PH125.9x: Data Science: Capstone-Project 1

Jeff Roberts

January 4, 2019

Overview

This program takes the MovieLens dataset, as provided by the Capstone-Project: All Learners as provided by EDX course PH125.9x: Data Science code which creates an edx(train) and Validation (test) Sets. The program below then process the data, splits the edx file into train and validation sets, generate a Random Tree Model (using the ranger package), calculates a Confusion Matrix/accuracy based on the validation set and then uses the model to predict Movie Ratings from the Class Validation set and save the file as a new file, submission.csv.

I tried to use cross-validation in model selection for this project, but due to memory constraints on my laptop use the ranger package for Random Forest without it, as tuning in caret always crashed my system.I also did data tuning for ranger on a smaller subset of the data and didn't find a material improvement in accuracy with 100, 500 or 2000 trees so I went with the lowest number 100.

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## Warning: package 'tidyverse' was built under R version 3.5.2

## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
```

```r
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

# Learners will develop their algorithms on the edx set
# For grading, learners will run algorithm on validation set to generate ratings

validation <- validation %>% select(-rating)

# Ratings will go into the CSV submission file below:

rm(dl, ratings, movies, test_index, temp, movielens, removed)
ls()
```

```
## [1] "edx"        "validation"
```

Process Data-break out genre into seperate fields, use first three columns as factors, impute missing

```r
##################################################################################Process Data
library(anytime)
```

```
## Warning: package 'anytime' was built under R version 3.5.2
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(doParallel)
```

```
## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: iterators

## Loading required package: parallel
```

```r
cl <- makeCluster(4)
registerDoParallel(cl)


edx$genres<-as.factor(edx$genres)
edx$rating<-as.factor(edx$rating)

#time stamp as date factor
edx$timestamp<-anydate(edx$timestamp)
edx$date<-as.factor(format(edx$timestamp, "%Y-%m"))
edx$date<-as.factor(edx$date)
edx$userId<-NULL
edx$movieId<-as.integer(edx$movieId)
edx$timestamp<-NULL
edx$title<-NULL
str(edx)
```

```
## 'data.frame':    9000055 obs. of  4 variables:
```

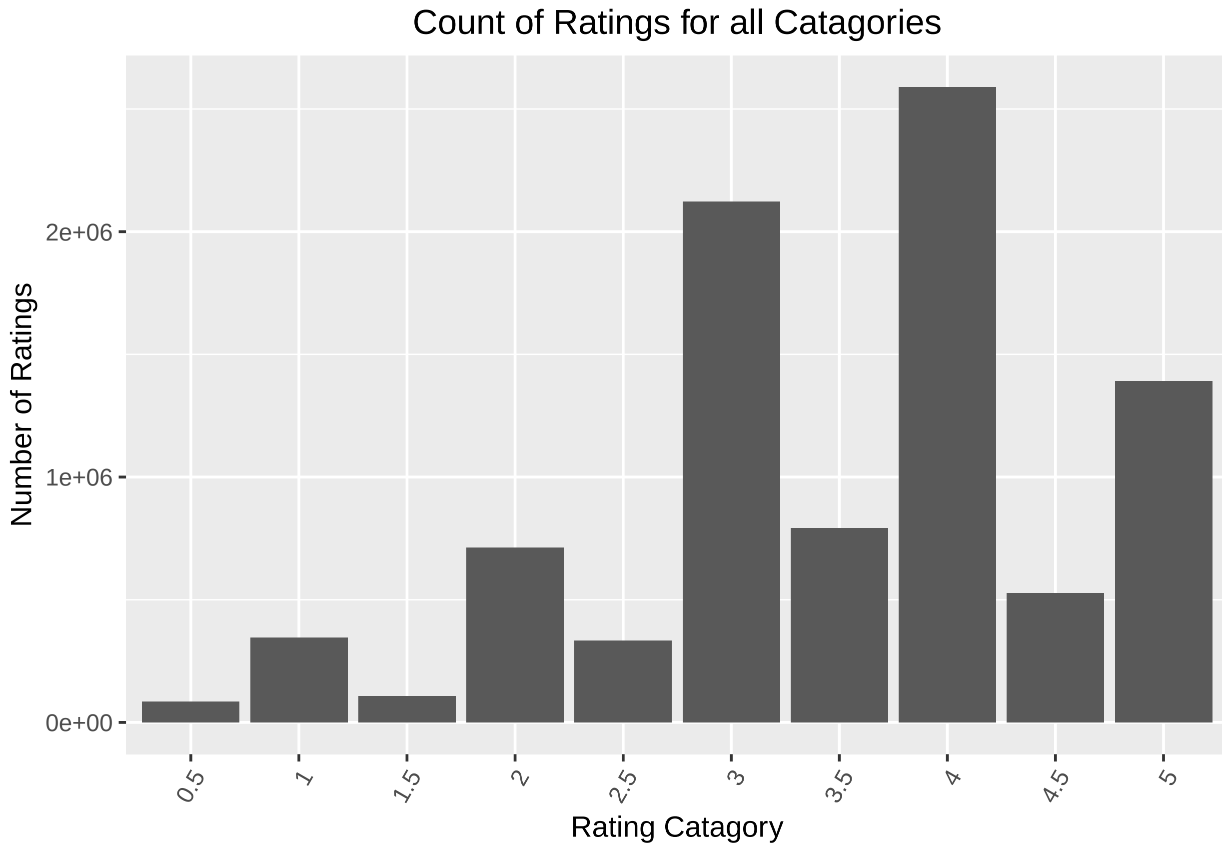
```
##  $ movieId: int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating : Factor w/ 10 levels "0.5","1","1.5",..: 10 10 10 10 10 10 10 10 10 10 ...
##  $ genres : Factor w/ 797 levels "(no genres listed)",..: 577 187 210 98 71 460 542 309 274 120 ...
##  $ date   : Factor w/ 157 levels "1995-01","1996-01",..: 9 9 9 9 9 9 9 9 9 9 ...
```

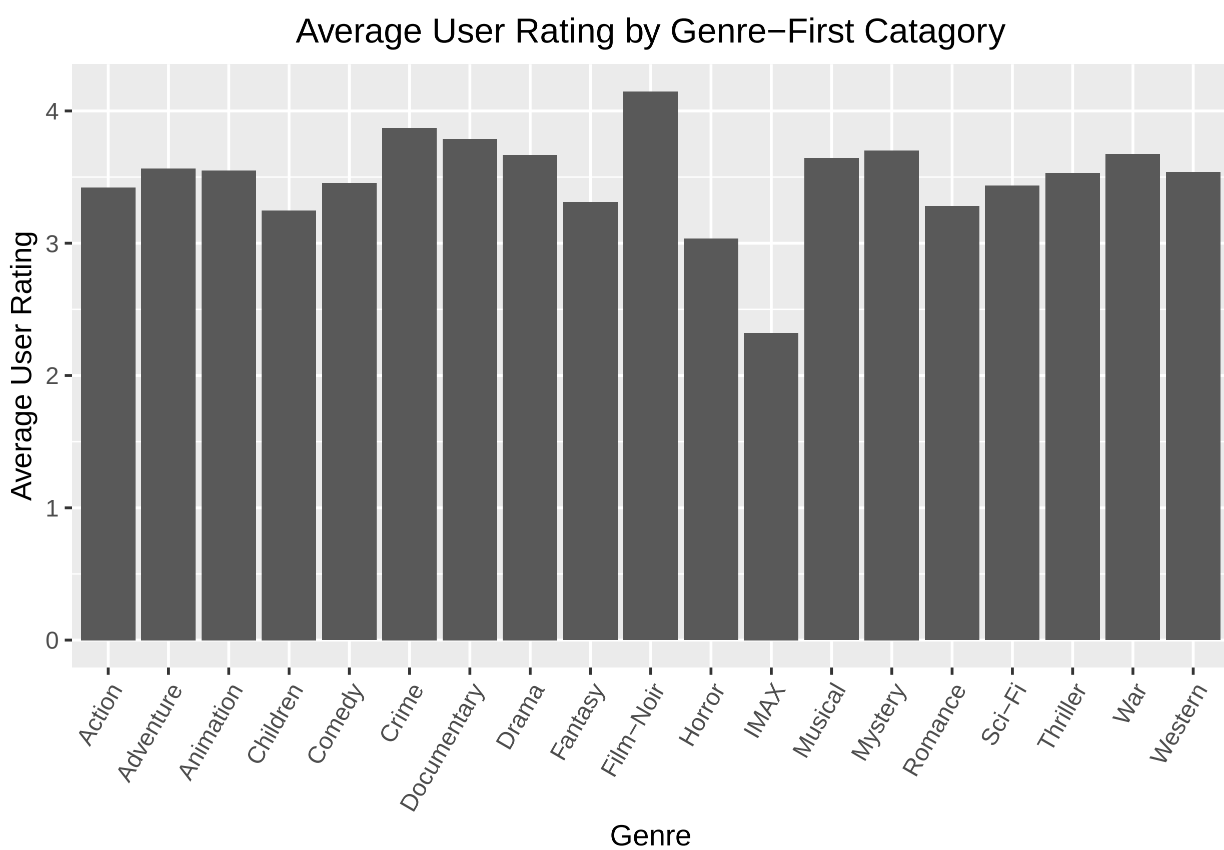
```r
#####################break out genre, use first three columns as factors, impute missing
temp <- as.data.frame(edx$genres, stringsAsFactors=FALSE)
temp2 <- as.data.frame(tstrsplit(temp[,1], '[|]', type.convert=TRUE), stringsAsFactors=FALSE)
colnames(temp2) <- c(1:7)
rm(temp)
temp2[,4:8] <- NULL
temp2 <- as.data.frame(lapply(temp2, factor))

#impute with mode per column
imp<-names(sort(table(temp2[,2]),decreasing=TRUE)[1])
temp2[,2][is.na(temp2[,2])] <- imp
imp1<-names(sort(table(temp2[,3]),decreasing=TRUE)[1])
temp2[,3][is.na(temp2[,3])] <- imp1
temp2[,1][temp2[,1] == "(no genres listed)"] <- "Action"
#Index <- which(temp2$X1 == "(no genres listed)")
temp2[,1]<-as.factor(temp2[,1])






#cbind to edx, remove genre
edx1<-cbind(edx, temp2)
rm(edx)
rm(temp2)
edx1$genres<-NULL

rm(cl)
rm(imp)
rm(imp1)

aa<-as.data.frame(edx1 %>%
  group_by(rating) %>%
  summarise (n = n()) %>%
  mutate(percent = n / sum(n)))
aa$percent<-percent(aa$percent)
aa
```

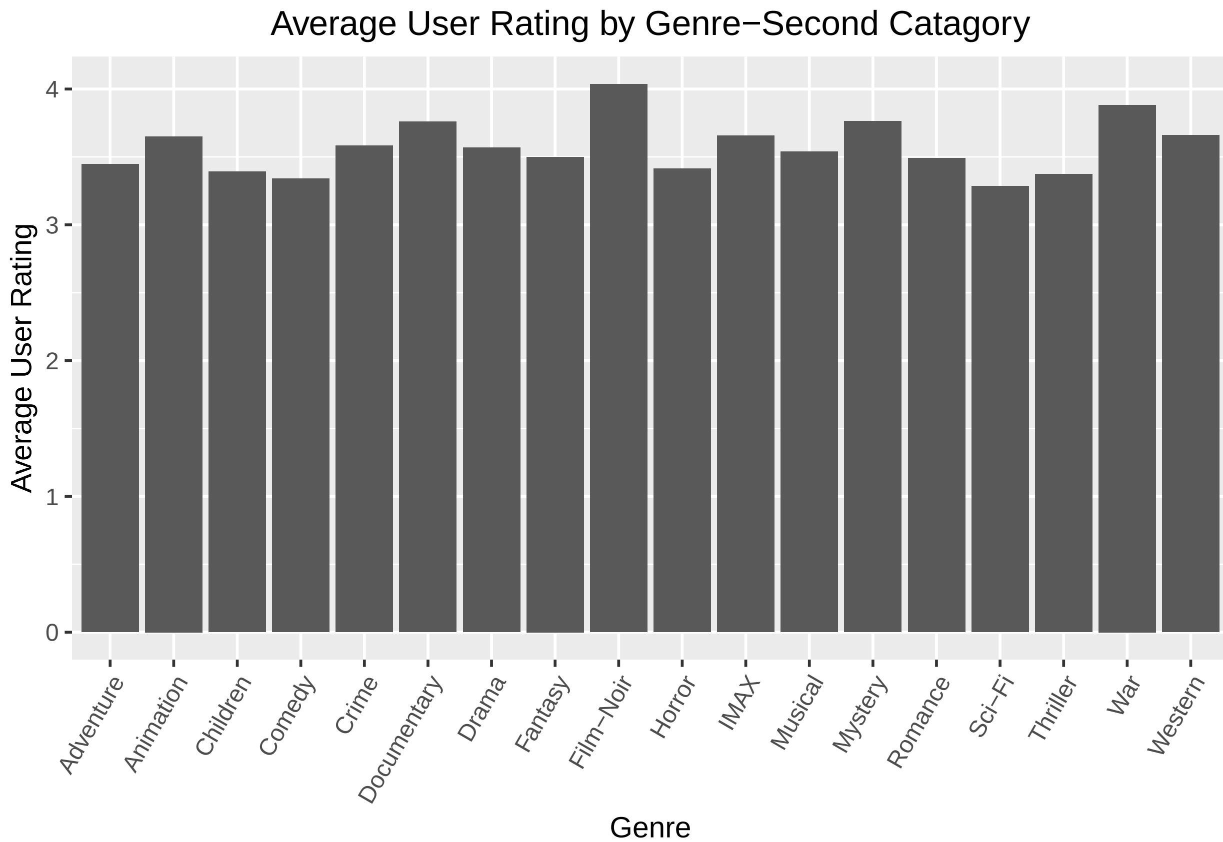
```
##    rating       n percent
## 1     0.5   85374    0.9%
## 2       1  345679    3.8%
## 3     1.5  106426    1.2%
## 4       2  711422    7.9%
## 5     2.5  333010    3.7%
## 6       3 2121240   23.6%
## 7     3.5  791624    8.8%
## 8       4 2588430   28.8%
## 9     4.5  526736    5.9%
## 10      5 1390114   15.4%
```

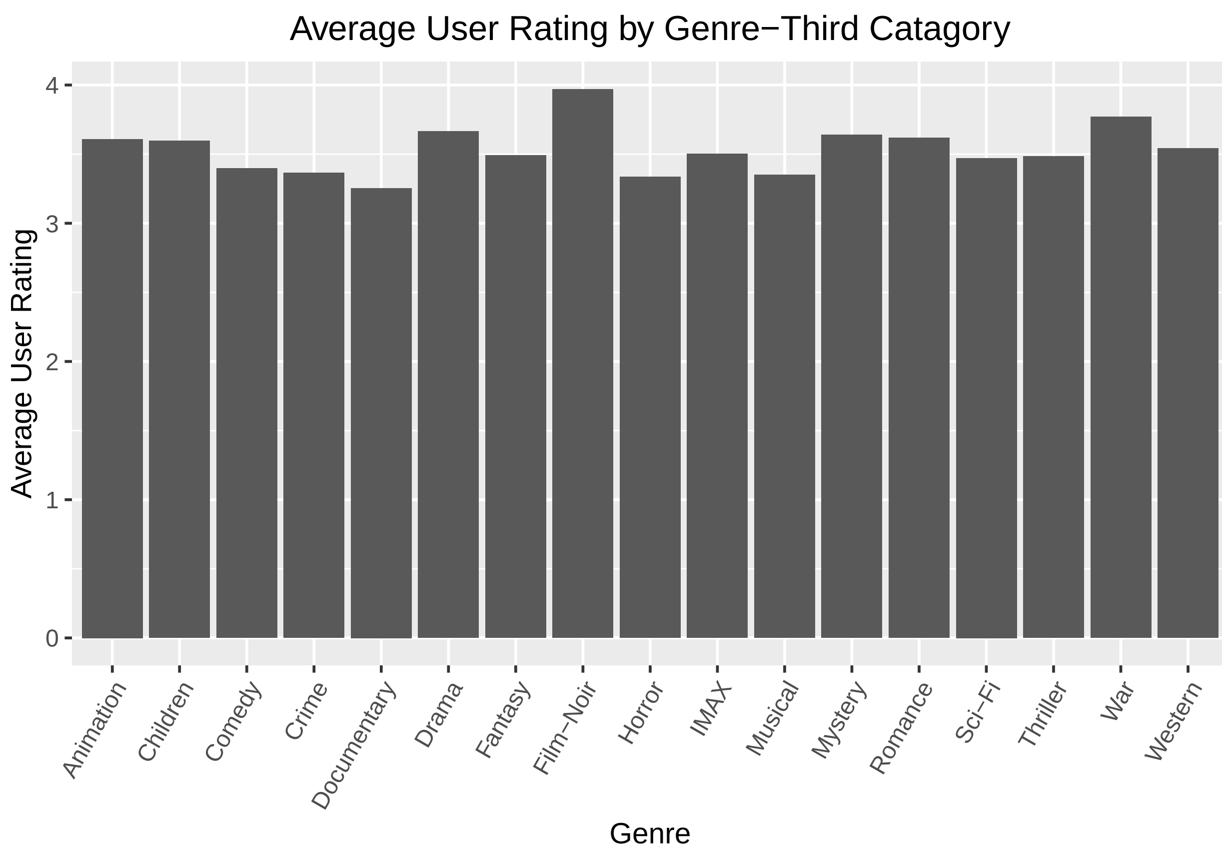
Summary-The ratings show that the largest number of reviewers gave ratings of 3(23.6%), 4 (28.8%) or 5 (15.4%). The lower ratings or x.5 ratings were all under 10% or the ratings given.

Plots-Data Visalization






Average User Rating by Genre−Second Catagory
Average User Rating
0
1
2
3
4
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film−Noir
Horror
IMAX
Musical
Mystery
Romance
Sci−Fi
Thriller
War
Western
Genre



Summary-The "Count of Ratings for all Categorizes" mirrors the proportion table given in the Process Data Chunk. The next three bar graphs show the first three Categories broken out from the genres variable. The first category had The Film-Noir, Mystery and Crime having the highest average rating, the second Film-Noir, War and Documentary with the highest average rating and the third category had Film-Noir and War as the highest category.

Partition Test and Train Sets-Run Random Forest(ranger) and Confusion Matrix

```
test_index1 <- createDataPartition(y = edx1$rating, times = 1, p = 0.3, list = FALSE)
train <- edx1[-test_index1,]
test <- edx1[test_index1,]

rm(test_index1)
rm(edx1)
str(train)
```

```
## 'data.frame':    6300035 obs. of  6 variables:
##  $ movieId: int  122 185 316 355 362 364 370 377 420 466 ...
##  $ rating : Factor w/ 10 levels "0.5","1","1.5",..: 10 10 10 10 10 10 10 10 10 10 ...
##  $ date   : Factor w/ 157 levels "1995-01","1996-01",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ X1     : Factor w/ 20 levels "(no genres listed)",..: 6 2 2 5 3 3 2 2 2 2 ...
##  $ X2     : Factor w/ 18 levels "Adventure","Animation",..: 14 5 1 4 3 2 4 14 4 4 ...
##  $ X3     : Factor w/ 17 levels "Animation","Children",..: 15 15 14 7 13 2 15 15 4 16 ...
```

```
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.5.2
```

```
mod_ranger <- ranger(rating ~ ., data = train, write.forest = TRUE, verbose = FALSE, importance = "impu
mod_ranger
```

```
## Ranger result
##
## Call:
##  ranger(rating ~ ., data = train, write.forest = TRUE, verbose = FALSE,      importance = "impurity"
##
## Type:                             Classification
## Number of trees:                  100
## Sample size:                      6300035
## Number of independent variables:  5
## Mtry:                             2
## Target node size:                 1
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             66.05 %
```

```
######confusion matrix/accuracy
xx <- test[,-2]
yy<- test$rating
rm(test)
RF_pred<-predict(mod_ranger, xx)
```

```
## Predicting.. Progress: 90%. Estimated remaining time: 3 seconds.
```

```
y_hat<-RF_pred$prediction
confusionMatrix(y_hat, yy)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0.5      1    1.5      2    2.5      3    3.5      4    4.5
##        0.5   1073    566    379    463    329    294    186    102     32
##        1      119   4918     87   3059     73   2373     33    908      3
##        1.5      6      6      4     15     13     10      9      3      4
##        2      834   2438    931   3448   1216   3012    659   1006     87
##        2.5    202    235    254    559    553    622    401    248     71
##        3     9362  50917  12495  95274  31888 263698  39369 164606   9156
##        3.5   2257   2759   3448   8039  11931  22537  24752  22200   8115
##        4    11404  38657  13996  95915  52822 314394 167384 514806 130104
##        4.5      9     13      5     34     43    115    230    529    548
##        5      347   3195    329   6621   1035  29317   4465  72121   9901
##           Reference
## Prediction      5
##        0.5     32
##        1      288
##        1.5      5
##        2      207
##        2.5     31
##        3    60157
##        3.5   3991
##        4   247842
##        4.5    456
##        5   104026
```

```
##
## Overall Statistics
##
##                Accuracy : 0.3399
##                  95% CI : (0.3394, 0.3405)
##     No Information Rate : 0.2876
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1189
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 0.5 Class: 1 Class: 1.5 Class: 2 Class: 2.5
## Sensitivity           0.0418928 0.047423  1.253e-04 0.016155  0.0055354
## Specificity           0.9991090 0.997326  1.000e+00 0.995822  0.9989912
## Pos Pred Value        0.3104745 0.414636  5.333e-02 0.249169  0.1741184
## Neg Pred Value        0.9908995 0.963251  9.882e-01 0.921830  0.9631606
## Prevalence            0.0094862 0.038409  1.183e-02 0.079046  0.0370008
## Detection Rate        0.0003974 0.001821  1.481e-06 0.001277  0.0002048
## Detection Prevalence  0.0012800 0.004393  2.778e-05 0.005125  0.0011763
## Balanced Accuracy     0.5205009 0.522375  5.000e-01 0.505988  0.5022633
##                      Class: 3 Class: 3.5 Class: 4 Class: 4.5 Class: 5
## Sensitivity           0.41438   0.104224   0.6630  0.0034679  0.24944
## Specificity           0.77069   0.965370   0.4424  0.9994359  0.94423
## Pos Pred Value        0.35784   0.224959   0.3243  0.2764884  0.44963
## Neg Pred Value        0.81016   0.917862   0.7648  0.9416343  0.87321
## Prevalence            0.23569   0.087958   0.2876  0.0585259  0.15446
## Detection Rate        0.09767   0.009167   0.1907  0.0002030  0.03853
## Detection Prevalence  0.27293   0.040751   0.5879  0.0007341  0.08569
## Balanced Accuracy     0.59253   0.534797   0.5527  0.5014519  0.59683
```

```r
rm(RF_pred)
rm(xx)
rm(yy)
rm(y_hat)
```

The accuracy of this 4 variable model was 33.99%, the Kappa was 0.1188 and it had an OOB prediction error rate of 66.05%.

validation file, process, impute, predict and save results

```r
#################################data processing
#time stamp as date factor
validation$timestamp<-anydate(validation$timestamp)
validation$date<-as.factor(format(validation$timestamp, "%Y-%m"))
validation$date<-as.factor(validation$date)

validation$movieId<-as.integer(validation$movieId)
validation$timestamp<-NULL

validation$genres<-as.factor(validation$genres)
####################break out genre, use first three columns as factors, impute missing
temp <- as.data.frame(validation$genres, stringsAsFactors=FALSE)
```

```r
temp2 <- as.data.frame(tstrsplit(temp[,1], '[|]', type.convert=TRUE), stringsAsFactors=FALSE)
colnames(temp2) <- c(1:7)
rm(temp)
temp2[,4:8] <- NULL
temp2 <- as.data.frame(lapply(temp2, factor))

#impute with mode per column
imp<-names(sort(table(temp2[,2]),decreasing=TRUE)[1])
temp2[,2][is.na(temp2[,2])] <- imp
imp1<-names(sort(table(temp2[,3]),decreasing=TRUE)[1])
temp2[,3][is.na(temp2[,3])] <- imp1
temp2[,1][temp2[,1] == "(no genres listed)"] <- "Action"
#Index <- which(temp2$X1 == "(no genres listed)")
temp2[,1]<-as.factor(temp2[,1])
summary(temp2)
```

```
##          X1                  X2                 X3
##  Action   :284804   Drama    :372312   Thriller:589669
##  Comedy   :270039   Adventure:128440   Sci-Fi  : 74053
##  Drama    :193991   Romance  : 88246   Romance : 72077
##  Adventure: 83742   Comedy   : 68211   Fantasy : 51872
##  Crime    : 58507   Crime    : 67006   Drama   : 50189
##  Horror   : 25966   Thriller : 49050   Comedy  : 36452
##  (Other)  : 82950   (Other)  :226734   (Other) :125687
```

```r
val1<-cbind(validation, temp2)
rm(val, temp2)
```

```
## Warning in rm(val, temp2): object 'val' not found
```

```r
val1$genres<-NULL
val1$title<-NULL
val1$movieId<-as.integer(val1$movieId)
str(val1)
```

```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId: int  231 480 586 151 858 1544 590 4995 34 432 ...
##  $ date   : Factor w/ 157 levels "1995-01","1996-01",..: 9 9 9 20 20 20 120 120 11 11 ...
##  $ X1     : Factor w/ 19 levels "Action","Adventure",..: 5 1 4 1 6 1 2 8 4 2 ...
##  $ X2     : Factor w/ 18 levels "Adventure","Animation",..: 7 1 4 7 7 1 7 13 4 4 ...
##  $ X3     : Factor w/ 17 levels "Animation","Children",..: 15 14 15 13 15 9 17 13 6 17 ...
```

```r
val_pred<-predict(mod_ranger, val1)
y_hat1<-val_pred$prediction



validation$rating<-y_hat1
validation$date<-NULL
validation$timestamp<-NULL
validation$title<-NULL
validation$genres<-NULL
write.csv(validation, file = "submission.csv", row.names=FALSE)
```

The prediction results file was saved as submission.csv.