# Main-Proj5

*Group 2*

*12/5/2017*

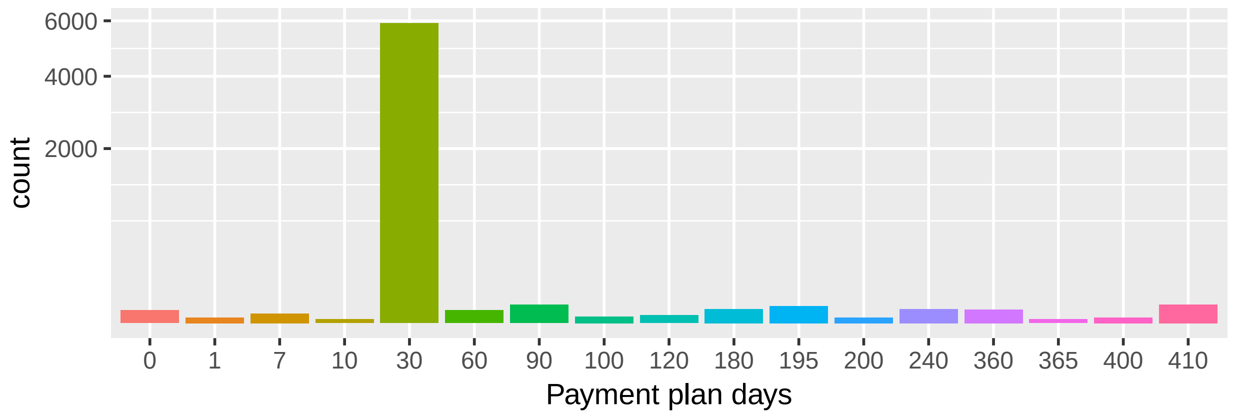
```
packages.used<-c("xgboost","adabag","data.table","Matrix","dplyr","nnet",
              "glmnet","randomforest","gbm","caret","e1071","ggplot2","readr","tidyr","forcats")

# Check packages that need to be installed.
packages.needed<-setdiff(packages.used,
                         intersect(installed.packages()[,1],
                                   packages.used))
# Install additional packages
if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE,
                   repos='http://cran.us.r-project.org')
}
# Load packages
library("xgboost")
library("adabag")
library("data.table")
library("Matrix")
library("dplyr")
library("nnet")
library("glmnet")
library("randomForest")
library("e1071")
library("gbm")
library("caret")
library("ggplot2")
library("readr")
library("gridExtra")
library("tidyr")
library("forcats")
```

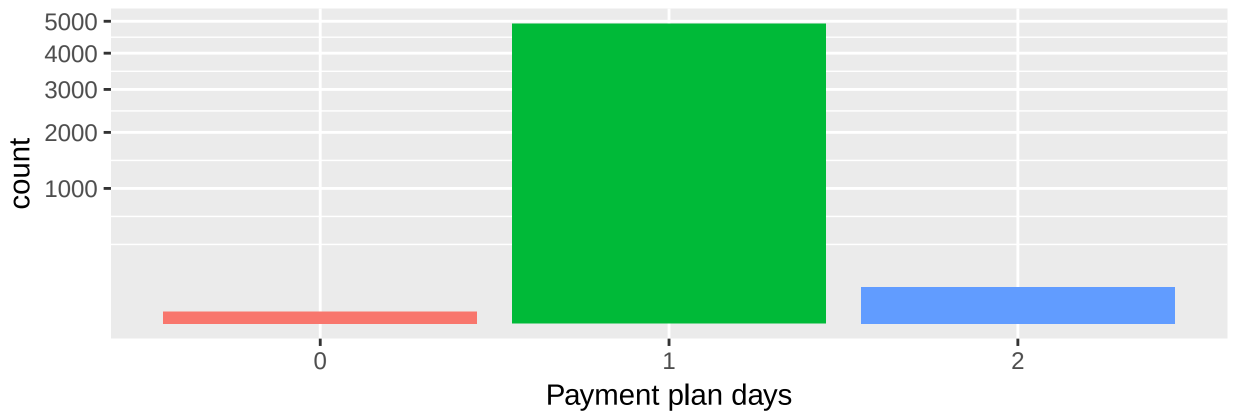
### Step 1 Features Analysis

```
# Set your working directory as where the Main File is located
setwd("~/Desktop/fall2017-project5-proj5-grp2-master/doc")


########################## Features Analysis #############################
##########################################################################

######## Data Cleaning #######
load("../data/trans.before.aggregate.RData")
load("../data/new.transaction.RData")
sample_id<-read_csv("../data/ids_subset.csv") # sample user
train <- read_csv("../data/train_v2.csv") # Training data
final_train <- merge(sample_id, train)
```

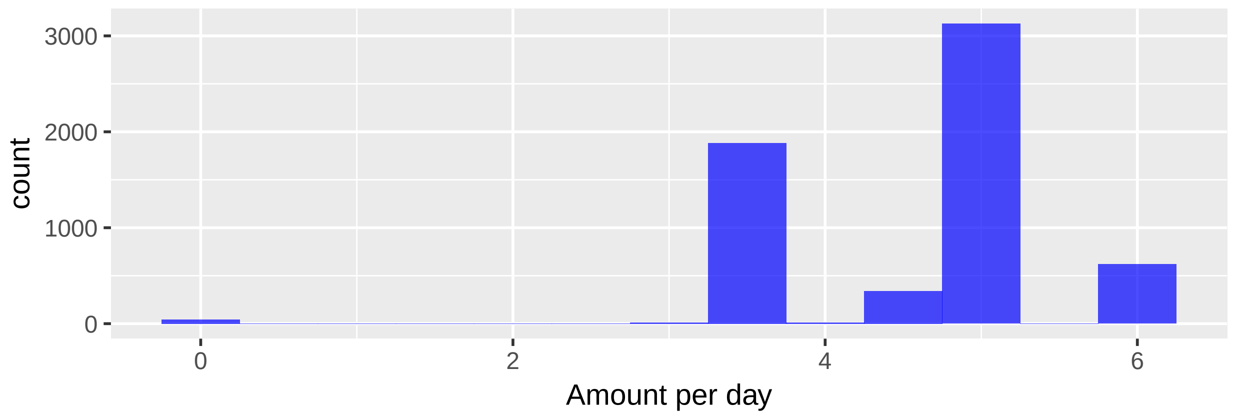
```r
combined = sub_trans
#View(combined)
combined$is_discount <- as.factor(ifelse(combined$actual_amount_paid<combined$plan_list_price,1,0))
### auto_renew & is_cancel
total_renew = as.data.frame(aggregate(combined$is_auto_renew,by=list(msno=combined$msno),sum))
colnames(total_renew) <- c("msno","total_renew")

total_cancel = as.data.frame(aggregate(combined$is_cancel,by=list(msno=combined$msno), sum))
colnames(total_cancel) <- c("msno","total_cancel")

final_combined = merge(sample_id,total_renew)
final_combined = merge(final_combined, total_cancel)
final_combined = final_combined[order(final_combined$msno),]
### is_discount & sum_auto_renew update
load("../data/data.all.Rdata")
data.all$is_discount <- ifelse(data.all$discount==0, 0, 1)
data.all$sum_auto_renew <- ifelse(data.all$sum_auto_renew>=3, 3, data.all$sum_auto_renew)
#Further Data cleaning for EDA
combined$is_auto_renew<-as.factor(combined$is_auto_renew)
combined$is_cancel<-as.factor(combined$is_cancel)
combined$payment_method_id<-as.factor(combined$payment_method_id)


######## EDA For Transactions #######
#payment_plan_days
p1<-sub_trans %>%
  mutate(payment_plan_days = factor(payment_plan_days)) %>%
  ggplot(aes(payment_plan_days, fill = payment_plan_days)) +
  geom_bar() +
  scale_y_sqrt() +
  theme(legend.position = "none") +
  labs(x = "Payment plan days")
#Convert payment_plan_days to categorical:
p2<-new.trans %>%
  ggplot(aes(payment_plan_days, fill = payment_plan_days)) +
  geom_bar() +
  scale_y_sqrt() +
  theme(legend.position = "none") +
  labs(x = "Payment plan days")
#amt_per_day
p3<-ggplot(sub_trans) +
  geom_histogram(aes(x=amt_per_day),binwidth  = 0.5, fill="blue",alpha=0.7)+
  theme(legend.position = "none") +
  labs(x = "Amount per day")
#plan_list_price
p4<-sub_trans %>%
  mutate(plan_list_price = factor(plan_list_price)) %>%
  ggplot(aes(plan_list_price, fill = plan_list_price)) +
  geom_bar() +
  scale_y_sqrt() +
  theme(legend.position = "none") +
  labs(x = "Plan list price")
```

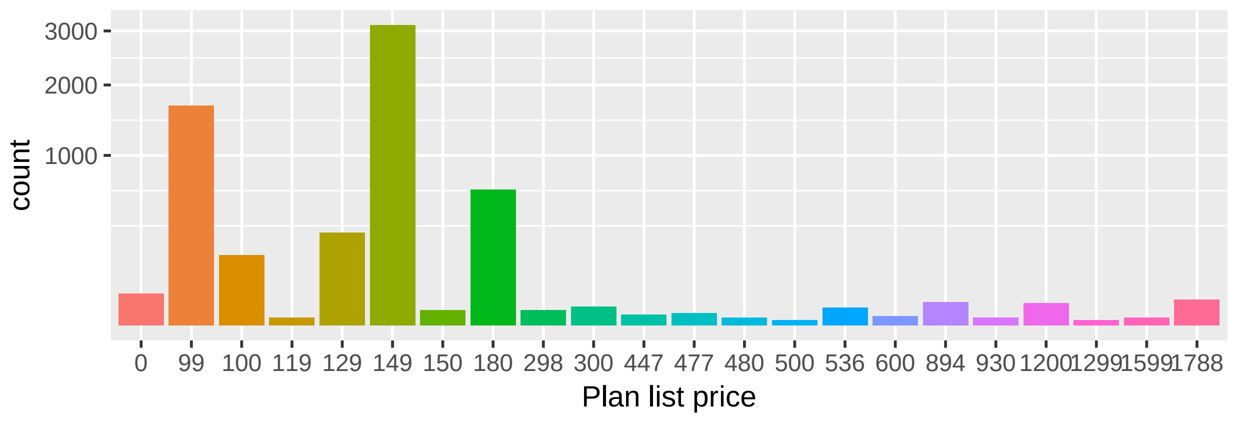
```
### is_cancel
p5<-combined %>%
  ggplot(aes(is_cancel, fill=is_cancel))+
  geom_bar() +
  theme(legend.position = "none")+
  labs(x = "is_cancel")
### auto_renew
p6<-combined %>%
  ggplot(aes(is_auto_renew,fill=is_auto_renew))+
  geom_bar() +
  theme(legend.position = "none")+
  labs(x = "is_auto_renew")

Mode <- function(x){
  ux <- unique(x)
  ux[which.max(tabulate(match(x,ux)))]
}

get_binCI <- function(x,n) {
  rbind(setNames(c(binom.test(x,n)$conf.int),c("lwr","upr")))
}
### auto renew & is churn
p7<-combined %>%
  group_by(msno) %>%
  summarise(is_auto_renew = Mode(is_auto_renew))%>%
  inner_join(train,by="msno") %>%
  group_by(is_auto_renew,is_churn)%>%
  count() %>%
  spread(is_churn,n)%>%
  mutate(frac_churn=`1`/(`1`+`0`)*100,
         lwr = get_binCI(`1`,(`1`+`0`))[[1]]*100,
         upr = get_binCI(`1`,(`1`+`0`))[[2]]*100
  )%>%
  ggplot(aes(is_auto_renew,frac_churn,fill=is_auto_renew))+
  geom_col()+
  geom_errorbar(aes(ymin=lwr,ymax=upr),width=0.5,size=0.7,color="gray30")+
  theme(legend.position = "none")+
  labs(x="Auto renew",y="Churn[%]")

### is cancel & is churn
p8<-combined %>%
  group_by(msno) %>%
  summarise(is_cancel = Mode(is_cancel))%>%
  inner_join(train,by="msno") %>%
  group_by(is_cancel,is_churn)%>%
  count() %>%
  spread(is_churn,n)%>%
  mutate(frac_churn=`1`/(`1`+`0`)*100,
         lwr = get_binCI(`1`,(`1`+`0`))[[1]]*100,
         upr = get_binCI(`1`,(`1`+`0`))[[2]]*100
  )%>%
  ggplot(aes(is_cancel,frac_churn,fill=is_cancel))+
  geom_col()+
  geom_errorbar(aes(ymin=lwr,ymax=upr),width=0.5,size=0.7,color="gray30")+
  theme(legend.position = "none")+
  labs(x="Cancelled",y="Churn[%]")
### member duration
p9<-new.trans %>%
  mutate(member_duration = factor(member_duration)) %>%
  ggplot(aes(member_duration, fill = member_duration)) +
  geom_bar() +
  scale_y_sqrt() +
  theme(legend.position = "none") +
  labs(x = "member duration")
grid.arrange(p1,p2)
```



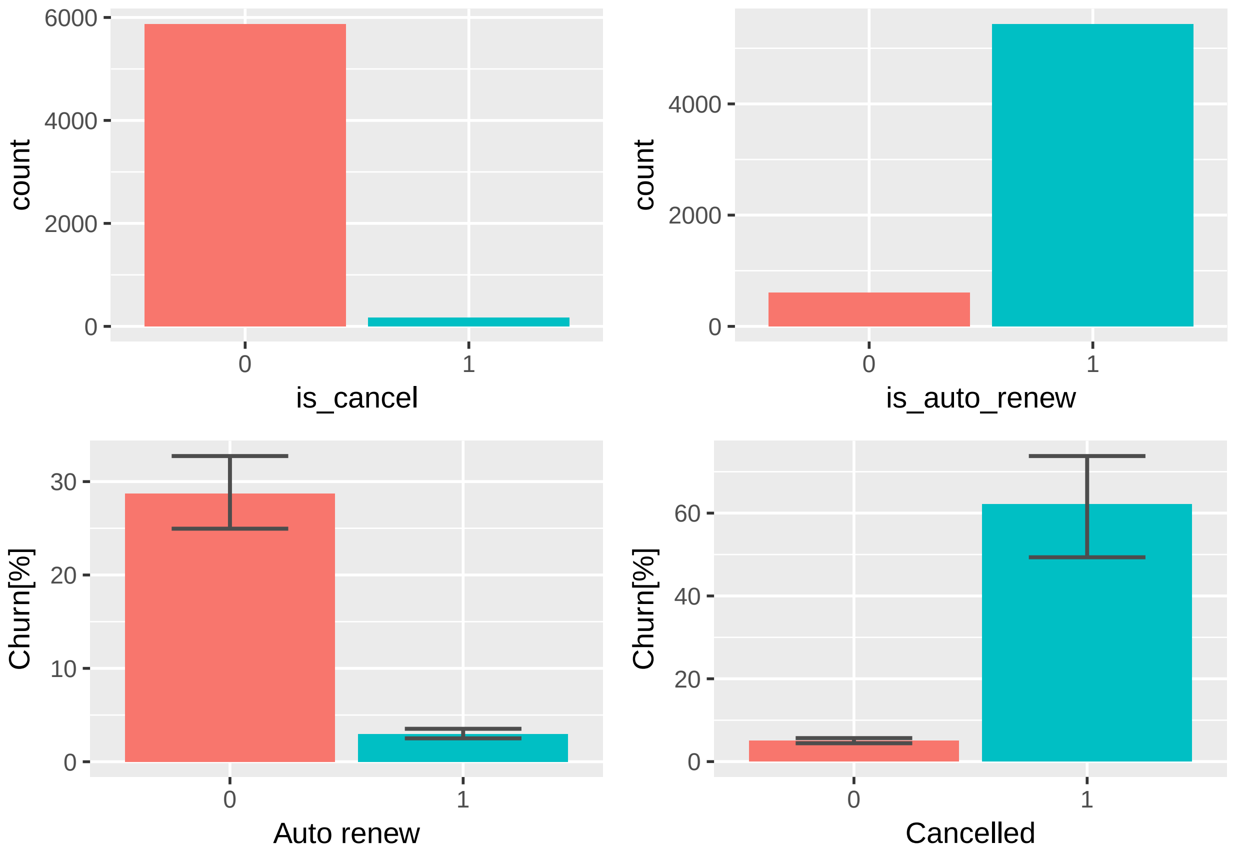


```
grid.arrange(p3,p4)
```

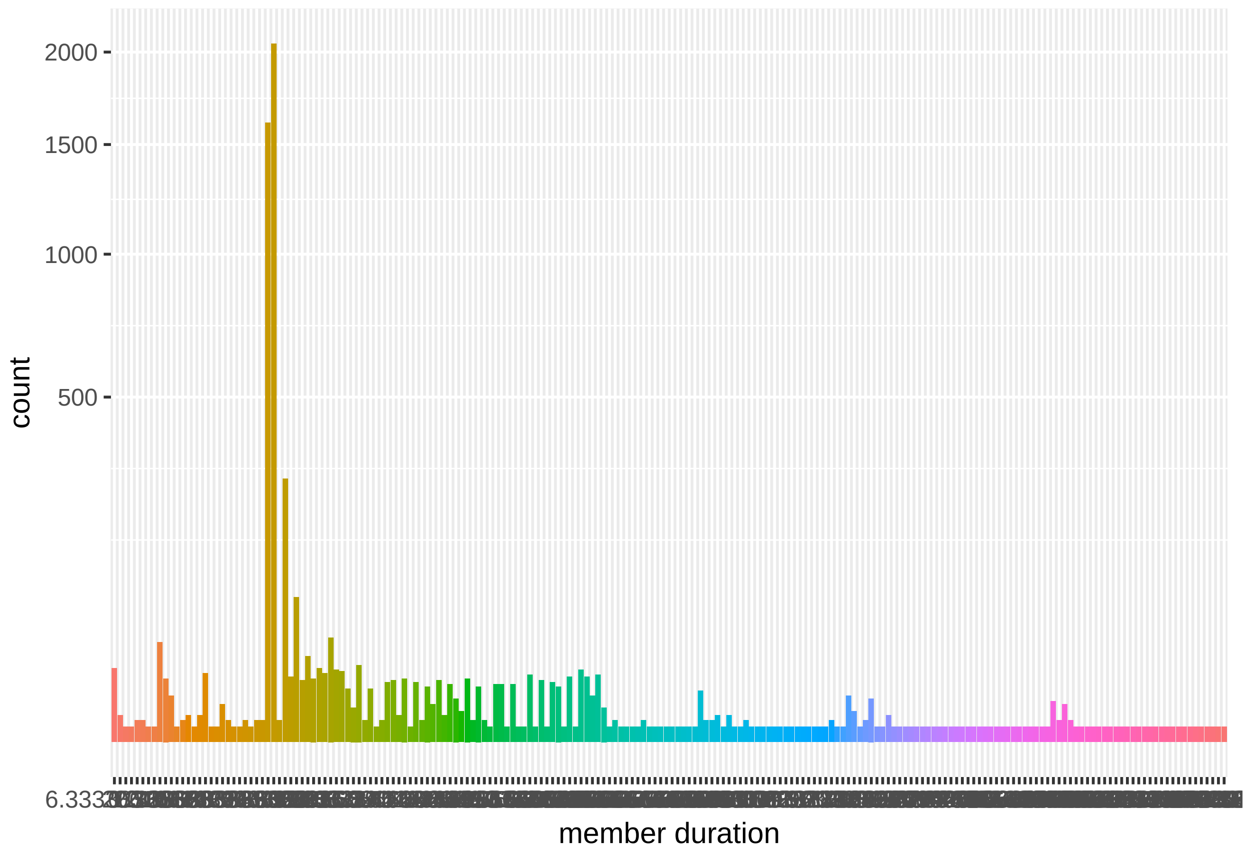




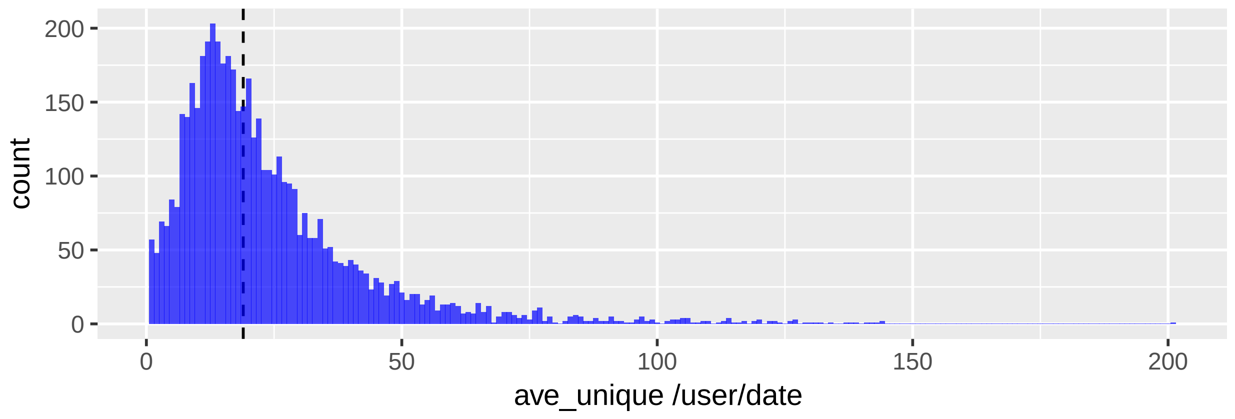
```r
grid.arrange(p5,p6,p7,p8)
```

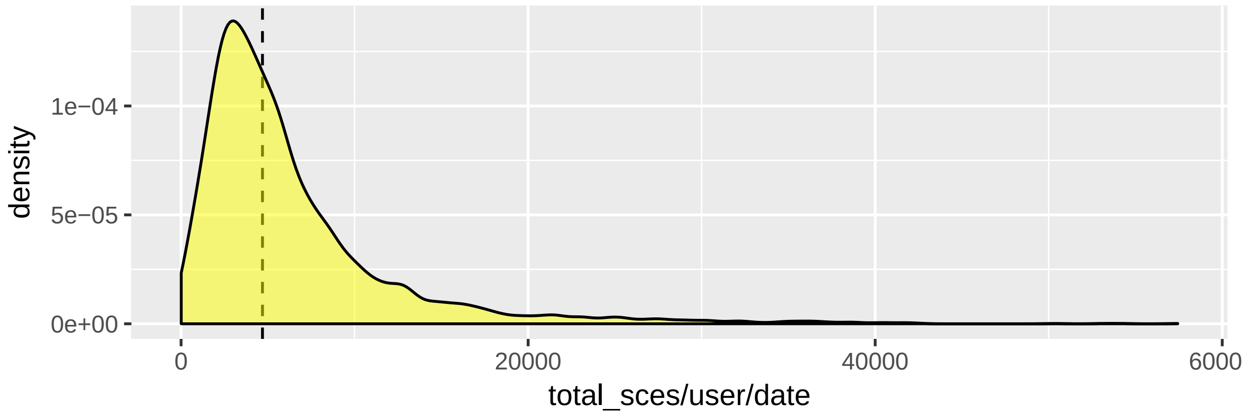

count
6000
4000
2000
0
0
1
is_cancel
count
4000
2000
0
0
1
is_auto_renew
Churn[%]
30
20
10
0
0
1
Auto renew
Churn[%]
60
40
20
0
0
1
Cancelled

```r
print(p9)
```

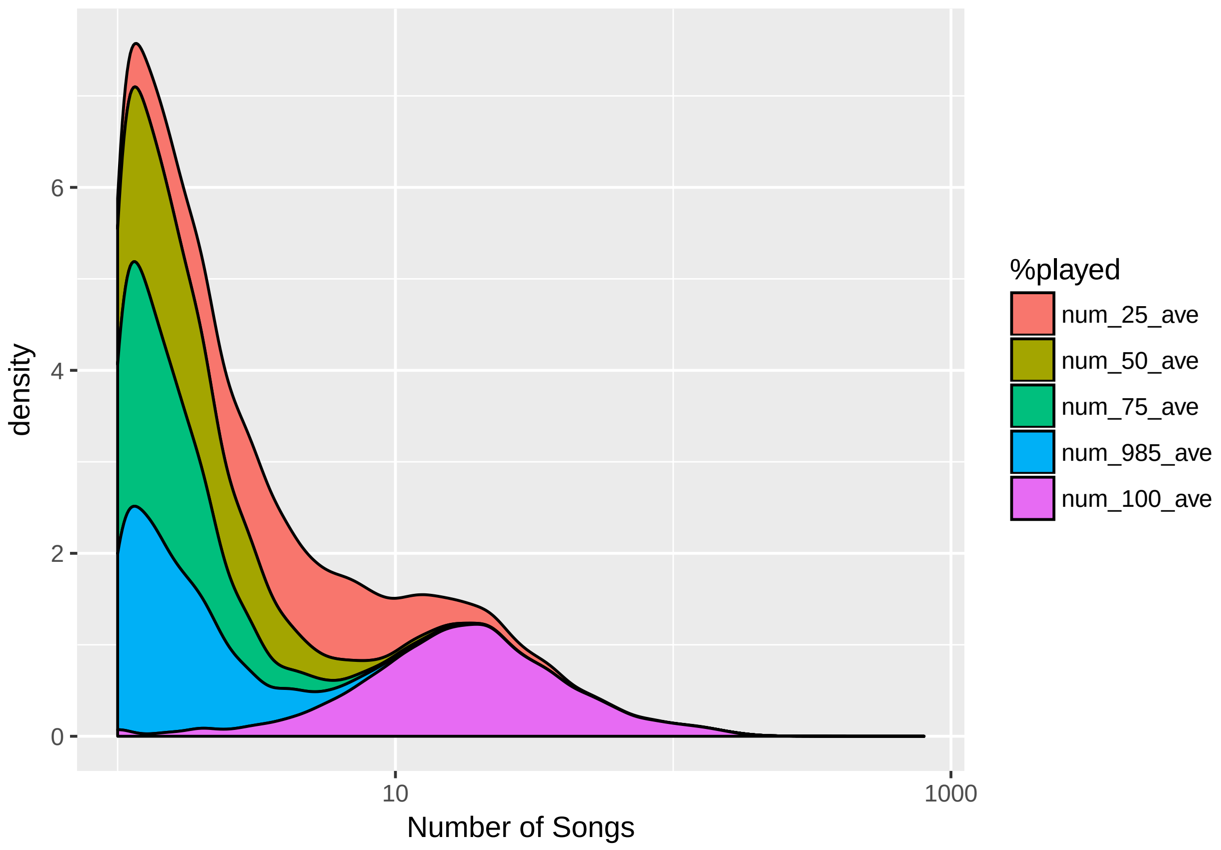


```
######## EDA For UserLog #######
load("../data/userlog.samp.RData")
load("../data/logfeature.RData")
p10<-ggplot(userlog.samp)+
  geom_vline(xintercept = median(userlog.samp$ave_unq), linetype=2)+
  geom_histogram(aes(x=ave_unq),binwidth  = 1, fill="blue",alpha=0.7)+
  labs(x="ave_unique /user/date ")
p11<-ggplot(userlog.samp)+
  geom_vline(xintercept = median(userlog.samp$total_secs_perdate), linetype=2)+
  geom_density(aes(total_secs_perdate), fill="yellow",alpha=0.5)+
  labs(x="total_sces/user/date")
p12<-userlog.samp %>%
  gather(num_25_ave,num_50_ave,num_75_ave,num_985_ave,num_100_ave, key="slen", value="cases") %>%
  mutate(slen=fct_relevel(factor(slen),"num_100_ave", after=Inf ))%>%
  ggplot(aes(cases, fill=slen))+
  geom_density(position = "stack",bw=0.05) +
  scale_x_log10(lim=c(1,800))+
  labs(x="Number of Songs", fill="%played")
grid.arrange(p10,p11)
```

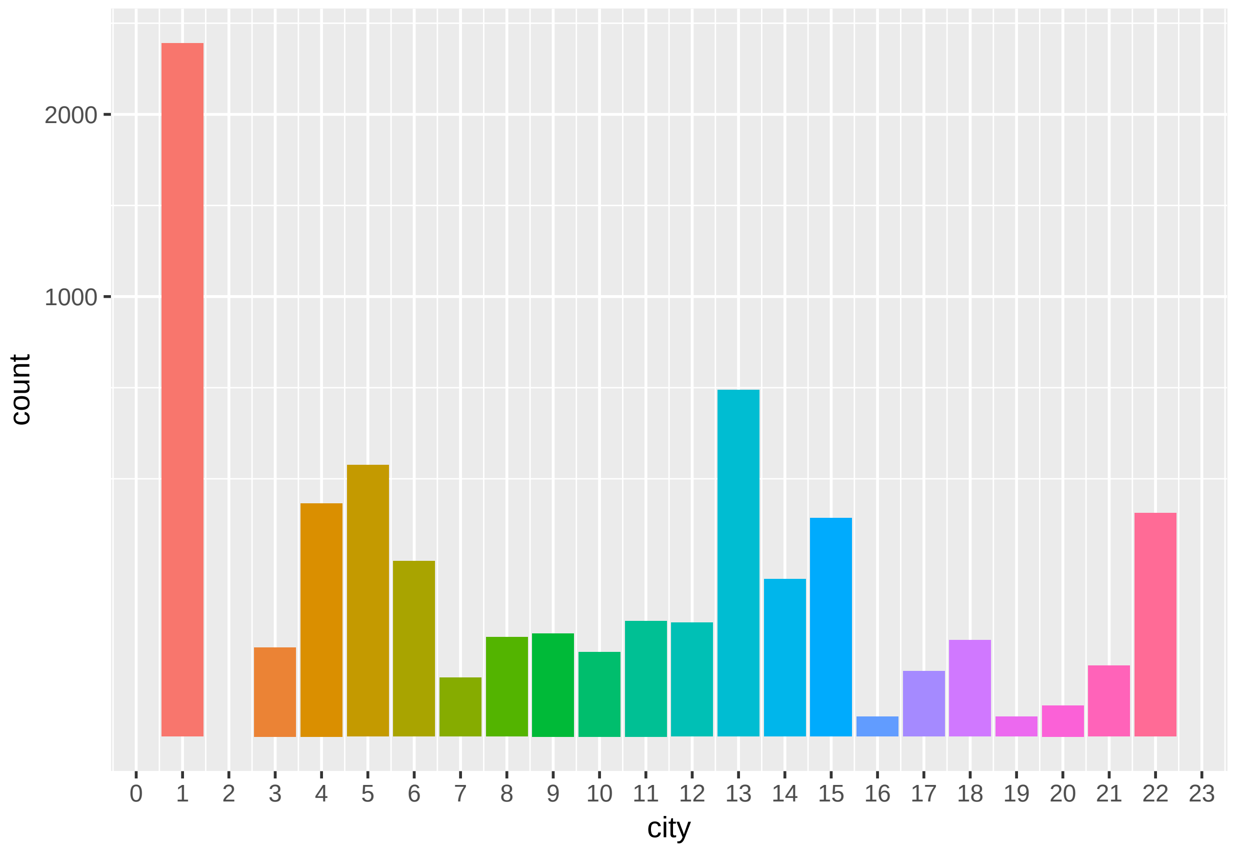

count
200
150
100
50
0
0
50
100
150
200
ave_unique /user/date


density
1e−04
5e−05
0e+00
0
20000
40000
6000
total_sces/user/date

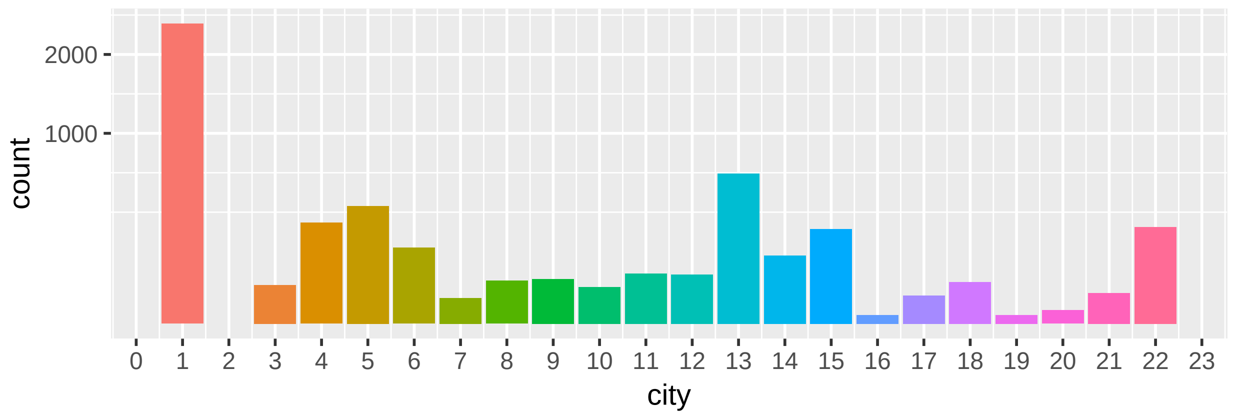
```r
print(p12)
```

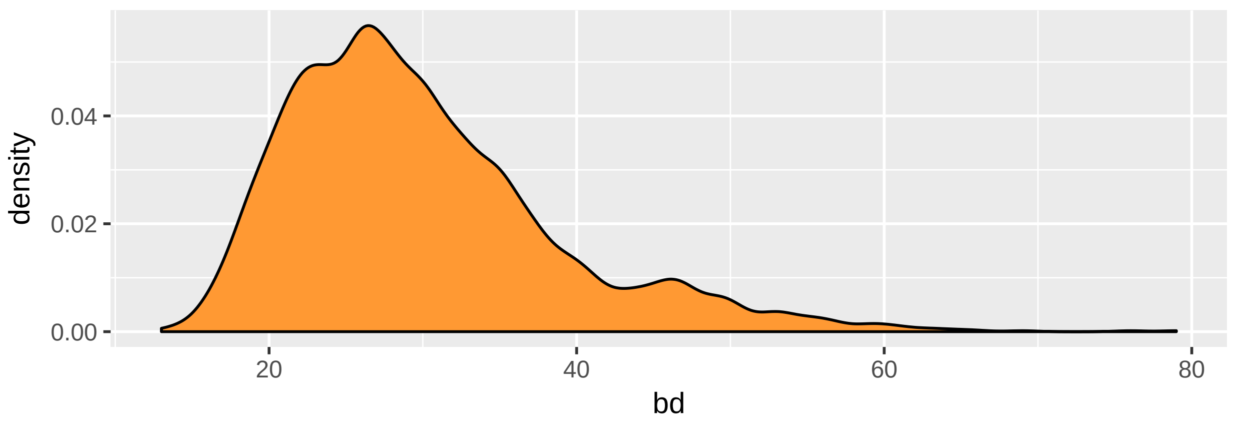


```r
######## EDA For Members #######
load("../data/data.all.RData")
p13<-data.all %>%
  ggplot(aes(city, fill = as.factor(city)) )+
  geom_bar() +
  theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0, 25, 1))+
  scale_y_sqrt()
p13
```



```
p14<-data.all %>%
  filter(bd > 0 & bd < 80) %>%
  ggplot(aes(bd)) +
  geom_density(fill="#FF9933",bw = 1)
grid.arrange(p13,p14)
```





**Step 2 Split the dataset and Set the evaluation metric**

```
# Loading the dataset that contains features we will use - data.all
load("../data/new3_data_all.RData")

# Split the dataset into Train data (80%) and Test data (20%)
smp_size <- floor(0.8 * nrow(data.all))
set.seed(123)
train_ind <- sample(seq_len(nrow(data.all)), size = smp_size)
train<- data.all[train_ind, ]
test <- data.all[-train_ind, ]

# The evaluation metric for this project is Log Loss
LogLossBinary <- function(actual, predicted, eps = 1e-15) {
  predicted <- pmin(pmax(predicted, eps), 1-eps)
  error<- - (sum(actual * log(predicted) + (1 - actual) * log(1 - predicted))) / length(actual)
  return(error)
}
```

**Step 3 train the models and get the error rate using testing data**

```
source("../lib/train-final.R")
source("../lib/test-final.R")
```

```r
result_table<-matrix(NA, 6,3)
result_table<-as.data.frame(result_table)
names(result_table)<-c("Model","Test_Error","CV_Error")
result_table$Model<-c("XGBoost","AdaBoost","Lasso","RandomForest","SVM","GBM")

# Train the six models using features in training data
fit.xgboost<-train.xgboost(train)
fit.adaboost<-train.adaboost(train)
fit.lasso<-train.lasso(train)
fit.rf<-train.rf(train)
fit.svm<-train.svm(train)
fit.gbm<-train.gbm(train)

# Test the six models  using features in testing data
pred.adaboost<-test.adaboost(fit.adaboost,test)
pred.xgboost<-test.xgboost(fit.xgboost,test)
pred.lasso<-test.lasso(fit.lasso,test)
pred.rf<-test.rf(fit.rf,test)
pred.svm<-test.svm(fit.svm,test)
pred.gbm<-test.gbm(fit.gbm,test)

# Calculate the test error rate for different models
actlabel<- as.numeric(as.character(test$is_churn))
err.xgboost<-LogLossBinary(actlabel, pred.xgboost)
err.adaboost<-LogLossBinary(actlabel, pred.adaboost)
err.lasso<-LogLossBinary(actlabel, pred.lasso)
err.rf<-LogLossBinary(actlabel, pred.rf)
err.svm<-LogLossBinary(actlabel, pred.svm)
err.gbm<-LogLossBinary(actlabel, pred.gbm)

result_table$Test_Error<-c(err.xgboost,err.adaboost,err.lasso,err.rf,err.svm,err.gbm)
```

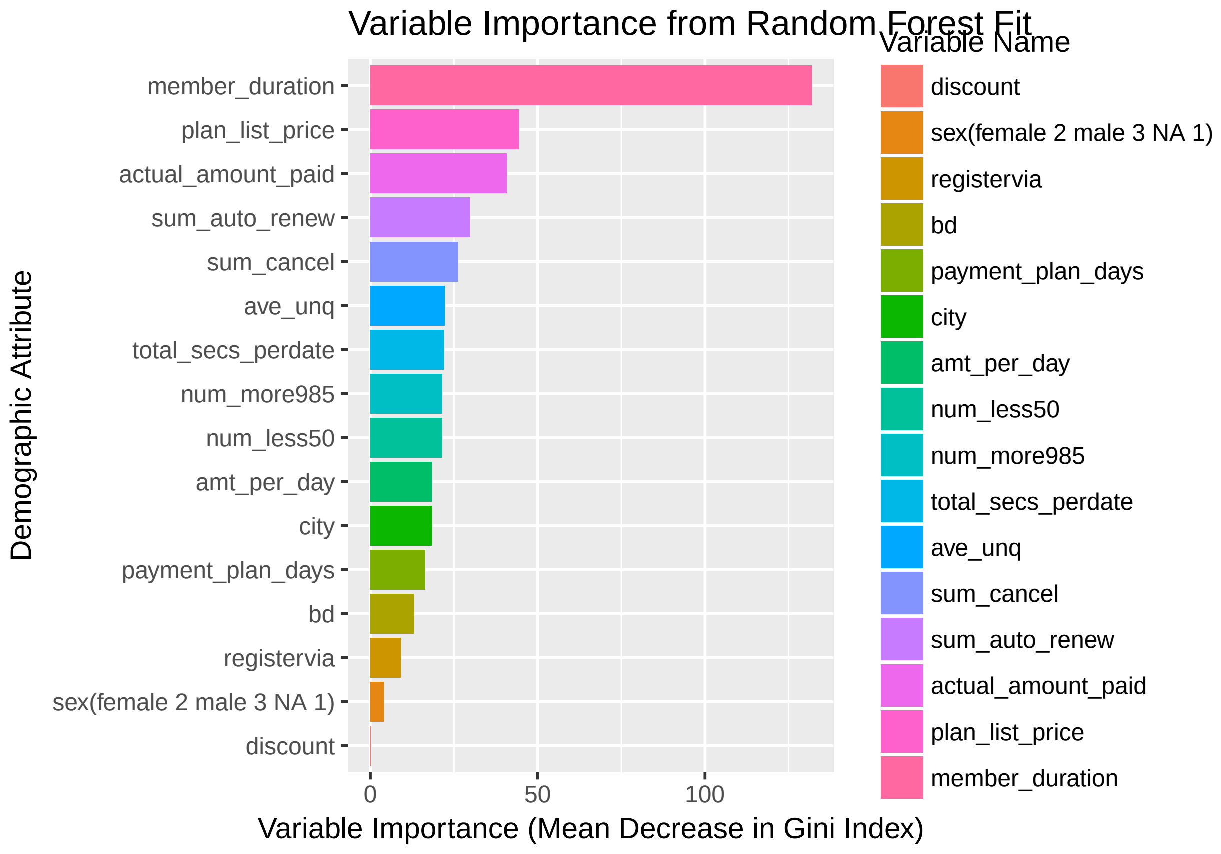
**Step 4: Cross-Validation to select the best model**

```r
### call the CV function to select the best model
source("../lib/CrossValidation.R")
cv.err<-cv.function(train,K=5)
result_table$CV_Error<-as.numeric(cv.err)
print(result_table)
```

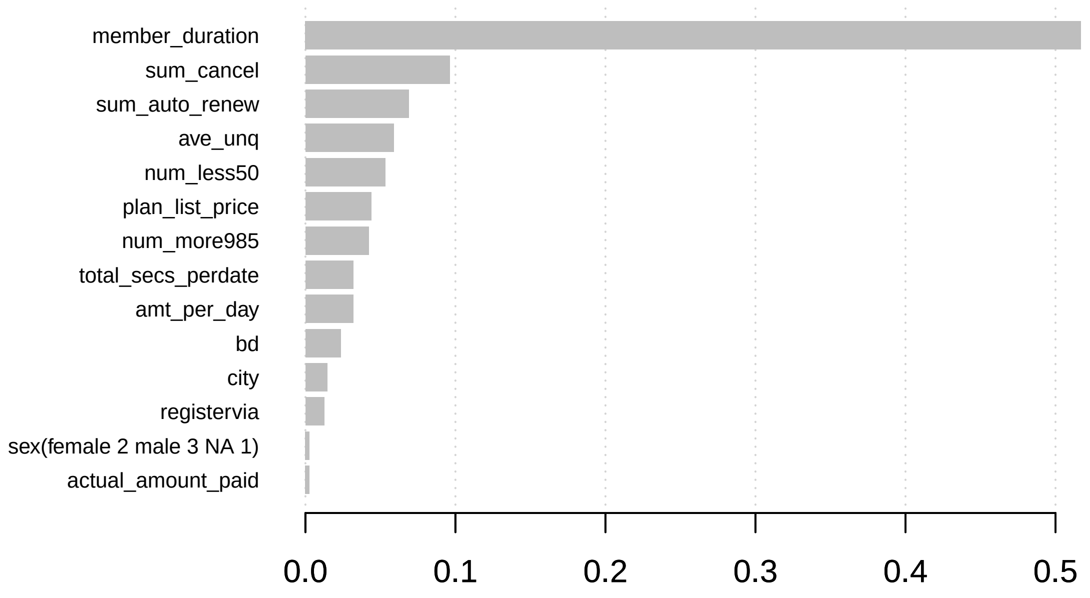
```
##          Model Test_Error   CV_Error
## 1      XGBoost 0.07489558 0.10835516
## 2     AdaBoost 0.11655770 0.13239858
## 3        Lasso 0.14458911 0.16744084
## 4 RandomForest 0.06012640 0.08337893
## 5          SVM 0.15439331 0.15695881
## 6          GBM 0.06630020 0.08804302
```

**Step 5: Feature Importance for Random Forest**

```
im = data.frame(rownames(fit.rf$importance),fit.rf$importance[,3],fit.rf$importance[,4])
colnames(im) = c("variables", "MeanDecreaseAccuracy","MeanDecreaseGINI")
im = im[with(im, order(MeanDecreaseGINI, variables)), ]
im$variables = as.factor(im$variables)
im$variables <- factor(im$variables, levels = im$variables[order(im$MeanDecreaseGINI)])
im$variables_MDA <- factor(im$variables, levels = im$variables[order(im$MeanDecreaseAccuracy)])

# Plot the importance of features for Random Forest Model
p <- ggplot(im, aes(x=variables, weight = MeanDecreaseGINI, fill = variables))
p <- p + geom_bar() + ggtitle("Variable Importance from Random Forest Fit")
p <- p + xlab("Demographic Attribute") + ylab("Variable Importance (Mean Decrease in Gini Index)")
p <- p + scale_fill_discrete(name="Variable Name")
p <- p + coord_flip()
print(p)
```



```
# Plot the importance of features for XGBoost Model
a<-train.xgboost(train)
varnames <- setdiff(colnames(train), c("msno", "is_churn"))
names <- dimnames(train[,c(-1,-2)])[[2]]
importance_matrix <- xgb.importance(names, model=a)
xgb.plot.importance(importance_matrix)
```



**Step 6: Use reduced features to test the model**

```r
data.all<-data.all[,-c(9,10,15)]
train<-train[,-c(9,10,15)]
test<-test[,-c(9,10,15)]
result_table_new<-matrix(NA, 6,3)
result_table_new<-as.data.frame(result_table_new)
names(result_table_new)<-c("Model","Test_Error","CV_Error")
result_table_new$Model<-c("XGBoost","AdaBoost","Lasso","RandomForest","SVM","GBM")

# Train the six models using features in training data
fit.xgboost<-train.xgboost(train)
fit.adaboost<-train.adaboost(train)
fit.lasso<-train.lasso(train)
fit.rf<-train.rf(train)
fit.svm<-train.svm(train)
fit.gbm<-train.gbm(train)

# Test the six models  using features in testing data
pred.adaboost<-test.adaboost(fit.adaboost,test)
pred.xgboost<-test.xgboost(fit.xgboost,test)
pred.lasso<-test.lasso(fit.lasso,test)
pred.rf<-test.rf(fit.rf,test)
pred.svm<-test.svm(fit.svm,test)
pred.gbm<-test.gbm(fit.gbm,test)

# Calculate the test error rate for different models
actlabel<- as.numeric(as.character(test$is_churn))
err.xgboost<-LogLossBinary(actlabel, pred.xgboost)
err.adaboost<-LogLossBinary(actlabel, pred.adaboost)
err.lasso<-LogLossBinary(actlabel, pred.lasso)
err.rf<-LogLossBinary(actlabel, pred.rf)
err.svm<-LogLossBinary(actlabel, pred.svm)
err.gbm<-LogLossBinary(actlabel, pred.gbm)

result_table_new$Test_Error<-c(err.xgboost,err.adaboost,err.lasso,err.rf,err.svm,err.gbm)

cv.err<-cv.function(train,K=5)
result_table_new$CV_Error<-as.numeric(cv.err)
print(result_table_new)
```

```
##          Model Test_Error   CV_Error
## 1      XGBoost 0.08475462 0.11676242
## 2     AdaBoost 0.11888597 0.12098767
## 3        Lasso 0.13511189 0.16831745
## 4 RandomForest 0.06082370 0.07765544
## 5          SVM 0.13208852 0.14474312
## 6          GBM 0.06877953 0.08793237
```

### Step 7: Further Investigation

```r
load("../data/data.all.raw.RData")
data.all = data.all[, -c(9,10,15)]
active_list = which(data.all$city != 1 & data.all$bd != 0)
act_data.all = data.all[active_list,]
pas_data.all = data.all[-active_list,]
pas_data.all = pas_data.all[,-c(7,8)]

################################################################################################
 #while using dataset for active member
 data.all = act_data.all
 data.all$num_less50 = scale(data.all$num_less50)
 data.all$num_more985 = scale(data.all$num_more985)
 data.all$ave_unq = scale(data.all$ave_unq)
 data.all$total_secs_perdate = scale(data.all$total_secs_perdate)
 data.all$bd = scale(data.all$bd)
 data.all$amt_per_day = scale(data.all$amt_per_day)
 data.all$plan_list_price = scale(data.all$plan_list_price)
 data.all$actual_amount_paid = scale(data.all$actual_amount_paid)
 data.all$member_duration = scale(data.all$member_duration)
```

```r
################################################################################################
 #while using dataset for passive member
 data.all = pas_data.all
 data.all$num_less50 = scale(data.all$num_less50)
 data.all$num_more985 = scale(data.all$num_more985)
 data.all$ave_unq = scale(data.all$ave_unq)
 data.all$total_secs_perdate = scale(data.all$total_secs_perdate)
 data.all$amt_per_day = scale(data.all$amt_per_day)
 data.all$plan_list_price = scale(data.all$plan_list_price)
 data.all$actual_amount_paid = scale(data.all$actual_amount_paid)
 data.all$member_duration = scale(data.all$member_duration)
```

```r
## choose either a active member dataset or passive member dataset
## then run the following code to get log-loss of random forest model for such data

smp_size = floor(0.8*nrow(data.all))
set.seed(123)
train_ind = sample(seq_len(nrow(data.all)), size = smp_size)
train <- data.all[train_ind,]
test <- data.all[-train_ind,]
ntree <- seq(100, 1000, by=100)

err_cv_rf <- c()
err_sd_rf <- c()

rf_cv <- function(dat_train, label_train, K=5, ntree=500){

  n <- length(label_train)
  n.fold <- floor(n/K)
  s <- sample(rep(1:K, c(rep(n.fold, K-1), n-(K-1)*n.fold)))
  cv.error <- rep(NA, K)

  for (i in 1:K){
    train.data <- dat_train[s != i,]
    train.label <- label_train[s != i]
    test.data <- dat_train[s == i,]
    test.label <- label_train[s == i]

    rf_fit <- randomForest(train.data, as.factor(train.label), ntree = ntree)
    rf_predict <- predict(rf_fit, test.data)
    cv.error[i] <- mean(rf_predict != test.label)

  }

  error <- mean(cv.error)
  sd <- sd(cv.error)

  return(c(error, sd))
}

for (j in 1:length(ntree)){
  cat("j=", j, "\n")
  result <- rf_cv(dat_train = train[,3:ncol(train)], label_train = train[,2], K = 5, ntree = ntree[j])
  err_cv_rf[j] <- result[1]
  err_sd_rf[j] <- result[2]
}
```

```
## j= 1
## j= 2
## j= 3
## j= 4
## j= 5
## j= 6
## j= 7
## j= 8
## j= 9
## j= 10
```

```
best_ntree = ntree[which.min(err_cv_rf)]
best.fit <- randomForest(train[,3:ncol(train)],
                         as.factor(train[,2]),
                         ntree = best_ntree,
                         importance = TRUE)

test_pred <- predict(best.fit, test[,3:ncol(train)], type = "prob")
eps = 1e-15
predicted <- pmin(pmax(test_pred, eps), 1-eps)


list_1 = which(test$is_churn == 1)
list_0 = which(test$is_churn == 0)
accuracy = - (sum(1*log(predicted[list_1, 2])) + sum(1*log(predicted[list_0,1]))) / nrow(test_pred)
```