R Notebook

Code

0 Load Package and Clean up Environment

Hide

```
library(gbm)
library(dplyr)
library(Metrics)
library(ggplot2)
library(corrplot)
library(pROC)
library(tidyverse)
library(randomForest)

# Clean up
rm(list=ls())
cat("\04")
```

1 Import and Merge

Hide

```
# import cleaned data
train_all <- read.csv('data/train_all.csv')
```

2 Explore Data

2.1 Variable Type and Dimension

Hide

```
# Examine Variables
str(train_all)
```

```
'data.frame':	174779 obs. of  106 variables:
 $ User            : int  0 0 0 1 1 1 10 10 100 100 ...
 $ Artist          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Track           : int  1 0 2 1 3 0 2 0 2 0 ...
 $ Rating          : int  32 32 31 28 49 27 30 30 25 61 ...
 $ Time            : int  6 6 6 6 6 6 6 6 6 6 ...
 $ GENDER          : int  0 0 0 0 0 0 0 0 1 1 ...
 $ AGE             : int  82 82 82 66 66 66 73 73 13 13 ...
 $ WORKING         : int  13 13 13 13 13 13 13 13 13 13 ...
 $ REGION          : int  4 4 4 1 1 1 6 6 4 4 ...
 $ MUSIC           : int  3 3 3 1 1 1 3 3 3 3 ...
 $ LIST_OWN        : int  3 3 3 3 3 3 3 3 2 2 ...
 $ LIST_BACK       : int  2 2 2 2 2 2 3 3 2 2 ...
 $ HEARD_OF        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ OWN_ARTIST_MUSIC: int  0 0 0 0 0 0 0 0 0 0 ...
 $ LIKE_ARTIST     : num  48 48 48 48 48 48 48 48 48 48 ...
 $ Q1              : int  11 11 11 9 9 9 86 86 72 72 ...
 $ Q2              : int  9 9 9 5 5 5 28 28 73 73 ...
 $ Q3              : int  29 29 29 7 7 7 72 72 86 86 ...
 $ Q4              : int  7 7 7 14 14 14 33 33 28 28 ...
 $ Q5              : int  8 8 8 14 14 14 50 50 46 46 ...
 $ Q6              : int  7 7 7 55 55 55 32 32 29 29 ...
 $ Q7              : int  8 8 8 12 12 12 78 78 16 16 ...
 $ Q8              : int  10 10 10 3 3 3 100 100 13 13 ...
 $ Q9              : int  51 51 51 88 88 88 30 30 21 21 ...
 $ Q10             : int  49 49 49 6 6 6 6 6 57 57 ...
 $ Q11             : int  8 8 8 10 10 10 32 32 74 74 ...
 $ Q12             : int  10 10 10 7 7 7 33 33 80 80 ...
 $ Q13             : int  9 9 9 5 5 5 9 9 86 86 ...
 $ Q14             : int  9 9 9 8 8 8 10 10 86 86 ...
 $ Q15             : int  9 9 9 4 4 4 9 9 70 70 ...
 $ Q16             : int  10 10 10 6 6 6 56 56 50 50 ...
 $ Q17             : int  28 28 28 6 6 6 56 56 68 68 ...
 $ Q18             : int  42 42 42 42 42 42 42 42 42 42 ...
 $ Q19             : int  41 41 41 41 41 41 41 41 41 41 ...
 $ Uninspired      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Sophisticated   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Aggressive      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Edgy            : int  0 0 0 1 1 1 1 1 0 0 ...
 $ Sociable        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Laid.back       : int  0 0 0 1 1 1 0 0 0 0 ...
 $ Uplifting       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Intriguing      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Free            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Thoughtful      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Outspoken       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Serious         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Unattractive    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Confident       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Youthful        : int  0 0 0 0 0 0 1 1 0 0 ...
 $ Boring          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Current         : int  0 0 0 0 0 0 1 1 1 1 ...
 $ Colourful       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Stylish         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cheap           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Irrelevant      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Heartfelt       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Calm            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Outgoing        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Inspiring       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Beautiful       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Fun             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Authentic       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Credible        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Way.out         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cool            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Catchy          : int  0 0 0 0 0 0 1 1 0 0 ...
 $ Sensitive       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Mainstream      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Superficial     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Annoying        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Passionate      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Not.authentic   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Good.Lyrics     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Background      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Timeless        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Depressing      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Original        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Talented        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Distinctive     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Approachable    : int  0 0 0 1 1 1 0 0 1 1 ...
 $ Genius          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Trendsetter     : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
 $ Noisy           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Upbeat          : int  0 0 0 0 0 0 1 1 0 0 ...
 $ Relatable       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Energetic       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Exciting        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Emotional       : int  0 0 0 0 0 0 0 0 1 1 ...
 $ None.of.these   : int  1 1 1 0 0 0 0 0 0 0 ...
 $ Sexy            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Over            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Rebellious      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Fake            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cheesy          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Popular         : int  0 0 0 0 0 0 1 1 0 0 ...
 $ Superstar       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Relaxed         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Intrusive       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Unoriginal      : int  0 0 0 0 0 0 0 0 0 0 ...
  [list output truncated]
```

Hide

```
# Shape of data
dim(train_all)
```

```
[1] 174779    106
```

Hide

```
# convert categorical data into 'factor'
train_all$REGION <- as.factor(train_all$REGION)
train_all$GENDER <- as.factor(train_all$GENDER)
train_all$WORKING <- as.factor(train_all$WORKING)
```

According to the data description, missing value

2.2 Explore the objective variable - Rating

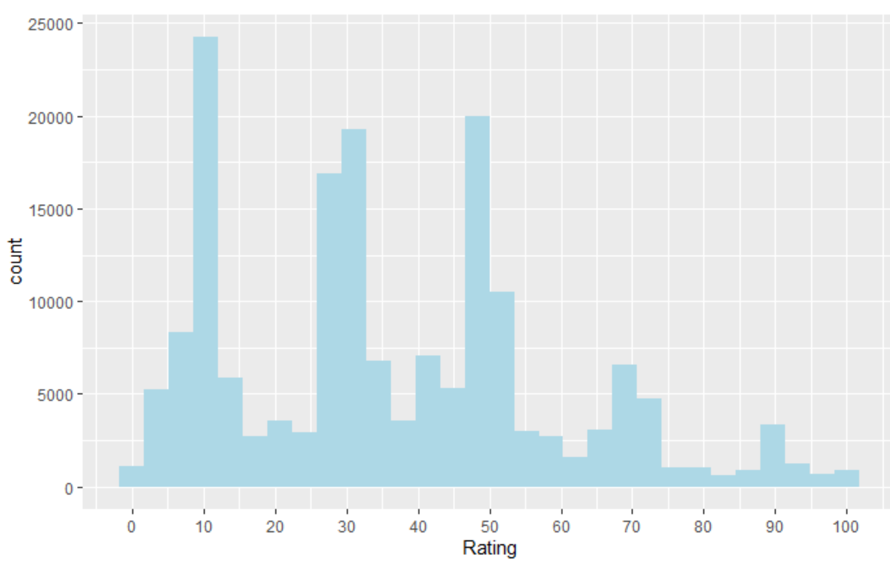
Hide

```
# Explore the objective variable - Rating
summary(train_all$Rating)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00   14.00   32.00   36.37   50.00  100.00 
```

Hide

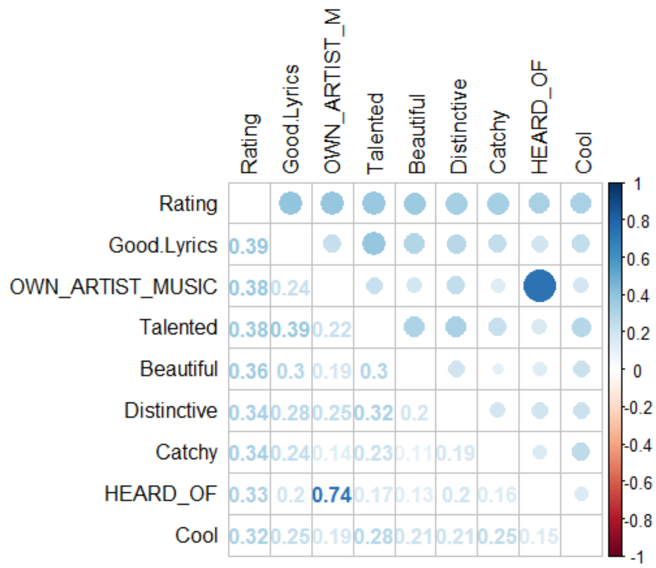
```
# Plot Rating
ggplot(data=train_all[!is.na(train_all$Rating),], aes(x=Rating)) +
  geom_histogram(fill="lightblue") + scale_x_continuous(breaks = seq(0,100, by=10))
```



2.3 Explore Correlation between Variables

Hide

```
# Variable Correlation
# Get all the numeric variables (this also includes ordinal variables since we encode them as integers)
numeric_var <- train_all[,which(!sapply(train_all, is.factor))]
# save name vector for later use
numeric_names <- colnames(numeric_var)
# get the pairwise correlation for all variables
cor <- cor(numeric_var[4:ncol(numeric_var)], use="pairwise.complete.obs")
# sort on decreasing correlation with Rating
cor_sorted <- as.matrix(sort(cor[,'Rating'], decreasing = TRUE))
# Select only high correlations
cor_high <- names(which(apply(cor_sorted, 1, function(x) abs(x)>0.32)))

# plot the correlation
cor_mix <- cor[cor_high,cor_high]
corrplot.mixed(cor_mix, tl.col="black", tl.pos = "lt")
```

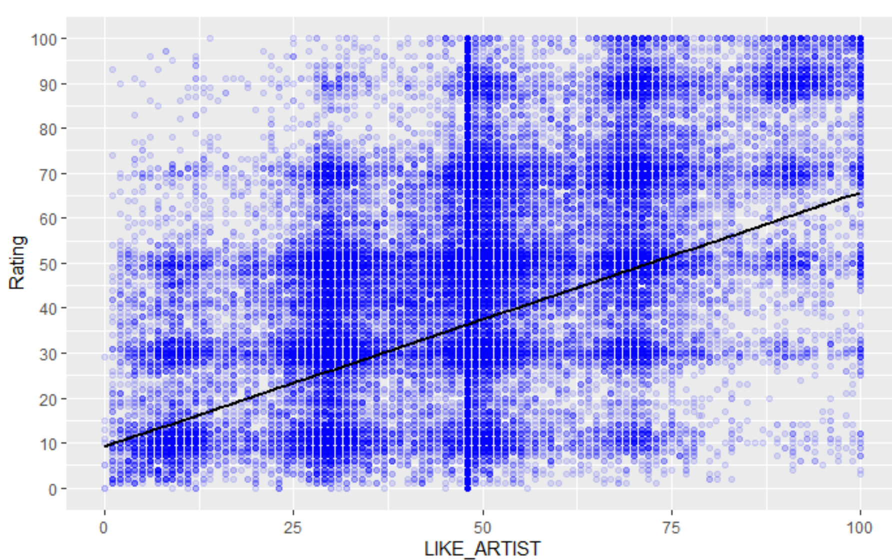


2.4 Dig Deeper into the Relationship Between Rating and Other Variables

2.4.1 Rating and LIKE_ARTIST

Hide

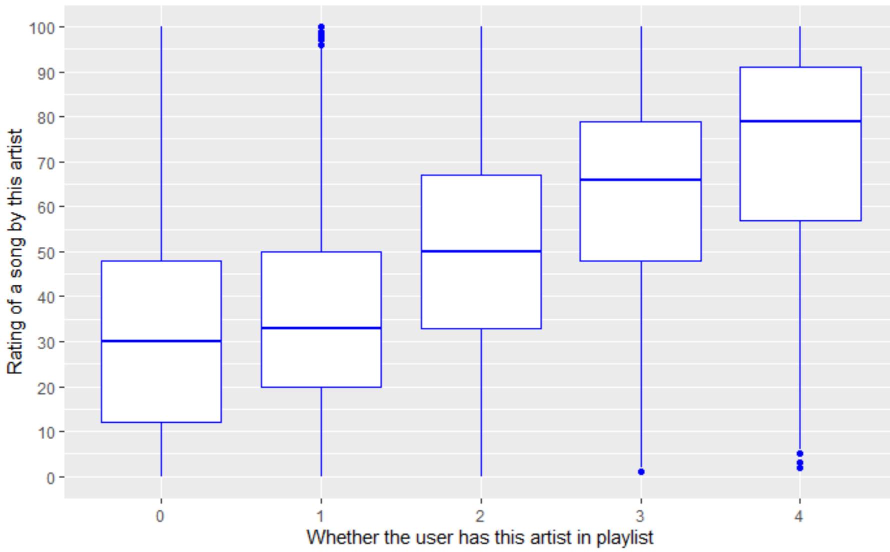
```
# Plot rating and LIKE_ARTIST
ggplot(data=train_all[!is.na(train_all$Rating),], aes(x=LIKE_ARTIST, y=Rating))+
        geom_point(col='blue', alpha = 0.1) + geom_smooth(method = "lm", se=FALSE, color="black", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 100, by=10))
```



2.4.2 Rating and OWN_ARTIST_MUSIC

Hide

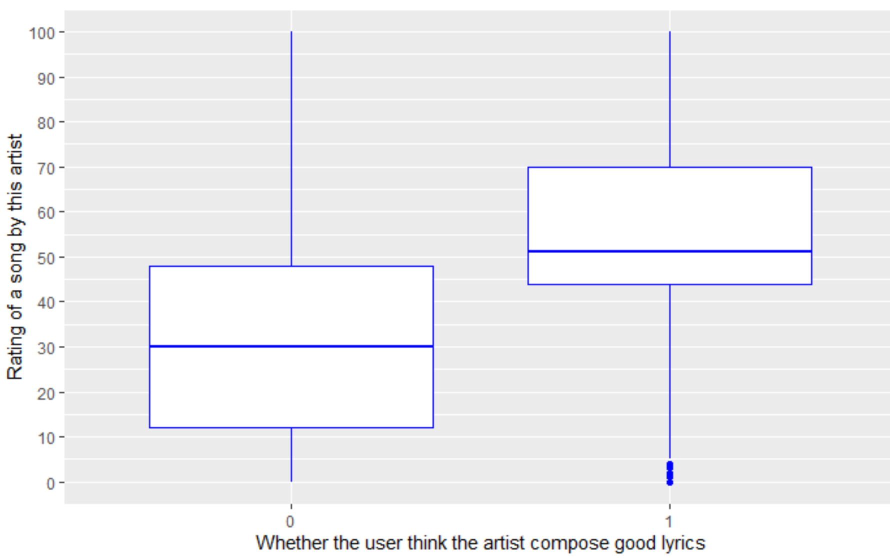
```
ggplot(data=train_all[!is.na(train_all$Rating),], aes(x=factor(OWN_ARTIST_MUSIC), y=Rating))+
        geom_boxplot(col='blue') + labs(x='Whether the user has this artist in playlist', y = "Rating of a song b
y this artist") +
        scale_y_continuous(breaks= seq(0, 100, by=10))
```



2.4.3 Rating and Good.Lyrics

Hide

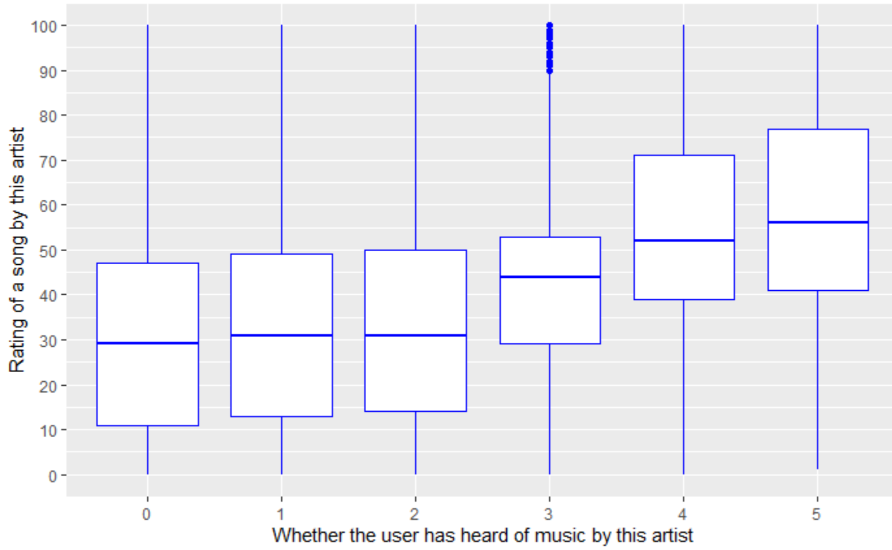
```
ggplot(data=train_all[!is.na(train_all$Rating),], aes(x=factor(Good.Lyrics), y=Rating))+
        geom_boxplot(col='blue') + labs(x='Whether the user think the artist compose good lyrics', y = "Rating of
a song by this artist") + scale_y_continuous(breaks= seq(0, 100, by=10))
```



2.4.4 Rating and HEARD_OF

Hide

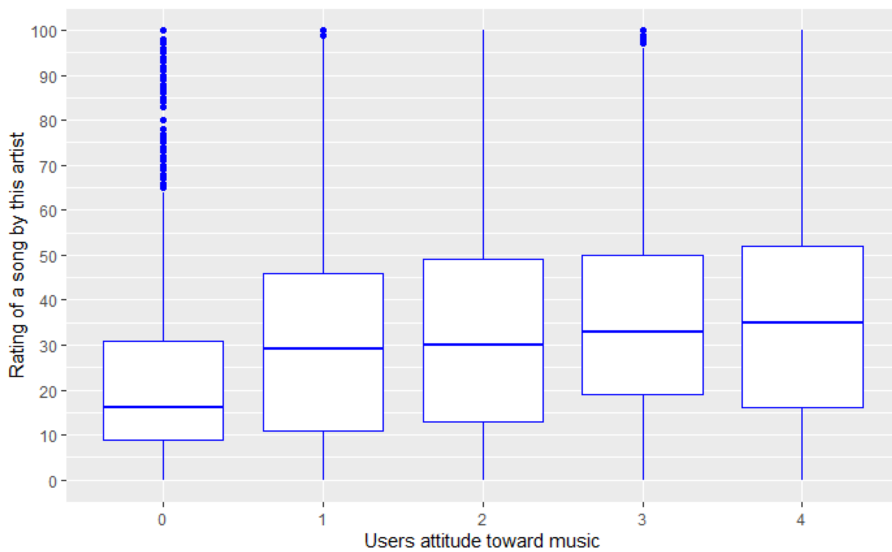
```
ggplot(data=train_all[!is.na(train_all$Rating),], aes(x=factor(HEARD_OF), y=Rating))+
        geom_boxplot(col='blue') + labs(x='Whether the user has heard of music by this artist', y = "Rating of a
 song by this artist") + scale_y_continuous(breaks= seq(0, 100, by=10))
```



2.4.5 Rating and Attitude toward music

Hide

```
ggplot(data=train_all[!is.na(train_all$Rating),], aes(x=factor(MUSIC), y=Rating))+
        geom_boxplot(col='blue') + labs(x='Users attitude toward music', y = "Rating of a song by this artist") +
scale_y_continuous(breaks= seq(0, 100, by=10))
```



3. Prepare the Data for Classification

3.1 Convert objective to Binary

Hide

```
# determine the cut_off for music recommendation
cut_off <- 50

# add a new binary column: 0 means not to recommend, 1 means recommend
train_all$Recommend <- ifelse(train_all$Rating>=cut_off, 1 ,0)
# Move the new objective column to front
train_all <- train_all %>% select(User, Artist, Track, Rating, Recommend, everything())

# Examine the distribution of the binary objective, 'Recommend'
per_zero <- train_all %>% select(Recommend) %>% summarise(sum(Recommend==0)/nrow(train_all))
per_one <- train_all %>% select(Recommend) %>% summarise(sum(Recommend==1)/nrow(train_all))
print(paste("The percentage of Don't Recommend is", per_zero, ", and the percentage of Recommend is", per_one, ".
The distribution is right skewed."))
```

```
[1] "The percentage of Don't Recommend is 0.724417693201128 , and the percentage of Recommend is 0.27558230679887
2 . The distribution is right skewed."
```

3.2 Split the dataset to training and testing

Hide

```
# Split dataset
n <- nrow(train_all)
indices <- sort(sample(1:n, round(0.8 * n)))
train <- train_all[indices,]     # training part
test <- train_all[-indices,]      # cv part


train_form <- as.formula(paste('Recommend ~',
                               paste(names(train)[6:ncol(train)], collapse=' + ')))
```

4. Fit with Different Classification Models

4.1 Logistic Regression

Hide

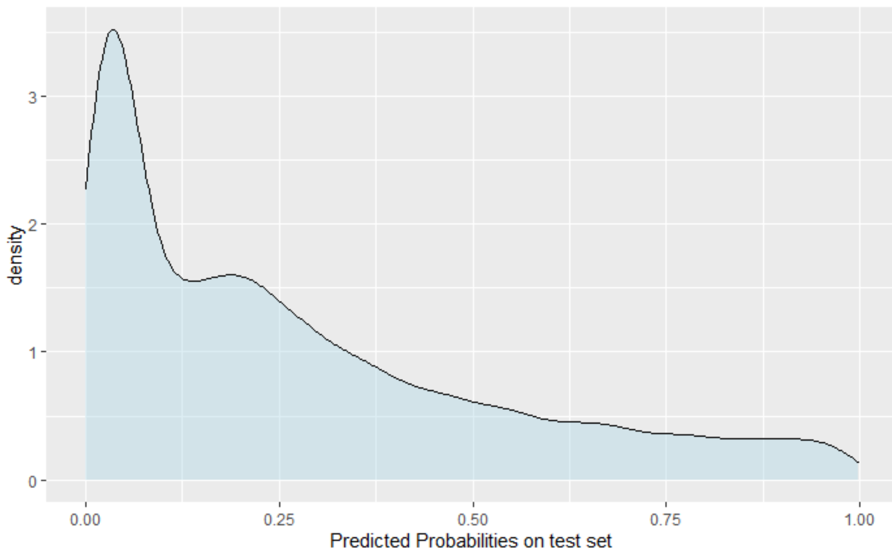
```
summary(model_glm)
```

```
Call:
glm(formula = train_form, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2830  -0.6770  -0.3166   0.4579   4.0023

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)      -3.744e+00  7.333e-02 -51.057  < 2e-16 ***
Time              3.073e-03  2.124e-03   1.446 0.148097
GENDER1          -2.429e-02  1.611e-02  -1.508 0.131540
AGE              -1.076e-04  6.715e-04  -0.160 0.872738
WORKING1         -2.487e-02  2.832e-02  -0.878 0.379696
WORKING2          4.880e-02  8.105e-02   0.602 0.547141
WORKING3         -3.676e-02  3.484e-02  -1.055 0.291400
WORKING4          2.114e-03  2.882e-02   0.073 0.941510
WORKING5         -1.119e-01  8.749e-02  -1.279 0.200767
WORKING6          3.576e-02  4.491e-02   0.796 0.425953
WORKING7          1.121e-01  1.072e-01   1.046 0.295663
WORKING8          1.242e-01  1.074e-01   1.157 0.247468
WORKING9          2.445e-02  3.590e-02   0.681 0.495823
WORKING10         8.776e-02  8.971e-02   0.978 0.327948
WORKING11         2.831e-02  3.885e-02   0.729 0.466187
WORKING12        -1.130e-01  4.004e-02  -2.821 0.004791 **
WORKING13         4.193e-02  3.987e-02   1.052 0.292964
REGION4           5.714e-02  1.816e-02   3.146 0.001654 **
REGION5          -1.319e-02  1.997e-02  -0.660 0.509010
REGION6           3.202e-01  3.887e-02   8.238  < 2e-16 ***
REGION7          -3.824e-02  5.539e-02  -0.690 0.489904
MUSIC             5.807e-02  1.135e-02   5.117 3.10e-07 ***
LIST_OWN          1.425e-02  3.246e-03   4.391 1.13e-05 ***
LIST_BACK        -9.653e-03  4.634e-03  -2.083 0.037219 *
HEARD_OF          1.006e-01  7.188e-03  13.998  < 2e-16 ***
OWN_ARTIST_MUSIC  1.538e-01  1.209e-02  12.722  < 2e-16 ***
LIKE_ARTIST       1.969e-02  6.903e-04  28.520  < 2e-16 ***
Q1                9.991e-04  4.815e-04   2.075 0.037985 *
Q2                2.415e-03  4.488e-04   5.382 7.39e-08 ***
Q3               -1.015e-03  5.346e-04  -1.899 0.057601 .
Q4                1.956e-03  4.024e-04   4.860 1.17e-06 ***
Q5                1.537e-03  4.081e-04   3.766 0.000166 ***
Q6                8.624e-04  3.174e-04   2.717 0.006584 **
Q7                1.403e-04  4.203e-04   0.334 0.738476
Q8                1.658e-03  4.492e-04   3.692 0.000223 ***
Q9                7.313e-04  3.155e-04   2.318 0.020453 *
Q10              -1.236e-04  3.943e-04  -0.314 0.753878
Q11               3.571e-03  5.181e-04   6.892 5.50e-12 ***
Q12               9.372e-04  4.824e-04   1.943 0.052070 .
Q13               8.196e-04  4.474e-04   1.832 0.067004 .
Q14              -4.521e-04  4.749e-04  -0.952 0.341086
Q15              -9.352e-05  4.531e-04  -0.206 0.836484
Q16               2.327e-03  4.148e-04   5.610 2.02e-08 ***
Q17               2.529e-03  3.875e-04   6.525 6.79e-11 ***
Q18               1.528e-03  6.156e-04   2.482 0.013076 *
Q19              -1.288e-03  5.845e-04  -2.203 0.027589 *
Uninspired       -5.074e-01  1.425e-01  -3.560 0.000371 ***
Sophisticated     1.435e-01  6.567e-02   2.185 0.028915 *
Aggressive        4.741e-02  4.739e-02   1.000 0.317109
Edgy              6.832e-02  2.389e-02   2.859 0.004249 **
Sociable         -4.001e-02  6.123e-02  -0.654 0.513406
Laid.back        -2.364e-01  4.452e-02  -5.310 1.10e-07 ***
Uplifting         1.009e-01  5.365e-02   1.881 0.059907 .
Intriguing        2.754e-02  5.299e-02   0.520 0.603237
Free             -2.363e-02  3.606e-02  -0.655 0.512323
Thoughtful        2.377e-02  2.411e-02   0.986 0.324273
Outspoken        -1.099e-02  7.228e-02  -0.152 0.879160
Serious          -9.681e-02  3.706e-02  -2.612 0.008992 **
Unattractive     -1.024e+00  8.494e-02 -12.056  < 2e-16 ***
Confident         1.269e-01  2.214e-02   5.733 9.87e-09 ***
Youthful         -1.908e-01  2.461e-02  -7.752 9.02e-15 ***
Boring           -1.701e+00  5.022e-02 -33.861  < 2e-16 ***
Current          -5.399e-02  2.104e-02  -2.567 0.010267 *
Colourful         9.352e-03  4.826e-02   0.194 0.846354
Stylish           1.931e-01  2.388e-02   8.086 6.15e-16 ***
Cheap            -3.590e-01  8.459e-02  -4.244 2.20e-05 ***
Irrelevant        1.406e-01  1.590e-01   0.884 0.376556
Heartfelt        -9.904e-02  4.991e-02  -1.984 0.047220 *
Calm             -6.726e-02  2.302e-02  -2.922 0.003483 **
Outgoing          2.365e-02  3.578e-02   0.661 0.508591
Inspiring         3.177e-01  3.109e-02  10.219  < 2e-16 ***
Beautiful         8.002e-01  2.315e-02  34.559  < 2e-16 ***
```

```
Fun              2.695e-01  2.351e-02  11.464  < 2e-16 ***
Authentic        1.685e-01  2.101e-02   8.020 1.06e-15 ***
Credible         5.592e-02  2.370e-02   2.359 0.018321 *
Way.out         -4.172e-01  8.141e-02  -5.125 2.98e-07 ***
Cool             2.889e-01  2.007e-02  14.394  < 2e-16 ***
Catchy           5.016e-01  1.837e-02  27.309  < 2e-16 ***
Sensitive        5.669e-02  3.026e-02   1.873 0.061056 .
Mainstream      -8.478e-02  5.377e-02  -1.577 0.114855
Superficial     -4.597e-01  7.166e-02  -6.415 1.41e-10 ***
Annoying        -1.478e+00  1.038e-01 -14.239  < 2e-16 ***
Passionate       8.974e-02  2.561e-02   3.504 0.000458 ***
Not.authentic    1.227e-02  1.709e-01   0.072 0.942745
Good.Lyrics      4.266e-01  1.908e-02  22.355  < 2e-16 ***
Background      -3.792e-01  5.965e-02  -6.357 2.05e-10 ***
Timeless         4.272e-01  2.444e-02  17.479  < 2e-16 ***
Depressing      -8.717e-01  5.233e-02 -16.660  < 2e-16 ***
Original         1.201e-01  1.982e-02   6.057 1.39e-09 ***
Talented         3.311e-01  1.988e-02  16.654  < 2e-16 ***
Distinctive      2.711e-01  1.810e-02  14.978  < 2e-16 ***
Approachable     1.027e-01  2.388e-02   4.301 1.70e-05 ***
Genius          -1.023e-01  6.547e-02  -1.563 0.117982
Trendsetter     -6.798e-03  3.654e-02  -0.186 0.852405
Noisy           -6.642e-01  4.803e-02 -13.830  < 2e-16 ***
Upbeat           7.407e-02  2.427e-02   3.052 0.002272 **
Relatable        1.586e-01  6.767e-02   2.343 0.019113 *
Energetic        1.916e-01  2.206e-02   8.686  < 2e-16 ***
Exciting         2.753e-01  5.403e-02   5.095 3.49e-07 ***
Emotional        7.788e-02  4.333e-02   1.797 0.072267 .
None.of.these   -1.339e+00  4.580e-02 -29.238  < 2e-16 ***
Sexy             1.085e-01  3.834e-02   2.830 0.004662 **
Over            -1.521e-01  7.839e-02  -1.941 0.052306 .
Rebellious      -3.859e-01  5.993e-02  -6.439 1.20e-10 ***
Fake            -1.273e-01  8.623e-02  -1.476 0.139963
Cheesy          -4.240e-01  5.189e-02  -8.170 3.07e-16 ***
Popular          2.275e-01  4.747e-02   4.792 1.65e-06 ***
Superstar        8.010e-02  8.943e-02   0.896 0.370374
Relaxed         -1.370e-02  5.149e-02  -0.266 0.790216
Intrusive        3.852e-01  1.677e-01   2.297 0.021601 *
Unoriginal      -6.227e-01  5.229e-02 -11.909  < 2e-16 ***
Dated           -7.130e-01  4.085e-02 -17.456  < 2e-16 ***
Unapproachable  -2.449e-01  1.142e-01  -2.145 0.031937 *
Classic          1.293e-01  2.418e-02   5.348 8.88e-08 ***
Playful          8.778e-02  3.317e-02   2.646 0.008143 **
Arrogant         2.235e-01  5.773e-02   3.872 0.000108 ***
Warm             1.839e-01  2.428e-02   7.573 3.65e-14 ***
Soulful          1.725e-01  6.220e-02   2.773 0.005547 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 164978  on 139822  degrees of freedom
Residual deviance: 115310  on 139705  degrees of freedom
AIC: 115546

Number of Fisher Scoring iterations: 6
```

Hide

```
test$prob_glm <- predict(model_glm, test, type="response")

# Density of probabilities
ggplot(data.frame(test) , aes(prob_glm)) +
        geom_density(fill = 'lightblue' , alpha = 0.4) +
        labs(x = 'Predicted Probabilities on test set')
```



Hide

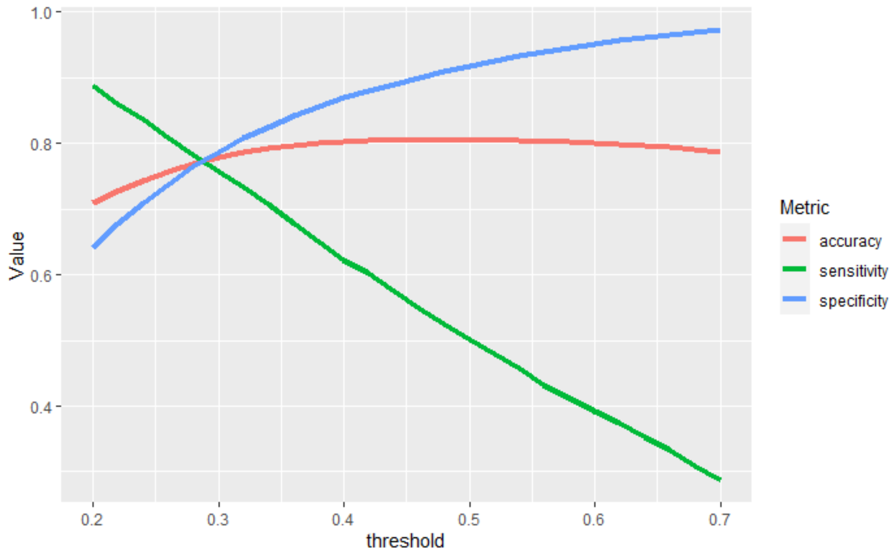
```
# Find optimum threshold
k = 0
accuracy = c()
sensitivity = c()
specificity = c()
for(i in seq(from = 0.2 , to = 0.7 , by = 0.02)){
        k = k + 1
        preds_binomial_glm = ifelse(test$prob_glm > i , 1 , 0)
        confmat = table(test$Recommend , preds_binomial_glm)
        accuracy[k] = sum(diag(confmat)) / sum(confmat)
        sensitivity[k] = confmat[2 , 2] / sum(confmat[2, ])
        specificity[k] = confmat[1 , 1] / sum(confmat[1, ])
}

# Put the result all into a dataframe
threshold = seq(from = 0.2 , to = 0.7 , by = 0.02)
data = data.frame(threshold , accuracy , sensitivity , specificity)
data
```

threshold <dbl>	accuracy <dbl>	sensitivity <dbl>	specificity <dbl>
0.20	0.7084907	0.8866328	0.6424314
0.22	0.7261414	0.8595601	0.6766667
0.24	0.7430198	0.8358714	0.7085882
0.26	0.7567513	0.8087986	0.7374510
0.28	0.7686520	0.7816201	0.7638431
0.30	0.7785502	0.7570854	0.7865098
0.32	0.7871610	0.7327623	0.8073333
0.34	0.7927394	0.7063240	0.8247843
0.36	0.7968589	0.6771362	0.8412549
0.38	0.7996338	0.6494289	0.8553333

1-10 of 26 rows Previous **1** 2 3 Next

Hide

```
# Gather accuracy , sensitivity and specificity in one column
ggplot(gather(data , key = 'Metric' , value = 'Value' , 2:4) ,
       aes(x = threshold , y = Value , color = Metric)) +
        geom_line(size = 1.5)
```



Hide

```
# Get the confusion matrix at a cut-off of 0.5
print("The confusion matrix at the default cut-off of 0.5:")
```

```
[1] "The confusion matrix at the default cut-off of 0.5:"
```

Hide

```
preds_binomial_glm_cut = ifelse(test$prob_glm > 0.5 , 1 , 0)
confmat = table(test$Recommend , preds_binomial_glm_cut)
confmat
```

```
   preds_binomial_glm_cut
        0     1
  0 23398  2102
  1  4725  4731
```

Hide

```
# print out the evaluation data at the cut-off of 0.5
data[16,]
```

	threshold <dbl>	accuracy <dbl>	sensitivity <dbl>	specificity <dbl>
16	0.5	0.8046973	0.5003173	0.9175686

1 row

Hide

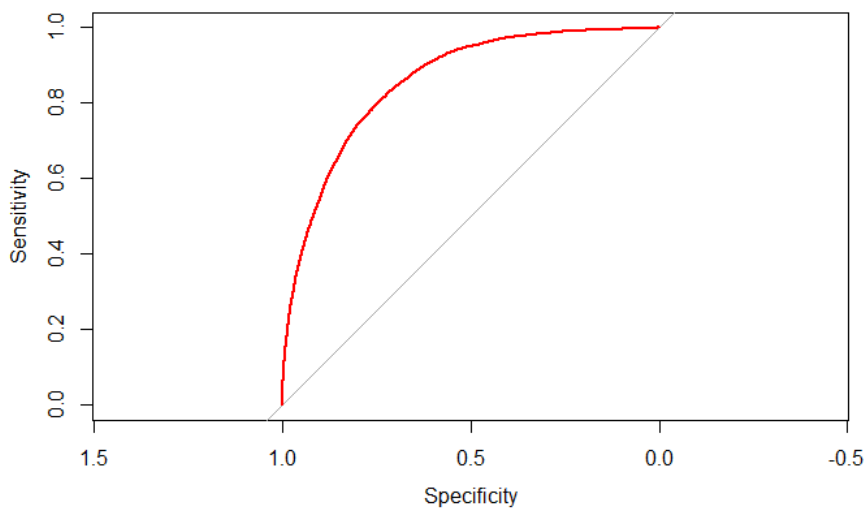
```
# update threshold
# test$recommend_glm <- ifelse(test$prob_glm > 0.5 , 1 , 0)

# Examine the distribution of the predicted objective variable, 'gbm_recommend'
per_zero_glm <- test %>% select(recommend_glm) %>% summarise(sum(recommend_glm==0)/nrow(test))
per_one_glm <- test %>% select(recommend_glm) %>% summarise(sum(recommend_glm==1)/nrow(test))
print(paste("The percentage of Don't Recommend by the gbm model is", per_zero_glm, "comparing to the actual perce
ntage of", per_zero, "; The percentage of Recommend by the gbm model is", per_one_glm, "comparing to the actual p
ercentage of Recommend of", per_one))
```

```
[1] "The percentage of Don't Recommend by the gbm model is 0.736125414807186 comparing to the actual percentage o
f 0.724417693201128 ; The percentage of Recommend by the gbm model is 0.263874585192814 comparing to the actual p
ercentage of Recommend of 0.275582306798872"
```

Hide

```
# report AUC
auc_rf = roc(test$Recommend, test$prob_glm, plot = TRUE, col = "red")
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```



Hide

```
print(auc_rf)
```

```

Call:
roc.default(response = test$Recommend, predictor = test$prob_glm,     plot = TRUE, col = "red")

Data: test$prob_glm in 25500 controls (test$Recommend 0) < 9456 cases (test$Recommend 1).
Area under the curve: 0.8556
```

4.2 Random Forest

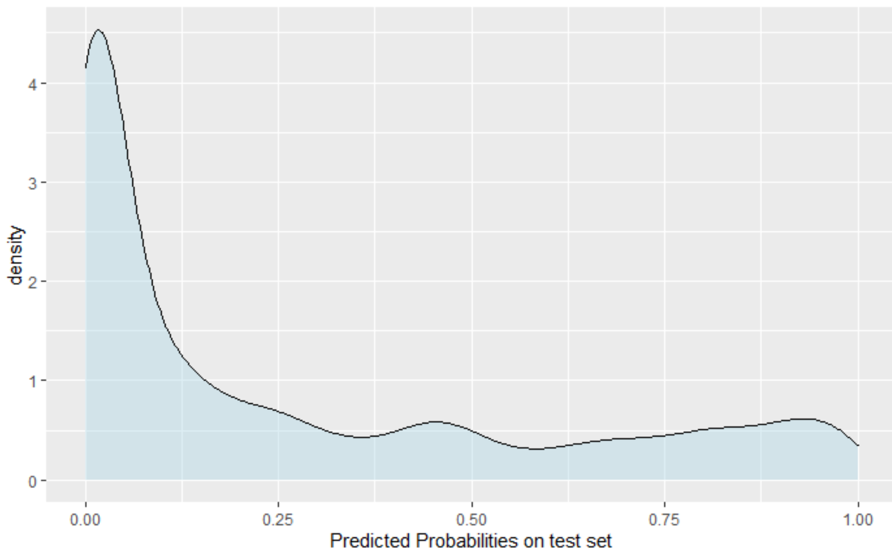
Hide

```

summary(model_rf)
```

```
                Length Class  Mode
call                 6 -none- call
type                 1 -none- character
predicted       139823 factor numeric
err.rate          1350 -none- numeric
confusion            6 -none- numeric
votes           279646 matrix numeric
oob.times       139823 -none- numeric
classes              2 -none- character
importance         102 -none- numeric
importanceSD         0 -none- NULL
localImportance      0 -none- NULL
proximity            0 -none- NULL
ntree                1 -none- numeric
mtry                 1 -none- numeric
forest              14 -none- list
y               139823 factor numeric
test                 0 -none- NULL
inbag                0 -none- NULL
terms                3 terms  call
```

Hide

```
predict_rf <- predict(model_rf, test, type='prob')
test$prob_rf <- predict_rf[,"1"]

#sum(test$prob_rf==test$Recommend)/nrow(test)
# Density of probabilities
ggplot(data=test , aes(prob_rf)) +
        geom_density(fill = 'lightblue' , alpha = 0.4) +
        labs(x = 'Predicted Probabilities on test set')
```



Hide

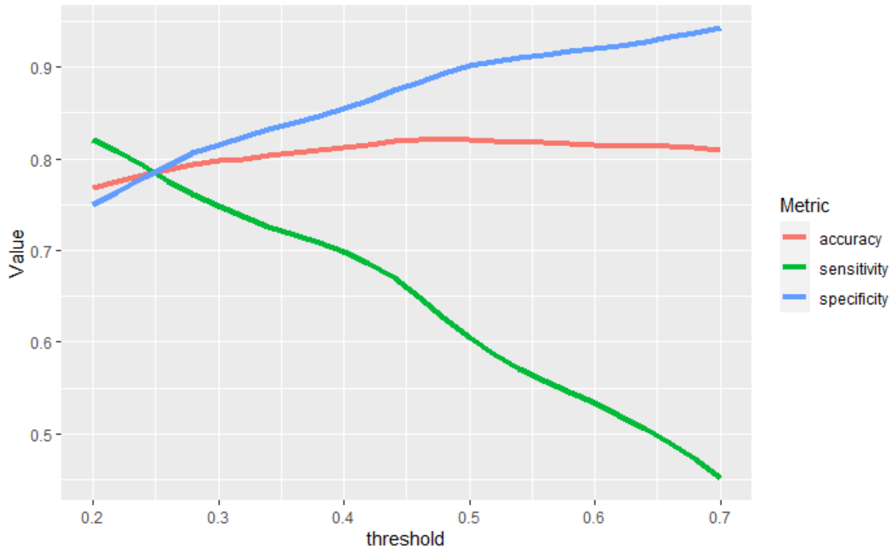
```
# Find optimum threshold
k = 0
accuracy = c()
sensitivity = c()
specificity = c()
for(i in seq(from = 0.2 , to = 0.7 , by = 0.02)){
        k = k + 1
        preds_binomial_rf = ifelse(test$prob_rf > i , 1 , 0)
        confmat = table(test$Recommend , preds_binomial_rf)
        accuracy[k] = sum(diag(confmat)) / sum(confmat)
        sensitivity[k] = confmat[2 , 2] / sum(confmat[2, ])
        specificity[k] = confmat[1 , 1] / sum(confmat[1, ])
}

# Put the result all into a dataframe
threshold = seq(from = 0.2 , to = 0.7 , by = 0.02)
data = data.frame(threshold , accuracy , sensitivity , specificity)
data
```

threshold <dbl>	accuracy <dbl>	sensitivity <dbl>	specificity <dbl>
0.20	0.7686806	0.8207487	0.7493725
0.22	0.7757753	0.8079526	0.7638431
0.24	0.7828413	0.7925127	0.7792549
0.26	0.7883339	0.7756980	0.7930196
0.28	0.7938265	0.7605753	0.8061569
0.30	0.7972880	0.7479907	0.8155686
0.32	0.8000915	0.7369924	0.8234902
0.34	0.8031811	0.7253596	0.8320392
0.36	0.8062421	0.7168993	0.8393725
0.38	0.8091029	0.7088621	0.8462745

1-10 of 26 rows Previous **1** 2 3 Next

Hide

```
# Gather accuracy , sensitivity and specificity in one column
ggplot(gather(data , key = 'Metric' , value = 'Value' , 2:4) ,
       aes(x = threshold , y = Value , color = Metric)) +
        geom_line(size = 1.5)
```



Hide

```
# Get the confusion matrix at a cut-off of 0.5
print("The confusion matrix at the default cut-off of 0.5:")
```

```
[1] "The confusion matrix at the default cut-off of 0.5:"
```

Hide

```
preds_binomial_rf_cut = round(test$prob_rf)
confmat = table(test$Recommend , preds_binomial_rf_cut)
confmat
```

```
   preds_binomial_rf_cut
        0     1
  0 22979  2521
  1  3743  5713
```

Hide

```
# print out the evaluation data at the cut-off of 0.5
data[16,]
```

	threshold <dbl>	accuracy <dbl>	sensitivity <dbl>	specificity <dbl>
16	0.5	0.8208033	0.6041667	0.9011373

1 row

Hide

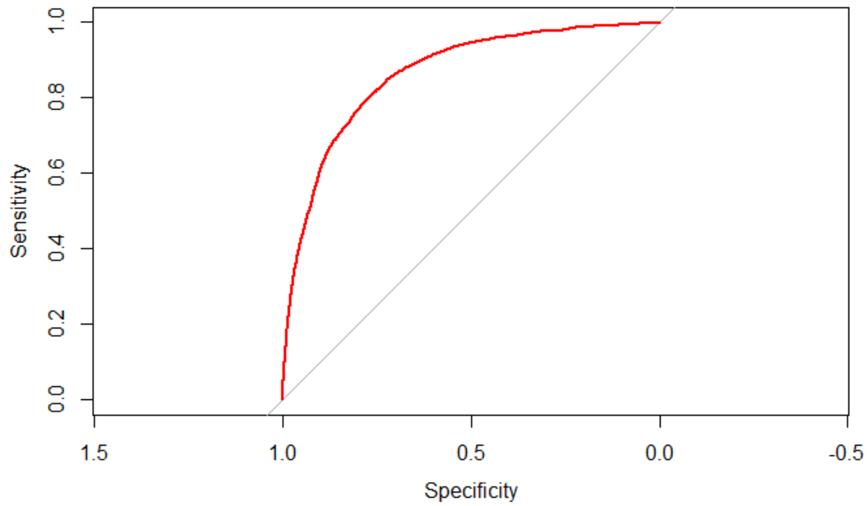
```
# update threshold
# test$recommend_rf <- ifelse(test$prob_rf > 0.5 , 1 , 0)

# Examine the distribution of the predicted objective variable, 'gbm_recommend'
per_zero_rf <- test %>% select(recommend_rf) %>% summarise(sum(recommend_rf==0)/nrow(test))
per_one_rf <- test %>% select(recommend_rf) %>% summarise(sum(recommend_rf==1)/nrow(test))
print(paste("The percentage of Don't Recommend by the gbm model is", per_zero_rf, "comparing to the actual percen
tage of", per_zero, "; The percentage of Recommend by the gbm model is", per_one_rf, "comparing to the actual per
centage of Recommend of", per_one))
```

```
[1] "The percentage of Don't Recommend by the gbm model is 0.704657283442041 comparing to the actual percentage o
f 0.724417693201128 ; The percentage of Recommend by the gbm model is 0.295342716557959 comparing to the actual p
ercentage of Recommend of 0.275582306798872"
```

Hide

```
# report AUC
auc_rf = roc(test$Recommend, predict_rf[,"1"], plot = TRUE, col = "red")
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```



Hide

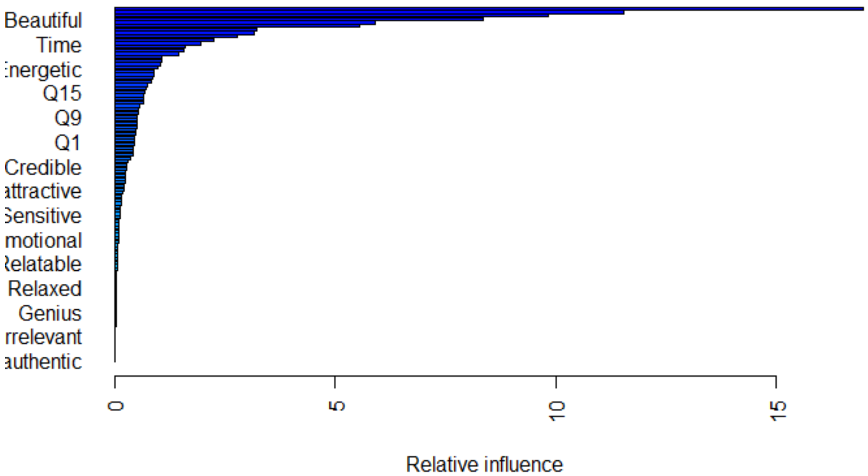
```
print(auc_rf)
```

```
Call:
roc.default(response = test$Recommend, predictor = predict_rf[,     "1"], plot = TRUE, col = "red")

Data: predict_rf[, "1"] in 25500 controls (test$Recommend 0) < 9456 cases (test$Recommend 1).
Area under the curve: 0.8633
```

Gradient Boosting Machine

	var <chr>	rel.inf <dbl>
Good.Lyrics	Good.Lyrics	16.970810518
LIKE_ARTIST	LIKE_ARTIST	11.548328329
Talented	Talented	9.818362323
Beautiful	Beautiful	8.354580816
Catchy	Catchy	5.908406996
OWN_ARTIST_MUSIC	OWN_ARTIST_MUSIC	5.559058194
Distinctive	Distinctive	3.218135008
Timeless	Timeless	3.156714790
Cool	Cool	2.784910951
Boring	Boring	2.237990377

1-10 of 102 rows Previous 1 2 3 4 5 6 ... 11 Next



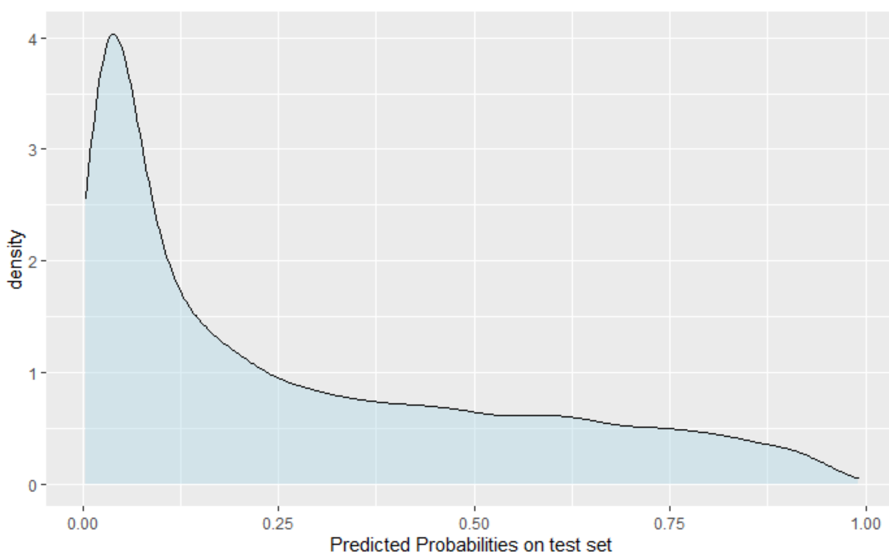
Hide

```
# reprot the result
summary(model_gbm, las=2)


# predict on cv set
test$prob_gbm <- predict(model_gbm, test, type='response')
```

```
Using 405 trees...
```

Hide

```
test$recommend_gbm <- round(test$prob_gbm)
test <- test %>% select(User, Artist, Track, Rating, Recommend, prob_gbm, recommend_gbm, everything())


# Density of probabilities
ggplot(data.frame(test) , aes(prob_gbm)) +
        geom_density(fill = 'lightblue' , alpha = 0.4) +
        labs(x = 'Predicted Probabilities on test set')
```



Hide

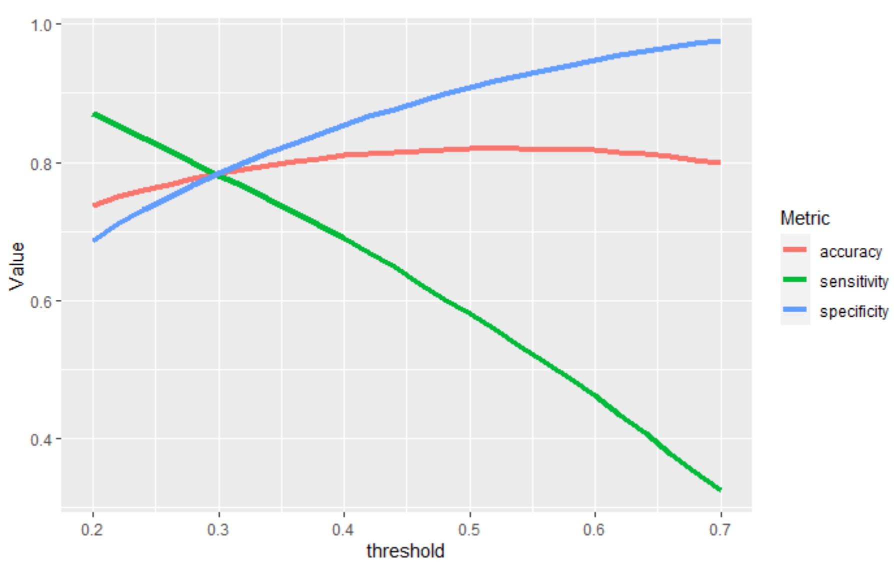
```
# Find optimum threshold
k = 0
accuracy = c()
sensitivity = c()
specificity = c()
for(i in seq(from = 0.2 , to = 0.7 , by = 0.02)){
        k = k + 1
        preds_binomial_gbm = ifelse(test$prob_gbm > i , 1 , 0)
        confmat = table(test$Recommend , preds_binomial_gbm)
        accuracy[k] = sum(diag(confmat)) / sum(confmat)
        sensitivity[k] = confmat[2 , 2] / sum(confmat[2, ])
        specificity[k] = confmat[1 , 1] / sum(confmat[1, ])
}

# Put the result all into a dataframe
threshold = seq(from = 0.2 , to = 0.7 , by = 0.02)
data = data.frame(threshold , accuracy , sensitivity , specificity)
data
```

threshold <dbl>	accuracy <dbl>	sensitivity <dbl>	specificity <dbl>
0.20	0.7366403	0.8714044	0.6866667
0.22	0.7494851	0.8534264	0.7109412
0.24	0.7592688	0.8353426	0.7310588
0.26	0.7680513	0.8182107	0.7494510
0.28	0.7764904	0.8001269	0.7677255
0.30	0.7835279	0.7810914	0.7844314
0.32	0.7899931	0.7650169	0.7992549
0.34	0.7962868	0.7468274	0.8146275
0.36	0.8013789	0.7284264	0.8284314
0.38	0.8056986	0.7098139	0.8412549

1-10 of 26 rows Previous **1** 2 3 Next

Hide

```
# Gather accuracy , sensitivity and specificity in one column
ggplot(gather(data , key = 'Metric' , value = 'Value' , 2:4) ,
       aes(x = threshold , y = Value , color = Metric)) +
        geom_line(size = 1.5)
```



Hide

```
# Get the confusion matrix at a cut-off of 0.5
print("The confusion matrix at the default cut-off of 0.5:")
```

```
[1] "The confusion matrix at the default cut-off of 0.5:"
```

Hide

```
preds_binomial_gbm_cut = ifelse(test$prob_gbm > 0.5 , 1 , 0)
confmat = table(test$Recommend , preds_binomial_gbm_cut)
confmat
```

```
   preds_binomial_gbm_cut
        0     1
  0 23157  2343
  1  3962  5494
```

Hide

```
# print out the evaluation data at the cut-off of 0.5
data[16,]
```

	threshold <dbl>	accuracy <dbl>	sensitivity <dbl>	specificity <dbl>
16	0.5	0.8196304	0.5810068	0.9081176

1 row

Hide

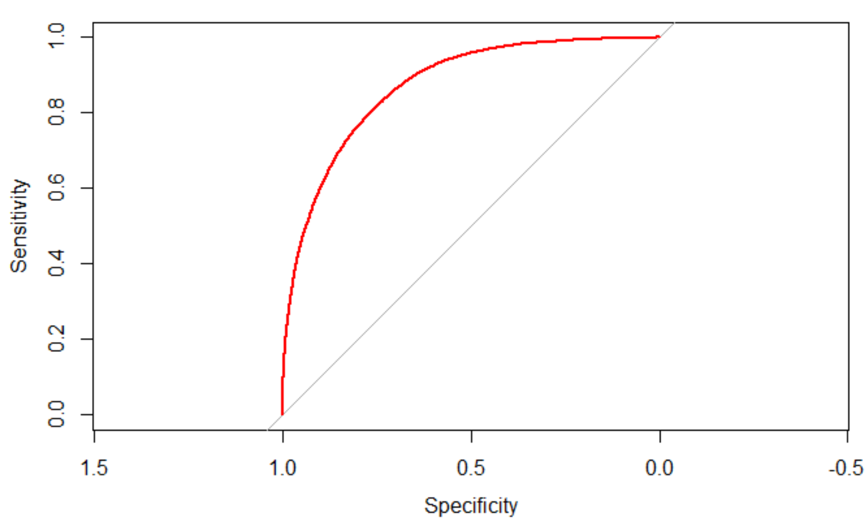
```
# update threshold
# test$recommend_gbm <- ifelse(test$prob_gbm > 0.5 , 1 , 0)

# Examine the distribution of the predicted objective variable, 'gbm_recommend'
per_zero_gbm <- test %>% select(recommend_gbm) %>% summarise(sum(recommend_gbm==0)/nrow(test))
per_one_gbm <- test %>% select(recommend_gbm) %>% summarise(sum(recommend_gbm==1)/nrow(test))
print(paste("The percentage of Don't Recommend by the gbm model is", per_zero_gbm, "comparing to the actual perce
ntage of", per_zero, "; The percentage of Recommend by the gbm model is", per_one_gbm, "comparing to the actual p
ercentage of Recommend of", per_one))
```

```
[1] "The percentage of Don't Recommend by the gbm model is 0.775803867719419 comparing to the actual percentage o
f 0.724417693201128 ; The percentage of Recommend by the gbm model is 0.224196132280581 comparing to the actual p
ercentage of Recommend of 0.275582306798872"
```

Hide

```
# report AUC
auc_gbm = roc(test$Recommend, test$prob_gbm, plot = TRUE, col = "red")
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```



Hide

```
print(auc_gbm)
```

```
Call:
roc.default(response = test$Recommend, predictor = test$prob_gbm,     plot = TRUE, col = "red")

Data: test$prob_gbm in 25500 controls (test$Recommend 0) < 9456 cases (test$Recommend 1).
Area under the curve: 0.8708
```

Coment on the evaluation method

Hide

```
print("Sensitivity: out of users who actually like the track, how many of them did we recommend? (predicted tru
e); Specificity: out of users who actually don't like the track, how many of them did we NOT recommend? (predict
 false). Not recommending a prefer song to a user won't cause much loss, but recommending songs to users who do
n't like can impact negatively on retention. Therefore we want relatively high SPECIFICTY, and we have relatively
high tolerance for low Sensitivity.")
```

```
[1] "Sensitivity: out of users who actually like the track, how many of them did we recommend? (predicted true);
Specificity: out of users who actually don't like the track, how many of them did we NOT recommend? (predict fals
e). Not recommending a prefer song to a user won't cause much loss, but recommending songs to users who don't lik
e can impact negatively on retention. Therefore we want relatively high SPECIFICTY, and we have relatively high t
olerance for low Sensitivity."
```

5 Output result file for futher analysis

Hide

```
model_results <- test %>% select(Rating, Recommend, prob_glm, prob_gbm, prob_rf)
write.csv(model_results, "model_results.csv", row.names=FALSE)
```