Hungary PST Document in R

Yan Xue

Aug23, 2016

## Summary of modeling process in R

The main purpose of this R program is to predict market/product sales for non-panel pharmacy stores that are in the pharmacy universe, using the panel pharmacy stores as training data. The R program will automatically select the best among 4 types of models, and then apply it in prediction of sales for those non-panel pharmacy stores. The R program will also caculate various model performance measurements based on the actual sales and the predicted sales for panel stores.It will generate csv outputs contains predictions and model performance measure and computation time of each model. And it will finally consolidate the results of all best models.

First of all, let's start from the 4 input data files that are generated by last step in SAS (i.e. data pre-processing step). They will be read into R and will be processed for next steps of modeling, model performance assessment and optimal model selection.

1. all\_MERCK\_HUN\_PST.csv: main data with sales/log sales (sell-out sales), all predictors and log of continous predictors;
2. zip sales.csv: sell-in sales at zip level;
3. pharmacy\_universe.csv: pharmacy universe;
4. Full\_geo\_sheet.csv: geographic information.

The processes can be splitted 5 steps as below. More details can be found in description of each step before each chunk of code.

**Step 0: Initial settings**

**Step 1: Create folders and copy input data for the first run**

**Step 2: Read in all input data and process them**

**Step 3: Run all models**

**Step 4: Combine results of all models**

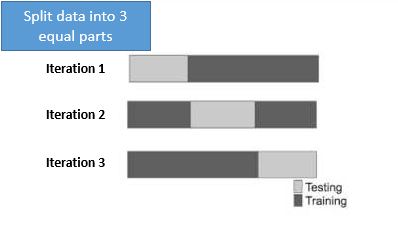
#### Some high level summary of Step 3.

There are 4 model functions to be called in Step 3. The four model types are stepwise regression, lasso regression, random forest and gradient boosting model. The stepwise regression and lasso regression will be run twice, one is for boolean predictors + orinigal continous predictors and another is for boolean predictors + log of original continous predictors. In all, there will be 6 models to be run for each market/each product. The process flow in each model function is similar. And here is some description how the model input data are processed by them.

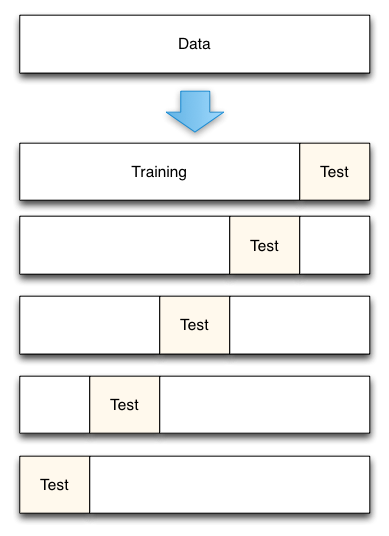
1. The panel stores in the main data will be first randomly splitted into nFolds (see definition of nFolds in Step 0, in this case it is 3) parts with equal size; then the model will be trained on (nFolds-1) parts (we call it training data) of panel stores. The optimal model will be selected by cvFolds (see definition of cvFolds in Step 0, in this case it is 5) folds cross-validation. The optimal model will then be applied to the 1 left-out part (we call it test data) of panel stores and will predict sale for them. The performance measure for selecting optimal model is MSE/RMSE. This process will be repeated nFolds times and each time a different set of (nFolds-1) parts will be used. So there will be nFolds best models selected and built. The nFolds models will be applied to non-panel stores in main data and predict the sales for them. The predicted sales of these non-panel stores will be average of predictions of nFolds best models.

Let's take this case as an example where nFolds=3 and cvFolds = 5.

First, split Data into 3 equal parts:



Secondly, do 5 folds cross-validation on every 2 parts:



1. Stores that are in the universe but not included in the main data will also get predicted sales after the above are finished. The predicted sales is the average predicted sales of stores from main data that are in the same mb\_code or in the same sb\_code (if the former is unavailable).
2. All predicted sales will be adjusted according to 3 versions of methodologies that have been pre-defined.
3. The model performance measurement is calculated based on actual sales and predcited sales for the (nFolds-1) parts and 1 left-out part. So there will be nFolds set of model performance measurement (R-Square, MAPE, MAE and etc...). The overall model performance measurement for training is simple average of model performance measurement for the nFolds different training data; but the overall model performance measurement for test is based on actual sales and predicted sales for nFolds test data. (Note: Before calculating MAPE and MAE, the actual sales that is equal to 0 will be set to average sales.)

*Note: stepwise regression is different from other 3 types of models in Bulletin 1. This is because no hyper-parameter needs to be selected for this type of model. But the stepwise will still be run nFolds times and each time on a different set ofs (nFolds-1) parts*

#### Some high level summary of Step 4.

After the modeling process, the R code will automatically select the best model among the **6** best models of **4 types** using a criterion (see the detailed description of this criterion in the comments before Chunk 10). It will export the key model performance measures for best model.

### 

## Break down the process into steps and explain each step in details

### Step 0: Initial settings

This step is to set work directory for the project using the client name and the project name, and to specify market names and product names and etc (see parameter definition below.)

*Note: The code will automatically create a new folder if the work folder does not exist.*

#### Parameters to change in Step 0:

1. client: a character string which is typcially a combination of client name and project name.
2. prj\_root\_path: root path of this project where you save input data and output files.
3. market: a vector of all market names.
4. product: a vector of all product names.
5. nFolds: number of folds that you want to split the input data into.
6. cvFolds: number of folds for cross-validation within each (nFolds - 1).
7. cores: number of cpu cores to be used for multiple thread processing.
8. parallel: boolean value (TRUE/FALSE) for multiple thread processing.

rm(list=ls())  
options(warn = -1)  
  
client <- "MERCK\_HUN\_PST"  
prj\_root\_path <- "D:/Project Files2/Europe/zhiyu/HU PST/test run/"  
market = c("Market1", "Market2", "Market3")  
product = c("Product1", "Product2", "Product3")  
  
nFolds <- 3  
cvFolds <- 5  
cores <- 3  
parallel <- TRUE  
  
  
targetFolder <- paste0(prj\_root\_path, client)  
if (file.exists(targetFolder)==FALSE) {  
 dir.create(targetFolder, showWarnings = TRUE,   
 recursive = TRUE, mode = "0777")  
}  
setwd(targetFolder)  
  
source("../Codes/functions.R")   
source("../Codes/001 Stepwise model.R")  
source("../Codes/002 Lasso Model.R")  
source("../Codes/003 RF Model.R")  
source("../Codes/004 Boosting Model.R")  
  
print(getwd())

### Step 1: Create folders and copy input data for the first run

This step is to create two subfolders under the work directory if it is the first time you run the program (i.e. the subfoldes are not yet created before you run it). One is called "01 data", and the other is "02 results".

And then copy 4 input data files (csv) from the original folder to the "01 data" folder under the work directory.

#### Parameters to change in Step 1:

1. dataFile: it is the name of main input data containing panel stores sales and predictors
2. from\_path: the relative path or absolute path from where you copy all input data to the destination folder (i.e. "01 data" folder as mentioned above)

*Note: if you need to recopy input data to the destination folder, you should need to delete the entire destination folder and then re-run this code chunck.*

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
dataFile <- paste("all\_", client, ".csv", sep="")  
from\_path <- "../Raw data/"  
  
files\_to\_be\_copied <- c(dataFile,"zip sales.csv","pharmacy\_universe.csv", "Full\_geo\_sheet.csv")   
  
#Step 1: Create the final output table and copy the data  
 if (file.exists("./01 data")==FALSE){  
   
 dir.create("./01 data/", showWarnings = TRUE,   
 recursive = TRUE, mode = "0777")  
  
 for(f\_name in files\_to\_be\_copied){  
 file.copy(from= paste(from\_path, f\_name, sep=""),   
 to = "./01 data/",   
 overwrite = TRUE) }   
   
 }  
  
  
 if(file.exists("./02 results/")==FALSE){  
 dir.create("./02 results/", showWarnings = TRUE,   
 recursive = TRUE, mode = "0777")  
 }

### Step 2: Read in all input data and process them

This step is to read in all input data and then process them for next step.

#### Substep 2.1: Read in panel store sales data and proecss it

Read in main data with panel store sales and predictors;

drop non-predictors such as 'poi\_id', 'panel\_pharmacy' and 'zip'.

select boolean predictors and original continous/ordinal predictors, and put them in the vector "vars";

select original panel stores sales, and put their names in the vector "var\_response";

select boolean predictors and log of original continous/ordinal predictors, and put their names in the vector "log\_vars";

select log of original panel store sales, and put their names in the vector "log\_var\_response".

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 rawData <- read.csv(paste("./01 data/", dataFile, sep=""))  
  
 names(rawData) <- tolower(names(rawData))  
 modelData <- rawData[, setdiff(names(rawData), paste("s\_",1:200, sep=""))]  
   
 # Clarify the variables needed to be dropped  
 dropVars <- c("poi\_id","panel\_pharmacy","zip")  
   
 # variables for normal model   
 vars <- setdiff(names(modelData),   
 c(dropVars,  
 names(modelData)[grep("log\_|lg\_", names(modelData))]))  
   
 var\_response <- vars[grep("pp\_market|pp\_product", vars)]  
 vars <- setdiff(vars, var\_response)  
 cat(">>> There are", length(vars), "Variables for model <<<\n")  
   
 # variables for Log-Log model   
 log\_vars <- c(names(modelData)[grep("log\_|lg\_|dummy",   
 names(modelData))],  
 "chain")  
 log\_var\_response <- log\_vars[grep("pp\_market|pp\_product", log\_vars)]  
 #change to the same order with var\_response  
 id <- numeric()  
 for (i in 1:length(var\_response)){  
 ii <- grep(var\_response[i], log\_var\_response)  
 id <- cbind(id, ii)  
 }  
 log\_var\_response <- log\_var\_response[id]  
   
 log\_vars <- setdiff(log\_vars, log\_var\_response)  
 cat(">>> There are", length(log\_vars), "Variables for log model <<<\n")

#### Substep 2.2: Read in the other data and process them

Read in zip sales data, pharmacy universe and the full geographic data; make necessary changes about these data

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 # Read the sell-in sales data  
 zipData <- read.csv("./01 data/zip sales.csv")  
 names(zipData) <- c("ZIP", var\_response, "No..Of.Pharmacies")  
 cat(nrow(zipData), "records of sell-in data.\n")  
   
 #Pharamacies not in model data  
 PhData <- read.csv("./01 data/pharmacy\_universe.csv")  
 no\_POI <- PhData[PhData$POI.ID %in% rawData$poi\_id == FALSE,c("POI.ID","ZIP")]  
 names(no\_POI) <- c("poi\_id","zip")  
 dim(no\_POI)  
   
 # Full geography table   
 PhData <- read.csv("./01 data/Full\_geo\_sheet.csv")  
 PhData <- PhData[,c("zip", "mb\_code", "sb\_code")]  
 dim(PhData)

### Step 3: Run all models

Run all models, including:

1. Stepwise regression model (original predictors/log of original predictors);
2. Lasso regression model (original predictors/log of original predictors);
3. Random forest model;
4. Gradient boosting model.

Basically, the code will execute a set of models of each type if there are more than one market or more than one prodcut. The size of the set is determined by the sum of number of markets and number of products.

#### Substep 3.1: Stepwise regression model

Run stepwise regressin model, the first set of models use original continuous predictors, the second set of models use log of orignial continous predictors.

The outcomes of stepwise models will be saved in a sub-folder called "Stepwise" under "02 results" folder.

The code will automatically create a new folder called "Stepwise" if it is the first time you run the code.

The code will run loops thru the set of dependent variables (markts and products) in parallel on multiple cores of the CPU.

Outcomes of stepwise models are:

1. Basic metrics for each market/product; it includes RSquare, MSE, MAPE and MAE, all\_Rank, RankIn10\_gps; The code will also stack all models performance metrics and save them in "allMetrics.csv" file under the "Stepwise" folder.
2. Coefficients of stepwise regression for each market/product;
3. Hit matrix for each market/product;
4. Lift, effectiveness and hit rate for each market/product;
5. Predicted sales and adjusted predicted sales (3 versions) for each market/product.

In the meantime, the run time will also be recorded and printed out.

### this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd()) ###end  
install\_load("doParallel")  
resultDir <- "./02 results/Stepwise"  
if (file.exists(resultDir) == FALSE) {  
 dir.create(resultDir, showWarnings = TRUE, recursive = TRUE, mode = "0777")  
}  
  
  
s <- Sys.time()  
bParallel <- parallel #Whether or not to do parallel Run  
if (bParallel) {  
 cl <- makeCluster(cores)  
 registerDoParallel(cl, cores = cores)  
}  
all\_step <- foreach(mktid = 1:length(var\_response), .packages = c("caret", "plyr",   
 "dplyr", "Metrics")) %dopar% {  
 stepwiseModel(type = "original", standardize = "TRUE", responseid = mktid,   
 numberOfpart = nFolds)  
 stepwiseModel(type = "log", standardize = "TRUE", responseid = mktid, numberOfpart = nFolds)  
}  
if (bParallel) {  
 stopCluster(cl)  
}  
  
  
allMetrics <- combineResults(resultDir)  
write.csv(allMetrics, paste(resultDir, "allMetrics.csv", sep = "/"), na = "",   
 row.names = FALSE)  
e <- Sys.time()  
cat("Stepwise Models finished! using time:", e - s, attr(e - s, "units"), "\n")  
  
stepwise\_time <- data.frame(Model = "Stepwise", nfolds = nFolds, cvfolds = 0,   
 usingTime = e - s)  
stepwise\_time$usingTime <- paste(round(stepwise\_time$usingTime, 4), attr(stepwise\_time$usingTime,   
 "units"), sep = " ")

#### Substep 3.2: Lasso regression model

Run lasso regression models, the first set of models use original continuous predictors, the second set of models use log of orignial continous predictors.

The outcomes of lasso regression models will be saved in a sub-folder called "Lasso" under "02 results" folder.

The code will automatically create a new folder called "Lasso" if it is the first time you run the code.

The code will run loops thru the set of dependent variables (markts and products) in parallel on multiple cores of the CPU.

Outcomes of lasso regression models are:

1. Basic metrics for each market/product; it includes RSquare, MSE, MAPE and MAE, all\_Rank, RankIn10\_gps; the code will also stack all models performance metrics and save them in "allMetrics.csv" file under the "Lasso" folder.
2. Coefficients of lasso regression model for each market/product;
3. Hit matrix for each market/product;
4. Lift, effectiveness and hit rate for each market/product;
5. Predicted sales and adjusted predicted sales (3 versions) for each market/product.

In the meantime, the run time will also be recorded and printed out.

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 resultDir <- "./02 results/Lasso"  
 if (file.exists(resultDir)==FALSE) {  
 dir.create(resultDir, showWarnings = TRUE,   
 recursive = TRUE, mode = "0777")  
 }  
   
  
 s <- Sys.time()  
 bParallel <- parallel #Whether or not to do parallel Run  
 if (bParallel) {  
 cl <- makeCluster(cores)  
 registerDoParallel(cl, cores = cores)  
 }  
   
 all\_lasso <- foreach(mktid = 1:length(var\_response),   
 .packages = c("caret", "plyr", "dplyr",   
 "Metrics",   
 "glmnet")) %dopar% {  
 lassoModel(type = "original",  
 standardize = "TRUE",   
 responseid = mktid,   
 numberOfpart = nFolds,   
 cv = cvFolds)  
 lassoModel(type = "log",   
 standardize = "TRUE",   
 responseid = mktid,   
 numberOfpart = nFolds,   
 cv = cvFolds)  
 }  
 if (bParallel) {  
 stopCluster(cl)  
 }  
   
  
 allMetrics <- combineResults(resultDir)  
 write.csv(allMetrics, paste(resultDir, "allMetrics.csv", sep = "/"),  
 na = "",  
 row.names = FALSE)  
   
 e <- Sys.time()  
 cat("Lasso Models finished! using time:", e-s, attr(e-s, "units"), "\n")  
 lasso\_time <- data.frame(Model = "Lasso",  
 nfolds = nFolds,  
 cvfolds = cvFolds,  
 usingTime = e-s)  
 lasso\_time$usingTime <- paste(round(lasso\_time$usingTime,4),   
 attr(lasso\_time$usingTime, "units"), sep=' ')

#### Substep 3.3: Random Forest model

Run random forest models, the code only run the model with original continous predictors as random forest model is not sensitive to changed form of predictors.

The outcomes of random forest models will be saved in a sub-folder called "RF" under "02 results" folder.

The code will automatically create a new folder called "RF" if it is the first time you run the code.

The code will run loops thru the set of dependent variables (markts and products) in parallel on multiple cores of the CPU.

Outcomes of random forest models are:

1. Basic metrics for each market/product; it includes RSquare, MSE, MAPE and MAE, all\_Rank, RankIn10\_gps; the code will also stack all models performance metrics and save them in "allMetrics.csv" file under the "RF" folder.
2. Coefficients: No longer available;
3. Hit matrix for each market/product;
4. Lift, effectiveness and hit rate for each market/product;
5. Predicted sales and adjusted predicted sales (3 versions) for each market/product.

*Note: 2. Coefficients is no longer available for this type of model.*

In the meantime, the run time will also be recorded and will be printed out; the graphs for Hit Rate, Estimated lift, True Lift and Effectiveness will also be printed out.

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 resultDir <- "./02 results/RF"  
 if (file.exists(resultDir)==FALSE) {  
 dir.create(resultDir, showWarnings = TRUE,   
 recursive = TRUE, mode = "0777")  
 }  
   
   
 s <- Sys.time()  
 for (mktid in 1:length(var\_response)) {  
 cat("market", mktid, "start!\n")  
 sf <- Sys.time()  
 RFModel(responseid = mktid, #Market Number  
 numberOfpart = nFolds,#Number of part  
 cv = cvFolds, #Folds to CV  
 ntree\_min = 300, #Minimun ntrees  
 ntree\_max = 800, #Maximum ntrees  
 mtry\_min = 5, #Minimun mtry  
 mtry\_max = 10, #Maximum mtry  
 nodesize\_min = 5, #Minimun nodesize  
 nodesize\_max = 10 #Maximum nodesize  
 )  
 ef <- Sys.time()  
 cat("market", mktid, "end! using time:", ef-sf, attr(ef-sf, "units"), "\n")  
 }  
   
  
 allMetrics <- combineResults(resultDir)  
 write.csv(allMetrics, paste(resultDir, "allMetrics.csv", sep = "/"),  
 na = "",  
 row.names = FALSE)  
   
 e <- Sys.time()  
 cat("RF Models finished! using time:", e-s, attr(e-s, "units"), "\n")  
 RF\_time <- data.frame(Model = "RF",  
 nfolds = nFolds,  
 cvfolds = cvFolds,  
 usingTime = e-s)  
 RF\_time$usingTime <- paste(round(RF\_time$usingTime,4),   
 attr(RF\_time$usingTime, "units"), sep=' ')

#### Substep 3.4: Gradient boosting model

Run gradient boosting models

The outcomes of random forest models will be saved in a sub-folder called "Boost" under "02 results" folder.

The code will automatically create a new folder called "Boost" if it is the first time you run the code.

The code will run loops thru the set of dependent variables (markts and products) in parallel on multiple cores of the CPU.

Outcomes of gradient boosting models are:

1. Basic metrics for each market/product; it includes RSquare, MSE, MAPE and MAE, all\_Rank, RankIn10\_gps; the code will also stack all models performance metrics and save them in "allMetrics.csv" file under the "Boost" folder.
2. Coefficients: No longer available;
3. Hit matrix for each market/product;
4. Lift, effectiveness and hit rate for each market/product;
5. Predicted sales and adjusted predicted sales (3 versions) for each market/product.

*Note: 2. Coefficients is no longer available for this type of model.*

In the meantime, the run time will also be recorded and will be printed out; the graphs for Hit Rate, Estimated lift, True Lift and Effectiveness will also be printed out.

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 resultDir <- "./02 results/Boost"  
 if (file.exists(resultDir)==FALSE) {  
 dir.create(resultDir, showWarnings = TRUE,   
 recursive = TRUE, mode = "0777")  
 }  
   
   
 s <- Sys.time()  
 for (mktid in 1:length(var\_response)) {  
 cat("market", mktid, "start!\n")  
 sf <- Sys.time()  
 gbmModel(responseid = mktid, #Market Number  
 numberOfpart = nFolds, #Number of part  
 cv = cvFolds, #Folds of CV  
 trees\_min = 5000, #Minimun trees  
 trees\_max = 10000, #Maximun trees  
 shrinkage\_min = 0.001, #Minimun shrinkage  
 shrinkage\_max = 0.01, #Maximun shrinkage  
 depth\_min = 1, #Minimun depth  
 depth\_max = 3, #Maximun depth  
 minobsinnode\_min = 5, #Minimun minobsinnode  
 minobsinnode\_max = 10 #Maximun minobsinnode  
 )  
 ef <- Sys.time()  
 cat("market", mktid, "end! using time:", ef-sf, attr(ef-sf, "units"), "\n")  
 }  
   
 # Combine all metrics results  
 allMetrics <- combineResults(resultDir)  
 write.csv(allMetrics, paste(resultDir, "allMetrics.csv", sep = "/"),  
 na = "",  
 row.names = FALSE)  
   
 e <- Sys.time()  
 cat("GBM Models finished! using time:", e-s, attr(e-s, "units"), "\n")  
 gbm\_time <- data.frame(Model = "GBM",  
 nfolds = nFolds,  
 cvfolds = cvFolds,  
 usingTime = e-s)  
 gbm\_time$usingTime <- paste(round(gbm\_time$usingTime,4),   
 attr(gbm\_time$usingTime, "units"), sep=' ')

#### Substep 3.5: Stack run time for all models

Combine run time of all types of models and write to csv file.

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 all\_time <- rbind(stepwise\_time,  
 lasso\_time,  
 RF\_time,  
 gbm\_time)  
 write.csv(all\_time, "Running time.csv", row.names = F)

### Step 4: Combine results of all models

Combine results of all models.

The code will rank all 6 types of models for the same market/product based on these 5 measures-- RSquare, Overfitting, MSE, MAE and RankIn10\_gps. Then it will calculate weighted sum of counts of rank 1-6 (given there are 6 models for each market/product) for each type of model. And finally it will pick the model with the highest weigthed sum of counts as the best model.

The code will also put together main outputs from each type of model and save them in csv files under "02 results" folder.

In this case study, the optimal models are lasso regression models for all markets and products, so the coefficients are also consolidated and saved under "02 results" folder.

Final output file under "02 results" folder are:

1. allHitLiftEff.csv: Lift, effectiveness and hit rate for all best models;
2. allHitMatix.csv: Hit matrix for all best models;
3. allMetrics.csv: Basic metrics (RSquare, MSE, MAPE and MAE, all\_Rank, RankIn10\_gps) for all best mdoels;
4. allResults.csv: Basic metrics (RSquare, MSE, MAPE and MAE, all\_Rank, RankIn10\_gps), ranks, weighted sum of rank counts for ALL models (regardless it is the best model or not);
5. allCoefs.csv: estimated coefficients of lasso or stepwise regression models if the best model is lasso or stepwise regression model.

**Note: "allCoefs.csv" is available only when stepwise regression model or lasso regressin model is the best one**

###this is only necessary if using R markdown  
setwd(targetFolder)  
print(getwd())###end  
 library(dplyr)  
 modelType <- c("Stepwise", "Lasso", "RF", "Boost")  
 allResults <- numeric()  
 for (i in 1:length(modelType)) {  
 type <- modelType[i]  
 resultsFile <- paste("./02 results/", type, "/allMetrics.csv",sep="")  
 re\_t <- read.csv(resultsFile, header = T)  
 selectVar <- c("PartID","Rsquare\_Training", "MSE\_Training", "MAPE\_Training",  
 "MAE\_Training", "Rsquare\_Test", "MSE\_Test",  
 "MAPE\_Test", "MAE\_Test", "all\_Rank", "RankIn10\_gps",  
 "Market")  
 re <- re\_t[re\_t$PartID=="all",selectVar]  
 re$modelType <- type  
 allResults <- rbind(allResults, re)  
 }  
   
 allResults$overfit <- allResults$Rsquare\_Training / allResults$Rsquare\_Test  
   
 allResults$modelType <- ifelse(grepl("lg\_", as.character(allResults$Market))==TRUE,  
 paste(allResults$modelType, "\_log", sep=""),  
 allResults$modelType)  
   
 allResults$Market <- gsub("lg\_", "", allResults$Market)  
 allResults <- tbl\_df(allResults[with(allResults, order(allResults$Market)),])  
   
   
 allResultsRank <- allResults %>%   
 group\_by(Market) %>%  
 arrange(Market, -Rsquare\_Test) %>%  
 mutate(rsquare\_rank = row\_number()) %>%  
 arrange(Market, overfit) %>%  
 mutate(overfit\_rank = row\_number()) %>%  
 arrange(Market, MSE\_Test) %>%  
 mutate(MSE\_rank = row\_number()) %>%  
 arrange(Market, MAE\_Test) %>%  
 mutate(MAE\_rank = row\_number()) %>%  
 arrange(Market, RankIn10\_gps) %>%  
 mutate(RankIn10\_gps\_rank = row\_number())  
   
  
 rankval <- names(allResultsRank)[grep("\_rank",names(allResultsRank))]   
  
 for (i in 1:6) {  
 eval(parse(text=paste("allResultsRank$count", i, " <- apply(allResultsRank[, rankval], 1, function(x) sum(x==", i, "))", sep="")))  
 }  
 # Select all rank variables to calculate the weighted scores  
 cntval <- names(allResultsRank)[grep("count",names(allResultsRank))]  
   
 rankweight <- matrix(c(1, 0.7, 0.5, 0.2, 0.1, 0), ncol = 1)  
 allResultsRank$Scores <- data.matrix(allResultsRank[, cntval]) %\*% rankweight  
   
 allResultsRank <- allResultsRank[with(allResultsRank, order(Market, -Scores, -count1, -count2)),]  
   
 allResultsRank <- allResultsRank %>%   
 group\_by(Market) %>%  
 mutate(Scores\_rk = row\_number())  
 allResultsRank$model\_select <- ifelse(allResultsRank$Scores\_rk == 1, "Select model", "Drop")  
   
 allResultsFile <- "./02 results/allResults.csv"  
 file.remove(allResultsFile)  
   
 output <- data.frame(allResultsRank[allResultsRank$model\_select == "Select model",c("Market","modelType")])  
 output$folder <- gsub("\_log", "", output$modelType)  
 output$lg\_type <- gsub(paste(unique(output$folder), sep = "", collapse = "|"),  
 "", output$modelType)  
   
 coeff <- numeric()  
 metrics <- numeric()  
 hitlift <- seq(0,1,0.01)  
 hitMatrix <- numeric()  
   
 for (i in 1:nrow(output)){  
 if (i == 1) {  
 cat("Out the the final model selection resuts...\n")  
 cat("=====================\n")  
 }  
 cat("Market: ", as.character(output$Market[i]), "\n")  
 cat(" Select Model: ", as.character(output$modelType[i]), "\n")  
 cat("=====================\n")  
 folder <- output$folder[i]  
 mkt <- output$Market[i]  
 if (output$lg\_type[i] == "") {  
 coef\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_", mkt,  
 "\_Coefficients.csv", sep=""))  
 coef\_temp$Market <- mkt  
 met\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_", mkt,  
 "\_Basic\_Metrics.csv", sep=""))  
 met\_temp <- met\_temp[met\_temp$PartID == "all",]  
 met\_temp$Market <- mkt  
   
 hel\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_", mkt,  
 "\_hitLiftEffect.csv", sep=""))  
 hel\_temp <- hel\_temp[,-1]  
 names(hel\_temp) <- paste0(mkt, "\_",names(hel\_temp))  
   
 hit\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_", mkt,  
 "\_hitMatrix.csv", sep=""))  
 hit\_temp$Market <- mkt  
 } else {  
 coef\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_lg\_", mkt,  
 "\_Coefficients.csv", sep=""))  
 coef\_temp$Market <- mkt  
 met\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_lg\_", mkt,  
 "\_Basic\_Metrics.csv", sep=""))  
 met\_temp <- met\_temp[met\_temp$PartID == "all",]  
 met\_temp$Market <- mkt  
   
 hel\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_lg\_", mkt,  
 "\_hitLiftEffect.csv", sep=""))  
 hel\_temp <- hel\_temp[,-1]  
 names(hel\_temp) <- paste0(mkt, "\_",names(hel\_temp))  
   
 hit\_temp <- read.csv(paste("./02 results/", folder,   
 "/pharma\_panel\_lg\_", mkt,  
 "\_hitMatrix.csv", sep=""))  
 hit\_temp$Market <- mkt  
 }  
   
 coeff <- rbind(coeff, coef\_temp)  
 metrics <- rbind(metrics, met\_temp)  
 hitlift <- cbind(hitlift, hel\_temp)  
 hitMatrix <- rbind(hitMatrix, hit\_temp)  
   
 }  
  
   
 write.csv(data.frame(allResultsRank), allResultsFile,  
 na = "",  
 row.names = FALSE)  
  
 write.csv(data.frame(coeff), "./02 results/allCoefs.csv",  
 na = "",  
 row.names = FALSE)  
  
 write.csv(data.frame(metrics), "./02 results/allMetrics.csv",  
 na = "",  
 row.names = FALSE)  
  
 write.csv(data.frame(hitlift), "./02 results/allHitLiftEff.csv",  
 na = "",  
 row.names = FALSE)  
  
 write.csv(data.frame(hitMatrix), "./02 results/allHitMatix.csv",  
 na = "",  
 row.names = FALSE)