Machine Learning Final Project 2: Kaggle Competition

Chien-Yu (Elaine) Su, Runda Xiong, Andrew Hu, Tianye Wang

Data set preparation:

1. Building features:

In addition to using the already rolled up data containing basic rfm variables, we created new variables to extract deeper insights within the customer spending model from the bookstore data. These variables include: monetary per category (naming “mf”), monetary per year (naming “m_2007~ m_2014”), unique orders per year(naming “ord_2007~ ord_2014”), average monetary per order (m_per_order), average time length for purchasing per item (tof_per_item) and average time length for a unique order(tof_per_order).
Note: the data is in the customer level so the unit for every variable is per customer.

2. Transformation:

After viewing the histogram of all the features, we found out that the monetary related variables are right skewed. Therefore, we decided to **log all the features that is related to monetary**.

Models:

We tried multiple models including simple linear regression with all variables, simple linear regression with significant variables only, Stepwise, Ridge, Lasso, Gam model for non-linear regression and Random Forest. After analyzing different models, we figured out that **Lasso and Stepwise perform the best MSE.**

Lasso:

Lasso regression helped us to choose some of the important variables, including numbers of items purchased, numbers of purchase for category No.27, monetary for category No.6, monetary for category No.14, monetary in 2012, and unique orders in 2013 and 2014.

fit_lasso_formula = as.formula("logtarg~ fitem+f27+mf6+mf14+m_2012+ord_2013+ord_2014")

Stepwise:

fit_stepwise_formula = as.formula("logtarg ~ fitem + ford + f1 + f3 + f5 + f8 + f17 +
f20 + f23 + f27 + f30 + f35 + f40 + f41 + f44 + mf1 + mf3 +

```
mf6 + mf12 + mf23 + mf27 + mf30 + mf36 + mf41 + m_2013 +
ord_2007 + ord_2008 + ord_2009 + ord_2010 + ord_2011 + ord_2012 +
ord_2013")
```

The Stepwise method picks some of the variables from our simple linear regression: Numbers of items purchased, numbers of total orders, numbers of purchase for category 1,3,5,8,17,20,23,27,30,35,40,41,44. Monetary spent for category 1,3,6,12,23,27,30,36,41. Monetary spent in 2013, and unique orders in 2007~2013.

Evaluation:

We use K- Fold Cross Validation for measuring our models within the training data set. The MSE for Lasso is 0.8094, and the MSE for stepwise is 0.8040.

Explanations for R codes:

```
# Loading libraries and setting working directory

library(tidyverse)
library(xgboost)
library(dplyr)
library(splines)
library(tree)
library(randomForest)
library(Metrics)
library(gam)

setwd("/Users/andrewhu/desktop/Kaggle")

################################ Read File and create basic features

# read in the transaction file
ord = read.csv("orders.csv")
dim(ord)
head(ord)
# the date of the offer was 11/25/2014, so t is time since action
ord$t = as.numeric(as.Date("2014/11/25") - as.Date(ord$orddate, "%d%b%Y"))/365.25
summary(ord$t)
hist(ord$t)
```

```
#read in the customer file with one row per customer
customer = read.csv("customer.csv")
names(customer)
head(customer)
table(customer$train)

# rollup order file to create RFM variables
rfm = ord %>%
  group_by(id) %>%
  summarise(tof=max(t), r = min(t), fitem=n(), ford=n_distinct(ordnum), m=sum(price*qty))


# this shows you how you can roll up order file counting purchases by category
cats = sort(unique(ord$category))  # list of all unique categories
cats
rfm2 = ord %>%
  group_by(id, category) %>%
  summarise(f=n()) %>%
  spread(category,f, fill=0)  %>%
  setNames(c("id", paste("f", cats, sep="")))
head(rfm2)
summary(rfm2)
```

 # roll up : Money spent by category

```
cats = sort(unique(ord$category))  # list of all unique categories
cats
rfm3 = ord %>%
  group_by(id, category) %>%
  summarise(mf=sum(price*qty)) %>%
  spread(category,mf, fill=0)  %>%
  setNames(c("id", paste("mf", cats, sep=""))) #naming “mf”
head(rfm3)
summary(rfm3)

# this joins the customer, RFM and RFM by category tables
all = left_join(customer, rfm, by="id") %>%
  left_join(rfm2, by="id")
summary(all)
names(all)
```

#Join using original rfm features and monetary by category

```
all_alt = left_join(customer,rfm,by="id") %>%
```

```
  left_join(rfm2, by="id") %>%
  left_join(rfm3, by="id")
summary(all_alt)
names(all_alt)

ord$mydate = as.Date(ord$orddate, "%d%b%Y")
ord$year = format(ord$mydate,"%Y")
```

#Creating features of sales by years

```
monetart_df = ord %>% group_by(id, year) %>% dplyr::summarise(sls_amt=sum(price*qty))
monetart_df2 = monetart_df %>% tidyr::spread(year,sls_amt)
monetart_df2[is.na(monetart_df2)] = 0

#Merging into the dataset
all_alt2 = merge(x = all_alt, y = monetart_df2, by = "id", all = TRUE)
```

#Creating features of unique orders count by years

```
yearord_df = ord %>% group_by(id, year) %>% dplyr::summarise(year_ord=n_distinct(ordnum))
yearord_df2 = yearord_df %>% tidyr::spread(year,year_ord)
yearord_df2[is.na(yearord_df2)] = 0
```

#Merging all of the features

```
all_alt3 = merge(x = all_alt2, y = yearord_df2, by = "id", all = TRUE)


#Rename some of the variables

colnames(all_alt3)[69] <- "m_2007"
colnames(all_alt3)[70] <- "m_2008"
colnames(all_alt3)[71] <- "m_2009"
colnames(all_alt3)[72] <- "m_2010"
colnames(all_alt3)[73] <- "m_2011"
colnames(all_alt3)[74] <- "m_2012"
colnames(all_alt3)[75] <- "m_2013"
colnames(all_alt3)[76] <- "m_2014"
colnames(all_alt3)[77] <- "ord_2007"
colnames(all_alt3)[78] <- "ord_2008"
colnames(all_alt3)[79] <- "ord_2009"
colnames(all_alt3)[80] <- "ord_2010"
colnames(all_alt3)[81] <- "ord_2011"
```

```
colnames(all_alt3)[82] <- "ord_2012"
colnames(all_alt3)[83] <- "ord_2013"
colnames(all_alt3)[84] <- "ord_2014"

#Rename the data set

all3= all_alt3
```

#Create features of: money per order, time length per item, time length per order

```
all3$m_per_order= (all3$m / all3$ford)
all3$tof_per_item = (all3$tof/ all3$fitem)
all3$tof_per_order =(all3$tof/all3$ford)
```

######Transformation######:

#Log on monetary related variables

```
all3$m= log(all3$m+1)

for (i in 39:76) all3[[i]] = log(all3[[i]]+1)
```

####### Dividing into train and test set

```
#Create train and test set
train = (all$train==1)
train_df3 = all3 %>% filter(train==1)
test_df3 = all3 %>% filter(train==0)
```

#Self-written functions:

1. **K Fold CV:**

```
my_cv <- function(fm, df, cv_fold_number){
  set.seed(12345)

  df$cv = as.integer(runif(nrow(df))*cv_fold_number) # K=5 random split

  yhat = rep(NA, nrow(df)) # set up vector for held-out predictions
```

```
  # use a for loop to predict each fold after estimation using the other folds
  for(i in 0:cv_fold_number-1){
    fit = lm(fm, df, subset=(cv!=i))
    yhat[df$cv==i] = predict(fit, df[df$cv==i,])
  }

  cv_mse = mean((df$logtarg-yhat)^2) # CV MSE

  return(cv_mse)
}
```

2. **Lasso function:**

```
my_cv_glmnet <- function(y, x, alpha){
  # set seed for cross validation
  set.seed(1)

  # using cv.glmnet to build lasso. The following line calculate 3 fold cv for each lambda, so
there will be 1000*3 model fitting.
  fit.cv=cv.glmnet(x,y,alpha=alpha,nfold=3,lambda=seq(0,10,0.01))

  # get the lambda with the smallest Mean-Squared Error
  fitted_min_lambda=fit.cv$lambda.min

  # get the index of the smallest lambda, and use it to find our ideal coefficient
  small.lambda.index <- which(fit.cv$lambda == fit.cv$lambda.min)
  small.lambda.betas <- coef(fit.cv$glmnet.fit)[,small.lambda.index]

  return(list(lambda=fitted_min_lambda,
              small.lambda.betas=small.lambda.betas))
}
```

3. Random Forest function:

```
param_grid_rf <- function(df, response, ntree, Nrep, mtry_list, nodesize_list){
  # Get the best parameters from every combination of the input mtry and nodesize according to
OOB R^2 obtained from my_rf_oob function
  # Args
  # ----------
  #   df (data.frame)
```

```
  #   response (string): the response variable name
  #   ntree (int): a fix ntree value. Pick a number that is large enough according to the CV error
obtained from random Forest function
  #   Nrep (num): number of replicates of CV. Used in order to check the effect of the
randomness when doing CV
  #   mtry_list (list): a list mtry values
  #   nodesize_list (list): a list nodesize values
  # Returns
  # ----------
  # list:
  #    a list containing the oob R2 in each combination, the best number of mtry and nodesize

  r2list= c()
  i=1
  for (m in mtry_list){
    for (n in nodesize_list){
      r2list[i] = my_rf_oob(df=df,
                      response=response,
                      mtry=m,
                      ntree=ntree,
                      nodesize=n,
                      Nrep=Nrep)$oobr2
      i=i+1
    }
  }

  best_param = get_param_value(r2list, mtry_list, nodesize_list, max=TRUE)

  return(list(r2list=r2list,
              mtry=best_param$outer,
              nodesize=best_param$inner))
}


my_rf_oob <- function(df, response, mtry, ntree, nodesize=5, Nrep){
  # Calculating out of bag error score for random forest using the input fix parameters
  # The goal is to decrease the random effect of using only one single CV fold
  # The default of nodesize is 5 according to the function documentation
  #
  # Args
  # ----------
  #   df (data.frame)
  #   response (str): The response variable
```

```
  #   mtry (int): Number of variables randomly sampled as candidates at each split
  #   ntree (int): Number of trees to grow
  #   nodesize ():
  #     Minimum size of terminal nodes. The number of observation in the leaf node.
  #     Setting this number larger causes smaller trees to be grown (and thus take less time).
  #     Note that the default values are different for classification (1) and regression (5).
  #   Nrep (int):
  #     number of replicates of CV.
  #     Used in order to check the effect of the randomness when doing CV
  #
  # Returns
  # ----------
  # list
  #    return the mean OOB MSE and R^squared error of randomfoest using fix parameters
(mtry, ntree, nodesize)

  y = df[,c(response)]
  fm = as.formula(paste(response, '~.', sep=''))
  MSE =c()

  # run gbm for Nrep number of times to get the MSE for each time. We do this in order to get a
better estimate of MSE. Using a single CV to estimate the test MSE may suffer from random
bias
  for (i in seq(1, Nrep)){
    rf_nrep = randomForest(fm, data=df, mtry=mtry, ntree = ntree, nodesize=nodesize,
importance = TRUE)

    yhat = rf_nrep$predicted
    var_e = var(yhat-y) # var_e = MSE
    MSE[i] = var_e #oob MSE
  }
  oobr2 = 1- mean(MSE)/var(y)

  return(list(MSE = MSE,
             oobr2 = oobr2))

}
```

############# MODELS ################

```
#SLR
##### using all features, all3
fit_slr_formula = as.formula("logtarg~.")
fit_slr = lm(logtarg ~ ., all3[train, -c(1,2)])  # -c(1,2) drops columns 1 and 2
summary(fit_slr)

##### using selected feature, log on m related

fit_slr_sel_formula= as.formula("logtarg~tof+ ford+ f3+mf3+mf5+mf6+mf35+ ord_2007+
ord_2008+ord_2009+ord_2010+ord_2011+ord_2012+ord_2013")

fit_slr_sel_gam_formula= as.formula("logtarg~s(fitem)+ s(tof)+ ford+
f3+s(mf3)+s(mf5)+s(mf6)+s(mf35)+ s(ord_2007)+
s(ord_2008)+s(ord_2009)+s(ord_2010)+s(ord_2011)+s(ord_2012)+s(ord_2013)")

fit_slr_sel=lm(fit_slr_sel_formula, all3[train, -c(1,2)])

fit_slr_sel_gam = gam(fit_slr_sel_gam_formula, optimizer = "perf", data=all3[train,-c(1,2)])
summary(fit_slr_sel_gam)
```

#Lasso

```
y = all3 %>% filter(train==1)
y = y$logtarg
x=model.matrix(logtarg~.,all3[train, -c(1,2)])
lasso = my_cv_glmnet(y,x,1) #using the Lasso function
Lasso # return the lambda

#Formula: Lasso using linear model
fit_lasso_formula = as.formula("logtarg~ fitem+f27+mf6+mf14+m_2012+ord_2013+ord_2014")

#Formula: Lasso using gam non-linear model
fit_lasso_gam_formula = as.formula("logtarg~
s(fitem)+s(f27)+s(mf6)+s(mf14)+s(m_2012)+s(ord_2013)+s(ord_2014)")

#Lasso linear model
fit_lasso = lm(fit_lasso_formula, all3[train, -c(1,2)])  # -c(1,2) drops columns 1 and 2
#Lasso gam model
fit_lasso_gam = gam(fit_lasso_gam_formula, optimizer = "perf", data= all3[train, -c(1,2)])
```

#Predictions using Lasso

yhat_lasso <-predict(fit_lasso, all3[!train,-c(1,2)])

#Stepwise
step(fit_slr,direct="both") #Using Stepwise to choose variables from simple linear regression

#Formula for stepwise
fit_stepwise_formula = as.formula("logtarg ~ fitem + ford + f1 + f3 + f5 + f8 + f17 +
  f20 + f23 + f27 + f30 + f35 + f40 + f41 + f44 + mf1 + mf3 +
  mf6 + mf12 + mf23 + mf27 + mf30 + mf36 + mf41 + m_2013 +
  ord_2007 + ord_2008 + ord_2009 + ord_2010 + ord_2011 + ord_2012 +
  ord_2013")

#Stepwise linear model
fit_stepwise <-lm(fit_stepwise_formula, data = all3[train, -c(1, 2)])

#Formula for stepwise non-linear
fit_stepwise_gam_formula = as.formula("logtarg ~ s(fitem) + s(ford) + f1 + f3 + f8 + f17 +
  f20 + f27 + f35 + f40 + f44 + s(mf3) +
  s(mf6) + s(mf12) + s(mf36) + s(m_2013) +
  s(ord_2007) + s(ord_2008) + s(ord_2009) + s(ord_2010) + s(ord_2011) + s(ord_2012) +
  s(ord_2013)")

#Non linear stepwise model
fit_gam_step = gam(fit_stepwise_gam_formula, data= train_df3[,-c(1,2)])

#Prediction using stepwise: Linear and non-linear

yhat_step=predict(fit_stepwise, test_df3[,-c(1,2)])
yhat_gam= predict(fit_gam_step, test_df3[,-c(1,2)])

#Change the negative prediction value to zero
yhat_step[yhat_step<0]=0

```
### RandomForest

#Getting the best parameters
param_tune_rf = param_grid_rf(df=all3[train, -c(1,2)],
                        response="logtarg",
                        ntree=500,
                        Nrep=1,
                        mtry_list=c(4,5),
                        nodesize_list=c(28,29,30,31,32))

# get the oob r2 estimate using the best parameters
oobr2_list_best = my_rf_oob(df=all3[train, -c(1,2)],
                      response="logtarg",
                      mtry=param_tune_rf$mtry,
                      ntree=500,
                      nodesize=param_tune_rf$nodesize,
                      Nrep=1)


#Random forest regression

rf_base = randomForest(logtarg~., data=all3[train, -c(1,2)], mtry=param_tune_rf$mtry, ntree =
50, nodesize=param_tune_rf$nodesize, importance = TRUE)

#Prediction using RF
rf_predict = predict(rf_base, all3[!train,-c(1,2)])
```

####### Evaluation using CV####

```
my_cv(fit_slr_formula,train_df3,5)

my_cv(fit_lasso_formula,train_df3,5)

my_cv(fit_stepwise_formula,train_df3,5)
```