in testing the regression coefficients against the null of 0. Therefore, 5% of the coefficients may be falsely determined to be statistically significant. Additionally, if the model is refitted multiple times, the significance of a variable (whether significant or not) will be incorrect 5% of the time.

Both models will have issues with post-selection inference as they were fitted, selected, and evaluated using the same training set. The significance of the coefficients in the models will be

higher than they are in the population model. Therefore, specific coefficients that seem to be significantly associated with the outcome may not be in the population.

Some changes to the data collection process would greatly help the predictive ability of the models. 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. Because the current process allows users to self-report a lot of data, such as gender and age, the dataset has unreliable values; the veracity of the users' inputs cannot be confirmed. Users should not be allowed to leave inputs blank and the input must be reasonable (i.e. age must be between 0 and 120).

The addition of new covariates would also greatly aid the analysis. It would be helpful to know how many songs the user listened to that were available exclusively on KKBox. If a user's favorite artist is only available on KKBox, the user may be less likely to leave KKBox for another music streaming service. It would also be helpful to understand when users listen to music during the day. If users who are likely to churn listen to music during a certain time, it would be beneficial to send targeted retention advertisements during that time. Additionally, if users are actively using KKBox on multiple devices, this increased engagement could imply that they listen to more music and are less likely to churn. So it would be interesting to have data on the number of devices each user uses KKBox on. Lastly, KKBox could collect the genre of music a user listens to, which may help with both prediction of total daily seconds streamed and whether the user churns.

Certain strategies could be changed the next time this analysis is done. For example, different modeling techniques could be used to build more accurate models for predicting the daily seconds streamed. However, for classification task, the models should be kept simple enough to allow for inference as this analysis does. Additionally, the subset of data for analysis could be chosen in a different way. Currently, this analysis assumes that the is_churn column of the dataset is accurate. However, KKBox may have derived wrong values for this column as it is based on the subscription-plan transactional history of the users; it may be better to extract this information directly from the date users canceled their subscription.Additionally, missing data could have been handled differently. In this analysis, missing data was deleted; however, this introduces bias into the dataset. For example, it was seen that there was a relationship between missing gender data and the is_churn of a user. Because removing these observations would overestimate the churn rate, the missing data in gender could have been imputed instead. Additionally, multicollinearity of covariates should have been considered in the model building process rather than just in interpretation of the chosen models. Lastly, the analysis could be performed taking seasonality into account. Rather than having one data point per user, each transaction could be tracked to see the usage over time, which could be useful in predicting user churn rate. This could also help determine how the daily total seconds streamed is affected by seasonality.

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

Correlated Variables

For classification, the values that are correlated are bd & city, is_churn & city, plan_list_price & payment_plan_method, actual_amount_paid & payment_plan_method, num_25 & num_50 & num_75 & num_985 & num_100, and num_unq & total_secs.

Tables

	Regression	**Classification**
Estimate of Test Error	570.7867	0.2478048
Test Error	1096.4500	0.3151071

Table 1: Comparison of Estimate of Test Error and Actual Test Error for Regression and Classification

Variable	**β Estimate**	**Significance**
bd	3.536e-03	0.03045
registered_via4	2.234e-01	4.09e-05
registered_via9	-2.730e-01	5.24e-12
payment_plan_days	1.947e-02	0.00133
plan_list_price	2.738e-02	< 2e-16
actual_amount_paid	-2.405e-02	< 2e-16
is_auto_renew1	-1.544e+00	< 2e-16
is_cancel1	4.837e+00	< 2e-16
diffdate	-1.656e-02	< 2e-16
memdur	1.671e-02	< 2e-16

Table 2: statistically significant coefficients from the model fitted on the training set. Statistical significance set at the 95% level. In this table, registered_via4 and registered_via9 means that they user registered using the 4th and 9th methods, respectively, and is_auto_renew1 and is_cancel1 means the user did auto-renew and did cancel, respectively.

Variable	β Estimate	Significance
city14	4.197e-01	0.02826
city17	5.876e-01	0.04334
city20	1.485e+00	0.0227
city22	4.171e-01	0.01326
registered_via4	3.280e-01	0.00279
registered_via9	-1.959e-01	0.01345
payment_plan_days	3.747e-02	0.01510
plan_list_price	2.663e-02	2.93e-16
actual_amount_paid	-2.451e-02	< 2e-16
is_auto_renew1	-1.563e+00	< 2e-16
is_cancel1	4.521e+00	< 2e-16
diffdate	-1.640e-02	< 2e-16
memdur	1.654e-02	< 2e-16

Table 3: statistically significant coefficients from the model fitted on the test set. Statistical significance set at the 95% level.

Variable	β Estimate	Significance
is_cancel1	4.834e+00	< 2e-16
payment_plan_days	1.963e-02	0.00121
is_auto_renew1	-1.546e+00	< 2e-16
actual_amount_paid	-2.401e-02	< 2e-16
plan_list_price	2.730e-02	< 2e-16
memdur	1.671e-02	< 2e-16
diffdate	-1.656e-02	< 2e-16
registered_via4	2.242e-01	3.64e-05

registered_via9	-2.754e-01	2.85e-12
total_secs	-6.486e-06	0.00757
bd	3.797e-03	0.01953

Table 4: statistically significant coefficients from the new AIC model fitted on the training set. Statistical significance set at the 95% level.

Classification Model Coefficient Confidence Intervals

Covariate	**95% Confidence Interval**	**R output**
intercept	[3.478514, 9.855708]	[-2317.967, 2330.994]
city3	[-0.1989906, 0.3492136]	[-0.2042508, 0.3506775]
city4	[-0.1098351, 0.2073682]	[-0.1183485, 0.2101865]
city5	[-0.1294498, 0.1848089]	[-0.1310130, 0.1834917]
city6	[-0.1742569, 0.1888138]	[-0.1695208, 0.1860549]
city7	[-0.3677026, 0.3523989]	[-0.3698457, 0.3703319]
city8	[-0.2715885, 0.1912303]	[-0.2846411, 0.1964909]
city9	[-0.2476231, 0.2141843]	[-0.2510531, 0.2128659]
city10	[-0.3236897, 0.1842462]	[-0.3217497, 0.1909351]
city11	[-0.2786691, 0.1935316]	[-0.2727641, 0.1911869]
city12	[-0.3546784, 0.09659035]	[-0.34678375, 0.09205256]
city13	[-0.1656653, 0.1401322]	[-0.1697374, 0.1410654]
city14	[-0.2424076, 0.1248771]	[-0.2524901, 0.1226313]
city15	[-0.1708602, 0.1516799]	[-0.1766477, 0.1544261]
city16	[-0.6841555, 0.4497288]	[-0.6642165, 0.4384029]
city17	[-0.427381, 0.1542562]	[-0.4249907, 0.1498062]
city18	[-0.296427, 0.1930155]	[-0.2984158, 0.1882577]
city19	[-3.03781, 1.702383]	[-2.184505, 1.136269]
city20	[-1.608918, -0.005075291]	[-1.6627087, 0.1158868]

city21	[-0.4461894, 0.1175293]	[-0.4424172, 0.1202305]
city22	[-0.2092507, 0.1303257]	[-0.2092776, 0.1228727]
bd	[0.0003577328, 0.006725647]	[0.0003336747, 0.0067390358]
gender(Female)	[-0.05320691, 0.05399893]	[-0.05196573, 0.05320193]
registered_via4	[0.1107085, 0.3309254]	[0.1166496, 0.3300839]
registered_via7	[-0.1589585, 0.2427395]	[-0.1629636, 0.2519825]
registered_via9	[-0.3548757, -0.1939484]	[-0.3506208, -0.1954731]
registered_via13	[-1.010493, 1.111329]	[-0.9582535, 1.1637895]
payment_method_id6	[-16.35891, -1.182122]	[-4025.294, 4004.798]
payment_method_id8	[-4.695772, 3.236778]	[-2810.893, 2809.401]
payment_method_id10	[-23.33139, 4.923824]	[-2331.989, 2316.974]
payment_method_id11	[-29.43975, 0.9655695]	[-2333.296, 2315.667]
payment_method_id12	[0.5843709, 6.663878]	[-2344.941, 2351.771]
payment_method_id13	[-3.68572, 1.92192]	[-2346.580, 2344.381]
payment_method_id14	[-12.44127, -5.265159]	[-2333.060, 2315.902]
payment_method_id15	[-11.29047, 0.9621414]	[-2330.309, 2318.652]
payment_method_id16	[-11.71265, -5.308808]	[-2332.807, 2316.155]
payment_method_id17	[-9.06329, 5.278309]	[-2327.444, 2321.518]
payment_method_id18	[-11.9091, -5.281473]	[-2332.870, 2316.092]
payment_method_id19	[-12.32085, -5.943315]	[-2333.419, 2315.543]
payment_method_id20	[1.572112, 7.435452]	[-2330.909, 2339.344]
payment_method_id21	[-11.17551, -4.769185]	[-2332.284, 2316.678]
payment_method_id22	[-4.347481, 1.296913]	[-2336.624, 2333.152]
payment_method_id23	[-12.3177, -5.947902]	[-2333.434, 2315.528]

payment_method_id26	[-13.039, 6.910732]	[-2329.278, 2319.685]
payment_method_id27	[-12.39457, -5.992223]	[-2333.474, 2315.487]
payment_method_id28	[-11.20412, -4.936702]	[-2332.386, 2316.575]
payment_method_id29	[-11.64109, -5.337259]	[-2332.804, 2316.157]
payment_method_id30	[-10.86254, -4.593099]	[-2332.038, 2316.923]
payment_method_id31	[-13.1354, -6.841369]	[-2334.289, 2314.672]
payment_method_id32	[-9.441732, -1.762781]	[-2330.137, 2318.824]
payment_method_id33	[-12.33709, -6.083817]	[-2333.519, 2315.442]
payment_method_id34	[-12.52679, -6.257977]	[-2333.699, 2315.262]
payment_method_id35	[-11.27125, 11.2729]	[-2328.103, 2320.860]
payment_method_id36	[-11.02764, -4.740337]	[-2332.200, 2316.761]
payment_method_id37	[-12.03131, -5.758483]	[-2333.204, 2315.757]
payment_method_id38	[-10.99345, -4.741719]	[-2332.182, 2316.779]
payment_method_id39	[-11.15242, -4.893033]	[-2332.334, 2316.627]
payment_method_id40	[-11.05092, -4.782285]	[-2332.227, 2316.734]
payment_method_id41	[-11.08366, -4.830115]	[-2332.269, 2316.692]
payment_plan_days	[0.004969881, 0.03586786]	[0.007582044, 0.031351927]
plan_list_price	[0.02429725, 0.03068264]	[0.02441072, 0.03034722]
actual_amount_paid	[-0.02652533, -0.02198998]	[-0.02623687, -0.02186920]
is_auto_renew1	[-1.664186, -1.426502]	[-1.661453, -1.426654]
is_cancel1	[4.654533, 5.042907]	[4.659879, 5.013903]
num_25	[-0.002249802, 0.003465437]	[-0.002148379, 0.003391790]
num_50	[-0.01120641, -0.01120641]	[-0.011169330, 0.005270128]
num_75	[-0.01350168, 0.02375761]	[-0.01286051, 0.02327907]
num_985	[-0.02620572, 0.003185524]	[-0.022779452, 0.002023647]

num_100	[-0.003917655, 0.003509393]	[-0.003874312, 0.003441620]
num_unq	[-0.0008379772, 0.002805918]	[-0.000773805, 0.002799401]
total_secs	[-2.158667e-05, 1.105545e-05]	[-2.141767e-05, 1.082440e-05]
diffdate	[-0.018548, -0.0145406]	[-0.01783467, -0.01528068]
memdur	[0.0146936, 0.01869844]	[0.01543292, 0.01798402]

Table 5: each coefficient's bootstrapped confidence interval and estimate from R for the logistic regression model.

Graphs

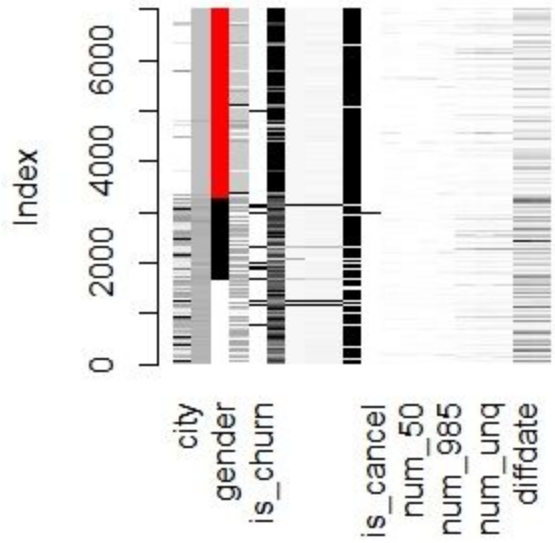

Figure 1: Sorted (and Missing) Gender Values and Missingness in Other Variables

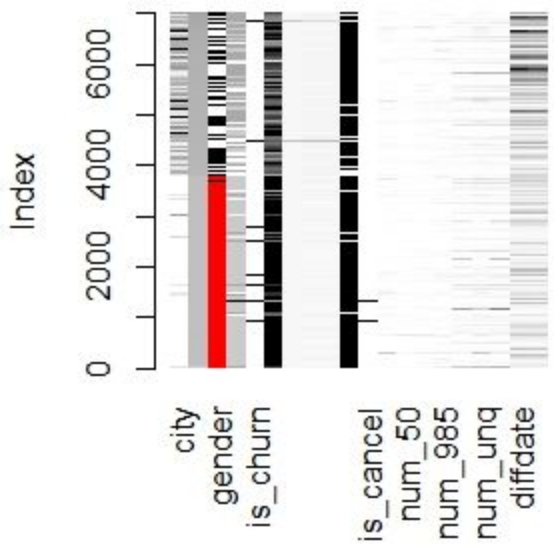

Figure 2: Sorted Age Values and Missingness in Other Variables (Variable to the Left of Gender is Age)

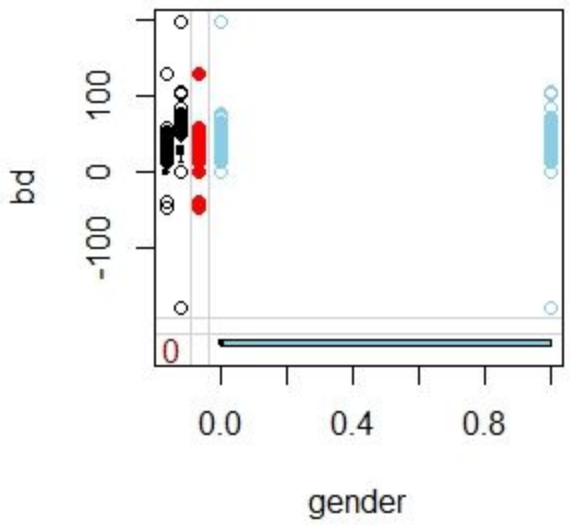

Figure 3: Margin Plot of Age and Gender

Code

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

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)

select <- dplyr::select
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),
7000), ]
sample[sample$gender ==
"female","gender"] <- 1
sample[sample$gender == "male","gender"]
<- 0
drops <- c("msno")
sample = sample[ , !(names(sample) %in%
drops)]

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
drops <- c("msno")
goodageG = goodageG[ ,
!(names(goodageG) %in% drops)]
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","transacti
on_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)*fi
naltrain$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
set.seed(123)
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[["registratio
n_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[["membersh
ip_expire_date"]]), format = "%Y%m%d")
InitDate =
as.Date(as.character(x.sub.train[["registratio
n_init_time"]]), format = "%Y%m%d")
x.sub.train$memdur = ExpDate - InitDate

# Drop msno and dates
drops <-
c("msno","registration_init_time","transacti
on_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)
```

```
# Part 2
# 1. Prediction on test set
# Apply same cleaning method to test set
# Drop msno and dates
drops <-
c("msno","registration_init_time","transacti
on_date","membership_expire_date","date")
finaltest = finaltest[ , !(names(finaltest)
%in% drops)]
# Gender
finaltest[finaltest$gender ==
"female","gender"] <- 1
finaltest[finaltest$gender ==
"male","gender"] <- 0
# Factors
finaltest$city <- as.factor(finaltest$city)
finaltest$registered_via <-
as.factor(finaltest$registered_via)
finaltest$is_churn  <-
as.factor(finaltest$is_churn)
finaltest$payment_method_id  <-
as.factor(finaltest$payment_method_id)
finaltest$is_auto_renew <-
as.factor(finaltest$is_auto_renew)
finaltest$is_cancel <-
as.factor(finaltest$is_cancel)
finaltest$diffdate =
as.numeric(finaltest$diffdate)
finaltest$memdur =
as.numeric(finaltest$memdur)

# Regression best model
# test_matrix =
model.matrix(total_secs~.-total_secs,data =
finaltest)
# Standardization: keep same mean from
train set?
# Wait..do we want to center Y_test tho?
centered_Y_test =
(finaltest[,18]-mean(finaltrain$total_secs))
library(glmnet)

# # standardize X matrix
# # "standardization" function
# # train = finaltrain[,i] and test =
finaltest[,i]
# standardize <- function(train,test) {
#   # for categorical
#   if (class(test)=="factor"){
#     testvals = as.numeric(levels(test))[test]
#     trainvals =
as.numeric(levels(train))[train]
#     (testvals-mean(trainvals))/sd(trainvals)
#   }
#   else{
#     testvals = as.numeric(test)
#     trainvals = as.numeric(train)
#     (testvals-mean(trainvals))/sd(trainvals)
#   }
#   }
#
# # Standardize function is working!
# a=
standardize(finaltrain[,15],finaltest[,15])
#
# test_matrix1 =
model.matrix(total_secs~.-total_secs,data =
finaltest)
#
# #testNoOutput = finaltest[,-18]
# #trainNoOutput = finaltrain[,-18]
# output <-
matrix(ncol=length(colnames(test_matrix1))
, nrow=nrow(finaltest))
# output[,1] = 1
# for (i in
seq(2,length(colnames(test_matrix1)))){
```

```
#
output[,i]=standardize(train_matrix[,i],test_
matrix1[,i])
# }
# # ans = mapply(standardize, finaltrain[,1],
finaltest[,1])


# #output = cbind(output,
finaltest$total_secs)
test_matrix1 =
model.matrix(total_secs~.-total_secs,data =
output)
# predict(fit_lasso_cv, output,s =
"lambda.min", link='link')
reg_testerr = mean(abs(centered_Y_test -
predict(fit_lasso_cv, test_matrix1, s =
"lambda.min", link='link')))


# Classification cleaning on test set (test set
was different due to proportion of 0s and 1s)
# Difference between membership start date
and when daily music user log was taken
# (In days)
date =
as.Date(as.character(x.sub.test[["date"]]),
format = "%Y%m%d")
MusicDate =  as.Date(date, "%Y%m%d")
MemberDate =
as.Date(as.character(x.sub.test[["registration
_init_time"]]), format = "%Y%m%d")
MemberDate =  as.Date(MemberDate,
"%Y%m%d")
x.sub.test$diffdate = MusicDate -
MemberDate

# Membership Duration (in days)
ExpDate =
as.Date(as.character(x.sub.test[["membershi
p_expire_date"]]), format = "%Y%m%d")
InitDate =
as.Date(as.character(x.sub.test[["registration
_init_time"]]), format = "%Y%m%d")
x.sub.test$memdur = ExpDate - InitDate

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

x.sub.test$city <- as.factor(x.sub.test$city)
x.sub.test$registered_via <-
as.factor(x.sub.test$registered_via)
x.sub.test$payment_method_id  <-
as.factor(x.sub.test$payment_method_id)
x.sub.test$is_auto_renew <-
as.factor(x.sub.test$is_auto_renew)
x.sub.test$is_cancel <-
as.factor(x.sub.test$is_cancel)
x.sub.test[x.sub.test$gender ==
"female","gender"] <- 1
x.sub.test[x.sub.test$gender ==
"male","gender"] <- 0

# Best classification model
summary(LogModel)
model_glm_pred =
ifelse(predict(LogModel,x.sub.test, type =
"response") > 0.5, 1, 0)
calc_err = function(actual, predicted) {
  difference = actual - predicted
  # Pred 0, actual 1
  falseN = 2*sum(difference[difference>0])
  # Pred 1, actual 0
```

```
  falseP = -1*sum(difference[difference<0])
  (falseN+falseP)/length(actual)
}
class_testerr = calc_err(actual =
x.sub.test$is_churn, predicted =
model_glm_pred)


# 2c) Bootstrap
# Resample from training set
# Fit model and get coefficient 1000 times
# Make histogram of coefficient
# Bootstrap on training set is okay - mention
post-inference issues
set.seed(123)
coeffs = rep( list(list()),
length(names(LogModel$coefficients)))
#bd_coeffs <- rep(NA, 100)
for (i in seq(1,1000)){
  bootstrap =
x.sub.train[sample(nrow(x.sub.train),size=nr
ow(x.sub.train),replace=TRUE),]
  LogModel_bs <- glm(is_churn ~.,
data=bootstrap, family="binomial")
  ind = 0
  for (cov in
names(LogModel$coefficients)){
    ind = ind+1
    coeffs[[ind]][i] =
LogModel_bs$coefficients[cov]
  }
}

# Create confidence intervals
cis = rep( list(list()),
length(names(LogModel$coefficients)))
for (j in seq(1,length(coeffs))){
  coefflst = unlist(coeffs[j], use.names =
FALSE)
  coefflst[is.na(coefflst)] <- 0
  std = sd(coefflst)
  cis[[j]][1] = mean(coefflst)-1.96*std
  cis[[j]][2] = mean(coefflst)+1.96*std
}

hist(coefflst)

# Show CI values
for (i in seq(1,length(cis))){
  print(i)
  print(c(cis[[i]][1],cis[[i]][2]))
}

# Part 2 b&c&d
#######
#find p-values
summary(LogModel)

#fit model on test set
set.seed(123)
LogModelTest <- glm(is_churn ~.,
data=x.sub.test, family="binomial")
summary(LogModelTest)


#AIC to make new model
fm.lower = glm(data = x.sub.train, is_churn
~ 1, family = "binomial")
fm.upper = glm(data = x.sub.train, is_churn
~ ., family = "binomial")
model = step(fm.lower,
             scope = list(lower = fm.lower,
                          upper = fm.upper),
             direction = "forward")
summary(model)

vif(LogModel)
vif(model)
```

#high VIF score implies high collinearity -
registered_viw, payment_method_id,
plan_list_price, num_100, total_secs,
diffdate, memdur