```
## Zapier Assignment :: Steam Games Recommendation Engine
## Author: Dhivya R

## Read the data
setwd("E:/Dhivya/Zapier")
library(data.table)
data <- fread("steam-200k.csv")
data$V5 <- NULL
## Add column names
colnames(data) <- c("user_id","game_title","behavior_name","value")

## Unique # of users- 12393
length(unique(data$user_id))

## Unique # of games- 5155
length(unique(data$game_title))

## Modify the data frame as follows:
## Create variable: played_hours= value if behavior_name= "play" or =0 if behavior_name= "purchase"
data$played_hours <- ifelse(data$behavior_name=="play",data$value,0)
## Create binary variables purchased and played for each behavior of the user
data$purchased <- ifelse(data$behavior_name=="purchase",1,0)
data$played <- ifelse(data$behavior_name=="play",1,0)
## Collapse the dataframe along the user_id and game_title
library(plyr)
collapse <- ddply(data,.(user_id, game_title),summarize,played_hours=sum(played_hours), purchased=sum(purchased), played=sum(played))

## Collapse again to find the number of games purchased, number of games played and hours played for each user
user <- ddply(data,.(user_id),summarize, number_games_purchased= sum(purchased), number_games_played=sum(played), overall_hours=sum(played_hours))
## Merge back/ Join with the original dataframe by user_id
collapse <- merge(collapse, user, by="user_id")
collapse$purchased[collapse$purchased==2] <- 1
collapse$played[collapse$played==2] <- 1

## Expand the dataset for every user game combination
library(dplyr)
library(tidyr)
expanded <- collapse %>%  expand(user_id, game_title)%>% left_join(collapse)
expanded$purchased[is.na(expanded$purchased)] <- 0

## Data is highly imbalanced, so perform undersampling, extract 100% of signal and a random fraction of the noise
expanded_signal <- expanded[expanded$purchased==1,]
expanded_noise <- expanded[expanded$purchased==0,]
expanded_noise_sample <- expanded_noise[sample(nrow(expanded_noise),500000),]
data <- rbind(expanded_signal, expanded_noise_sample)
data$played[is.na(data$played)] <- 0

## Convert games variable to lower case
data$game_title <- tolower(data$game_title)

## Clean up the games variable by removing the special characters, punctuations and white spaces as this will serve as the
## joining key for game level variables
library(stringr)
data$game_title<-str_replace_all(data$game_title, "[[:punct:]]", "")
data$game_title <- str_replace_all(data$game_title, fixed(" "), "")

## Steam game data extracted from Steam API and hosted by open source project at: https://data.world/craigkelly/steam-game-data
## This data is not part of the original data, but is collected seperately
games_features <- fread("games-features.csv")

## Similarly clean the games variable in this table
games_features$ResponseName <- tolower(games_features$ResponseName)
games_features$ResponseName <-str_replace_all(games_features$ResponseName, "[[:punct:]]", "")
games_features$ResponseName <- str_replace_all(games_features$ResponseName, fixed(" "), "")

## Match the column names
names(games_features)[names(games_features) == 'ResponseName'] <- 'game_title'
`%nin%` <- function (x, table) match(x, table, nomatch = 0L) == 0L

## Inner join for simplicity and to concentrate on games which have data
data <- merge(data, games_features, by="game_title")

## Bivariate Study and Feature Engineering
source("http://pcwww.liv.ac.uk/~william/R/crosstab.r")
crosstab(data, row.vars = "purchased", col.vars = 'played', type=c("f","r"))
data$number_games_purchased_bucket <- cut(data$number_games_purchased, c(0,1,5,10,50,100,200,500,2000), include.lowest = TRUE)
crosstab(data, row.vars = "number_games_purchased_bucket", col.vars = 'played', type=c("f","r"))
data$number_games_played_bucket <- cut(data$number_games_played, c(0,1,5,10,50,100,200,500,2000), include.lowest = TRUE)
crosstab(data, row.vars = "number_games_played_bucket", col.vars = 'played', type=c("f","r"))
data$overall_hours_bucket <- cut(data$overall_hours, c(0,10,200,1000,2000,15000), include.lowest = TRUE)
crosstab(data, row.vars = "overall_hours_bucket", col.vars = 'played', type=c("f","r"))
data$NoAgeRestriction <- ifelse(data$RequiredAge==0,1,0)
crosstab(data, row.vars = "NoAgeRestriction", col.vars = 'played', type=c("f","r"))
data$SixteenPlusAge <- ifelse(data$RequiredAge>=16,1,0)
crosstab(data, row.vars = "SixteenPlusAge", col.vars = 'played', type=c("f","r"))
data$metacritic_bucket <- cut(data$Metacritic, c(0,50,100), include.lowest = TRUE)
crosstab(data, row.vars = "metacritic_bucket", col.vars = 'played', type=c("f","r"))

data$recommendation_bucket <- cut(data$RecommendationCount, c(0,250,500,750,1000,3000,10000,10000000), include.lowest = TRUE)
crosstab(data, row.vars = "recommendation_bucket", col.vars = 'played', type=c("f","r"))
data$screenshot_bucket <- cut(data$ScreenshotCount, c(0,5,10,15,200), include.lowest = TRUE)
crosstab(data, row.vars = "screenshot_bucket", col.vars = 'played', type=c("f","r"))
data$SteamSpyOwnersInMillions <- data$SteamSpyOwners/1000000
data$SteamSpyPlayersEstimateInMillions <- data$SteamSpyPlayersEstimate/1000000
data$owners_bucket <- cut(data$SteamSpyOwnersInMillions, c(0,0.05,0.1,0.2,0.4,0.8,1,100), include.lowest = TRUE)
```

```
crosstab(data, row.vars = "owners_bucket", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "IsFree", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "PurchaseAvail", col.vars = 'played', type=c("f","r"))

data$PlatformWindows1 <- ifelse(data$PlatformWindows==TRUE,1,0)
data$PlatformLinux1 <- ifelse(data$PlatformLinux==TRUE,2,0)
data$PlatformMac1 <- ifelse(data$PlatformMac==TRUE,4,0)
data$Platform <- data$PlatformLinux1+ data$PlatformMac1+ data$PlatformWindows1
data$Platform <- ifelse(data$Platform==1,"OnlyWindows",ifelse(data$Platform==3,"Windows+Linux",ifelse(data$Platform==5,"Windows+Mac","Windows+Linux+Mac")))
crosstab(data, row.vars = "Platform", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "CategorySinglePlayer", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "CategoryMultiplayer", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "CategoryCoop", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "CategoryMMO", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "CategoryIncludeLevelEditor", col.vars = 'played', type=c("f","r"))
data[,c(29:63)] = apply(data[,c(29:63)], 2, function(x) as.integer(as.logical(x)))
data$Genre <- ifelse(data$GenreIsNonGame=="NonGame",1,ifelse(data$GenreIsIndie==1,"Indie",ifelse(data$IsAction==1,"Action",ifelse(data$GenreIsAdventure==1,"Adventure",
                                                                                                                                  ifelse(data$GenreIsCasual==1,"Casual",ifelse(data$GenreIsStrategy==1,"Strategy",ifelse(data$GenreIsRPG==1,"RPG",

ifelse(data$GenreIsSimulation=="Simulation",ifelse(data$GenreIsEarlyAccess==1,"EarlyAccess",ifelse(data$GenreIsFreeToPlay==1,"FreeToPlay",ifelse(data$GenreIsSports==1,"Sports",ifelse(data$GenreIsRacing,"Racing","MassivelyMultiPlayer")))))))))))
crosstab(data, row.vars = "GenreIsIndie", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsAction", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsAdventure", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsCasual", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsStrategy", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsSimulation", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsAction", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsEarlyAccess", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsFreeToPlay", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsSports", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsRacing", col.vars = 'played', type=c("f","r"))
crosstab(data, row.vars = "GenreIsMassivelyMultiplayer", col.vars = 'played', type=c("f","r"))

data$PriceFinalBucket <- cut(data$PriceFinal, c(0,1,5,10,20,50,100), include.lowest = TRUE)
crosstab(data, row.vars = "PriceFinalBucket", col.vars = 'played', type=c("f","r"))
data$PriceInitialBucket <- cut(data$PriceInitial, c(0,1,5,10,20,50,100,200), include.lowest = TRUE)
crosstab(data, row.vars = "PriceInitialBucket", col.vars = 'played', type=c("f","r"))
data$French <- grepl("French",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "French", col.vars = 'played', type=c("f","r"))
data$German <- grepl("German",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "German", col.vars = 'played', type=c("f","r"))
data$Italian <- grepl("Italian",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "Italian", col.vars = 'played', type=c("f","r"))
data$Russian <- grepl("Russian",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "Russian", col.vars = 'played', type=c("f","r"))
data$Spanish <- grepl("Spanish",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "Spanish", col.vars = 'played', type=c("f","r"))
data$Korean <- grepl("Korean",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "Korean", col.vars = 'played', type=c("f","r"))
data$Japanese <- grepl("Japanese",data$SupportedLanguages,1,0)
crosstab(data, row.vars = "Japanese", col.vars = 'played', type=c("f","r"))

library(lubridate)
data$ReleaseYear <- year(as.Date(data$ReleaseDate, format='%B %d %Y'))
data$ReleaseYear[data$ReleaseYear==10] <- 2010
data$ReleaseYear[data$ReleaseYear==11] <- 2011
data$ReleaseYear[data$ReleaseYear==12] <- 2012
data$ReleaseYear[data$ReleaseYear==13] <- 2013
data$ReleaseYear[data$ReleaseYear==14] <- 2014
data$ReleaseYear[data$ReleaseYear==15] <- 2015
data$By2010 <- ifelse(data$ReleaseYear <= 2010,1,0)
crosstab(data, row.vars = "By2010", col.vars = 'played', type=c("f","r"))

purchased <- data[data$purchased==1,]
## Find the top games basis purchase and basis hours played
head(sort(table(purchased$game_title),decreasing=T),10)
x <- ddply(purchased[,c("game_title","played_hours")],.(game_title),summarize,HoursSpent=sum(played_hours))
head(arrange(x,desc(x$HoursSpent)),10)
collapse <- ddply(purchased[,c("user_id","game_title","purchased")],.(user_id),summarize,GamesPurchased=sum(purchased),Titles=paste(game_title, collapse=", "))

## Create aggregate variables if users have purchased these top games
collapse$PurchasedDota2 <- grepl("dota2",collapse$Titles,1,0)
collapse$Purchasedcounterstrikeglobaloffensive <- grepl("counterstrikeglobaloffensive",collapse$Titles,1,0)
collapse$Purchasedteamfortress2 <- grepl("teamfortress2",collapse$Titles,1,0)
collapse$Purchasedcounterstrike <- grepl("counterstrike",collapse$Titles,1,0)
collapse$Purchasedcounterstrikesource <- grepl("counterstrikesource",collapse$Titles,1,0)
collapse$Purchasedtheelderscrollsvskyrim <- grepl("theelderscrollsvskyrim",collapse$Titles,1,0)
collapse$Purchasedgarrysmod <- grepl("garrysmod",collapse$Titles,1,0)
collapse$Purchasedfalloutnewvegas <- grepl("falloutnewvegas",collapse$Titles,1,0)
collapse$Purchasedleft4dead2 <- grepl("left4dead2",collapse$Titles,1,0)
collapse$Purchasedtotalwarshogun2 <- grepl("totalwarshogun2",collapse$Titles,1,0)
collapse$Purchasedcounterstrikeconditionzero <- grepl("counterstrikeconditionzero",collapse$Titles,1,0)
collapse$Purchasedportal2 <- grepl("portal2",collapse$Titles,1,0)
collapse$Purchasedunturned <- grepl("unturned",collapse$Titles,1,0)
collapse$Purchasedportal <- grepl("portal",collapse$Titles,1,0)
collapse$Purchasedcounterstrikeglobaloffensive <- grepl("counterstrikeglobaloffensive",collapse$Titles,1,0)
collapse$Purchasedhalflife2lostcoast <- grepl("halflife2lostcoast",collapse$Titles,1,0)

## Merge again to bring all 100+ variables together
data_new <- merge(data,collapse,by="user_id",all.x=T)
crosstab(data_new, row.vars = "Purchasedportal", col.vars = 'played', type=c("f","r"))

## Using only variables which had a positive/ negative or normal trend in the bivariate report
features <- c("PurchasedDota2","Purchasedcounterstrikeglobaloffensive","Purchasedteamfortress2","Purchasedcounterstrike","Purchasedcounterstrikesource",
              "Purchasedtheelderscrollsvskyrim","Purchasedgarrysmod","Purchasedfalloutnewvegas","Purchasedleft4dead2" ,"Purchasedtotalwarshogun2","Purchasedcounterstrikeconditionzero",
```

```
              "Purchasedportal2","Purchasedunturned","Purchasedportal","Purchasedhalflife2lostcoast","GamesPurchased","NoAgeRestriction","SixteenPlusAge","Metacritic","RecommendationCount","ScreenshotCount",
              "SteamSpyOwnersInMillions","IsFree","PurchaseAvail", "Platform","CategorySinglePlayer","CategoryMultiplayer","CategoryCoop","CategoryMMO","CategoryIncludeLevelEditor",
              "GenreIsIndie","GenreIsAction","GenreIsAdventure","GenreIsCasual","GenreIsStrategy","GenreIsSimulation","GenreIsAction","GenreIsEarlyAccess","GenreIsFreeToPlay","GenreIsSports","GenreIsRacing","GenreIsMassivelyMultiplayer",
              "PriceFinal","PriceInitial","French","German","Italian","Spanish","Russian","Korean","Japanese")

## Set seed
set.seed(1234)

## Divide the dataset into train and test (50:50)
index <- sample(nrow(data_new), nrow(data_new)/2)
train <- data_new[index,]
test <- data_new[-index,]

## Build a reproducible XGBoost model over the train dataset
library(xgboost)
model_steam <- xgboost(data        = data.matrix(train[,features]),
                       label       = train$played,
                       silent=0,
                       booster="gbtree",
                       eta = 0.01,
                       max_depth = 4,
                       min_child_weight=6, max_delta_step=10,
                       nrounds     = 1000,
                       subsample = 0.5,
                       colsample_bytree = 0.5,
                       seed = 1234,
                       objective   = "binary:logistic",
                       eval_metric = "auc")

## Identify the most important variables
xgb.importance(feature_names = features, model=model_steam)

## Save the model
save(model_steam, file="Model.RData")

## Steps to be done: Parameter tuning and more iterations of the model to only use 12-15 most predictive variables

## Predict over the test dataset
test$Prediction <- predict(model_steam, data.matrix(test[,features]))

## Approximate for confusion matrix statistics
test$PredApprox <- ifelse(test$Prediction>0.5,1,0)
library(caret)
confusionMatrix(as.factor(test$PredApprox),as.factor(test$played))

## Rank Ordering Algorithm
## Take actual and predicted values in a dataframe
my_solution <- data.frame(test$Prediction,test$played)

## Sort the dataframe by descending values of prediction probabilities
sorted.first <- my_solution[ order(-my_solution[,1]),]

## Split into 10 sample dataframes
for(i in 1:10) {
  print(paste(21600,'x', i, '=', 21600*i))
}

sample.1 <- sorted.first[1:21600,]
sample.2 <- sorted.first[21601:43200,]
sample.3 <- sorted.first[43201:64800,]
sample.4 <- sorted.first[64801:86400,]
sample.5 <- sorted.first[86401:108000,]
sample.6 <- sorted.first[108001:129600,]
sample.7 <- sorted.first[129601:151200,]
sample.8 <- sorted.first[151201:172800,]
sample.9 <- sorted.first[172801:194400,]
sample.10 <- sorted.first[194401:215998,]

## Build the rank ordering table
Samples <- c("sample.1","sample.2","sample.3","sample.4","sample.5","sample.6","sample.7","sample.8","sample.9","sample.10")
CountOfUserGameCombinations = c(nrow(sample.1),nrow(sample.2),nrow(sample.3),nrow(sample.4),nrow(sample.5),nrow(sample.6),nrow(sample.7),nrow(sample.8),nrow(sample.9),nrow(sample.10))
CumulativeCountOfUserGameCombinations <- c(CountOfUserGameCombinations[1], CountOfUserGameCombinations[1]+CountOfUserGameCombinations[2],
CountOfUserGameCombinations[1]+CountOfUserGameCombinations[2]+CountOfUserGameCombinations[3],CountOfUserGameCombinations[1]+CountOfUserGameCombinations[2]+CountOfUserGameCombinations[3]+CountOfUserGameCombinations[4],CountOfUserGameCombinations[1]+CountOfUserGameCombinatio


CountOfPlays <-
c(sum(sample.1$test.played[sample.1$test.played==1]),sum(sample.2$test.played[sample.2$test.played==1]),sum(sample.3$test.played[sample.3$test.played==1]),sum(sample.4$test.played[sample.4$test.played==1]),sum(sample.5$test.played[sample.5$test.played==1]),sum(sample.6$tes

CumulativeCountOfPlays <- c(sum(CountOfPlays[1]),sum(CountOfPlays[1:2]),sum(CountOfPlays[1:3]),sum(CountOfPlays[1:4]),sum(CountOfPlays[1:5]),sum(CountOfPlays[1:6]),sum(CountOfPlays[1:7]),sum(CountOfPlays[1:8]),sum(CountOfPlays[1:9]),sum(CountOfPlays[1:10]))
MinScore <- (c(min(sample.1[,1]),min(sample.2[,1]),min(sample.3[,1]),min(sample.4[,1]),min(sample.5[,1]),min(sample.6[,1]),min(sample.7[,1]),min(sample.8[,1]),min(sample.9[,1]),min(sample.10[,1])))*100
MaxScore <- (c(max(sample.1[,1]),max(sample.2[,1]),max(sample.3[,1]),max(sample.4[,1]),max(sample.5[,1]),max(sample.6[,1]),max(sample.7[,1]),max(sample.8[,1]),max(sample.9[,1]),max(sample.10[,1])))*100
MeanScore <- (c(mean(sample.1[,1]),mean(sample.2[,1]),mean(sample.3[,1]),mean(sample.4[,1]),mean(sample.5[,1]),mean(sample.6[,1]),mean(sample.7[,1]),mean(sample.8[,1]),mean(sample.9[,1]),mean(sample.10[,1])))*100

RankingTable <- data.frame(Samples,CountOfUserGameCombinations,CumulativeCountOfUserGameCombinations,CountOfPlays,CumulativeCountOfPlays,MinScore,MaxScore,MeanScore)

RankingTable$UserGamesComboShare <- round((CountOfUserGameCombinations/215998)*100,0)
RankingTable$CumulativeUserGamesComboShare <- round(CumulativeCountOfUserGameCombinations/215998*100,0)

RankingTable$PlaysShare <- (CountOfPlays/28550)*100
RankingTable$CumulativePlaysShare <- CumulativeCountOfPlays/28550*100

RankingTable <- RankingTable[,c(1,2,3,9,10,4,5,11,12,6,7,8)]
RankingTable$CumulativeLift <-
c(RankingTable$CumulativePlaysShare[1]/10,RankingTable$CumulativePlaysShare[2]/20,RankingTable$CumulativePlaysShare[3]/30,RankingTable$CumulativePlaysShare[4]/40,RankingTable$CumulativePlaysShare[5]/50,RankingTable$CumulativePlaysShare[6]/60,RankingTable$CumulativePlaysSha
```