Spotify analysis

Sumeet k Singh

7/2/2020

```
##   X acousticness danceability duration_ms energy instrumentalness key liveness
## 1 0      0.01020        0.833      204600  0.434         0.021900   2   0.1650
## 2 1      0.19900        0.743      326933  0.359         0.006110   1   0.1370
## 3 2      0.03440        0.838      185707  0.412         0.000234   2   0.1590
## 4 3      0.60400        0.494      199413  0.338         0.510000   5   0.0922
## 5 4      0.18000        0.678      392893  0.561         0.512000   5   0.4390
## 6 5      0.00479        0.804      251333  0.560         0.000000   8   0.1640
##   loudness mode speechiness   tempo time_signature valence target
## 1   -8.795    1      0.4310 150.062              4   0.286      1
## 2  -10.401    1      0.0794 160.083              4   0.588      1
## 3   -7.148    1      0.2890  75.044              4   0.173      1
## 4  -15.236    1      0.0261  86.468              4   0.230      1
## 5  -11.648    0      0.0694 174.004              4   0.904      1
## 6   -6.682    1      0.1850  85.023              4   0.264      1
##       song_title           artist
## 1       Mask Off           Future
## 2        Redbone Childish Gambino
## 3   Xanny Family           Future
## 4 Master Of None      Beach House
## 5 Parallel Lines      Junior Boys
## 6     Sneakinâ\200\231            Drake
```

This is a dataset consisting of features for tracks fetched using Spotify's Web API. The tracks are labeled '1' or '0' ('Hit' or 'Flop') depending on some criteria of the author. This dataset can be used to make a classification model that predicts whether a track would be a 'Hit' or not.

```r
str(spotify)
```

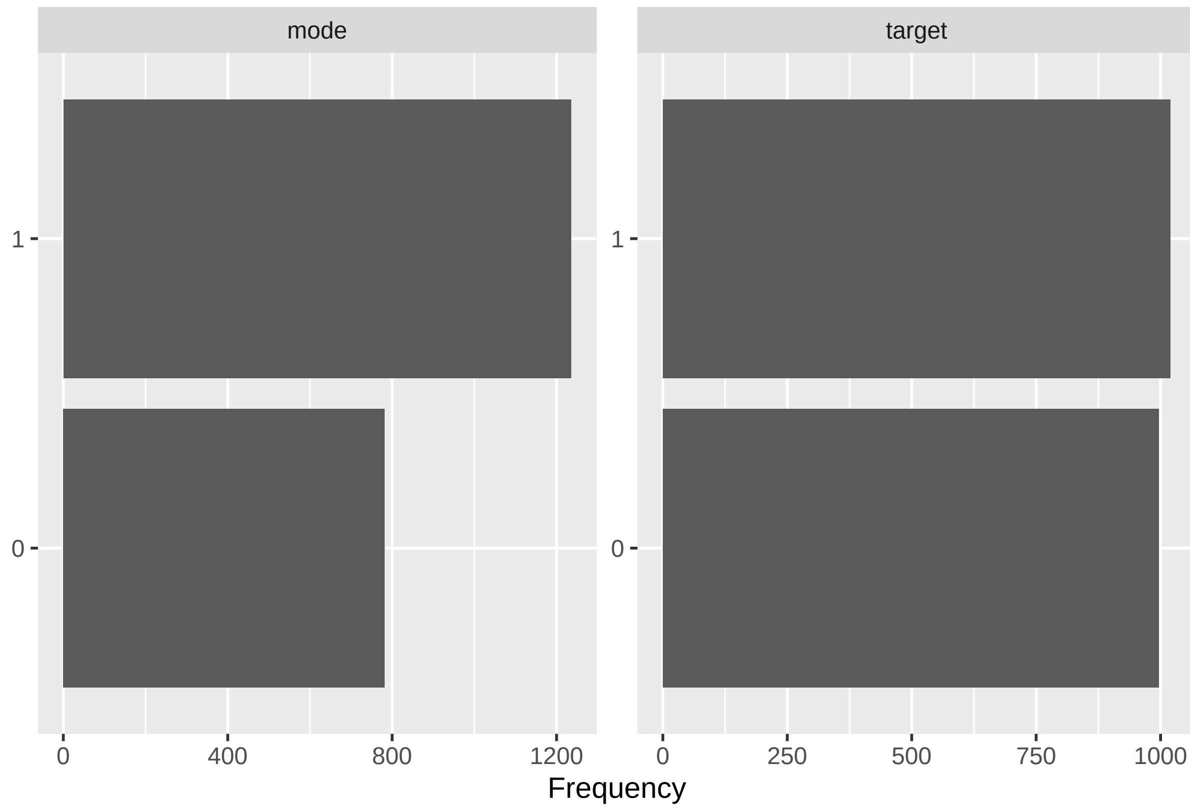
```
## 'data.frame':    2017 obs. of  17 variables:
##  $ X               : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ acousticness    : num  0.0102 0.199 0.0344 0.604 0.18 0.00479 0.0145 0.0202 0.0481 0.00208 ...
##  $ danceability    : num  0.833 0.743 0.838 0.494 0.678 0.804 0.739 0.266 0.603 0.836 ...
##  $ duration_ms     : int  204600 326933 185707 199413 392893 251333 241400 349667 202853 226840 ...
##  $ energy          : num  0.434 0.359 0.412 0.338 0.561 0.56 0.472 0.348 0.944 0.603 ...
##  $ instrumentalness: num  2.19e-02 6.11e-03 2.34e-04 5.10e-01 5.12e-01 0.00 7.27e-06 6.64e-01 0.00 0
##  $ key             : int  2 1 2 5 5 8 1 10 11 7 ...
##  $ liveness        : num  0.165 0.137 0.159 0.0922 0.439 0.164 0.207 0.16 0.342 0.571 ...
##  $ loudness        : num  -8.79 -10.4 -7.15 -15.24 -11.65 ...
##  $ mode            : int  1 1 1 1 0 1 1 0 0 1 ...
##  $ speechiness     : num  0.431 0.0794 0.289 0.0261 0.0694 0.185 0.156 0.0371 0.347 0.237 ...
##  $ tempo           : num  150.1 160.1 75 86.5 174 ...
```

```
##  $ time_signature  : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ valence         : num  0.286 0.588 0.173 0.23 0.904 0.264 0.308 0.393 0.398 0.386 ...
##  $ target          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ song_title      : chr  "Mask Off" "Redbone" "Xanny Family" "Master Of None" ...
##  $ artist          : chr  "Future" "Childish Gambino" "Future" "Beach House" ...
```

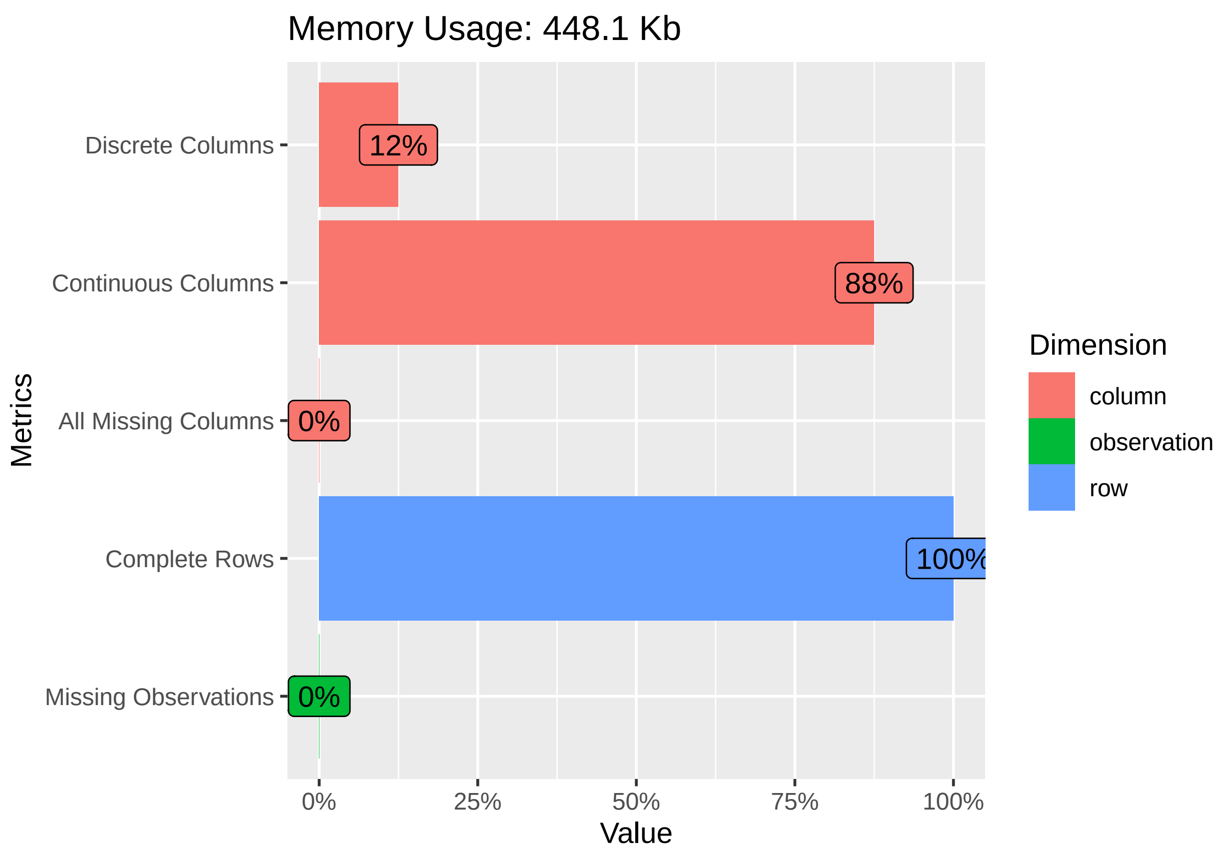
```r
#if (!require(devtools)) install.packages("devtools")
#library(devtools)
#install_github("boxuancui/DataExplorer")
```

```r
#DataExplorer::create_report(spotify_data1)


plot_bar(spotify_data1)
```

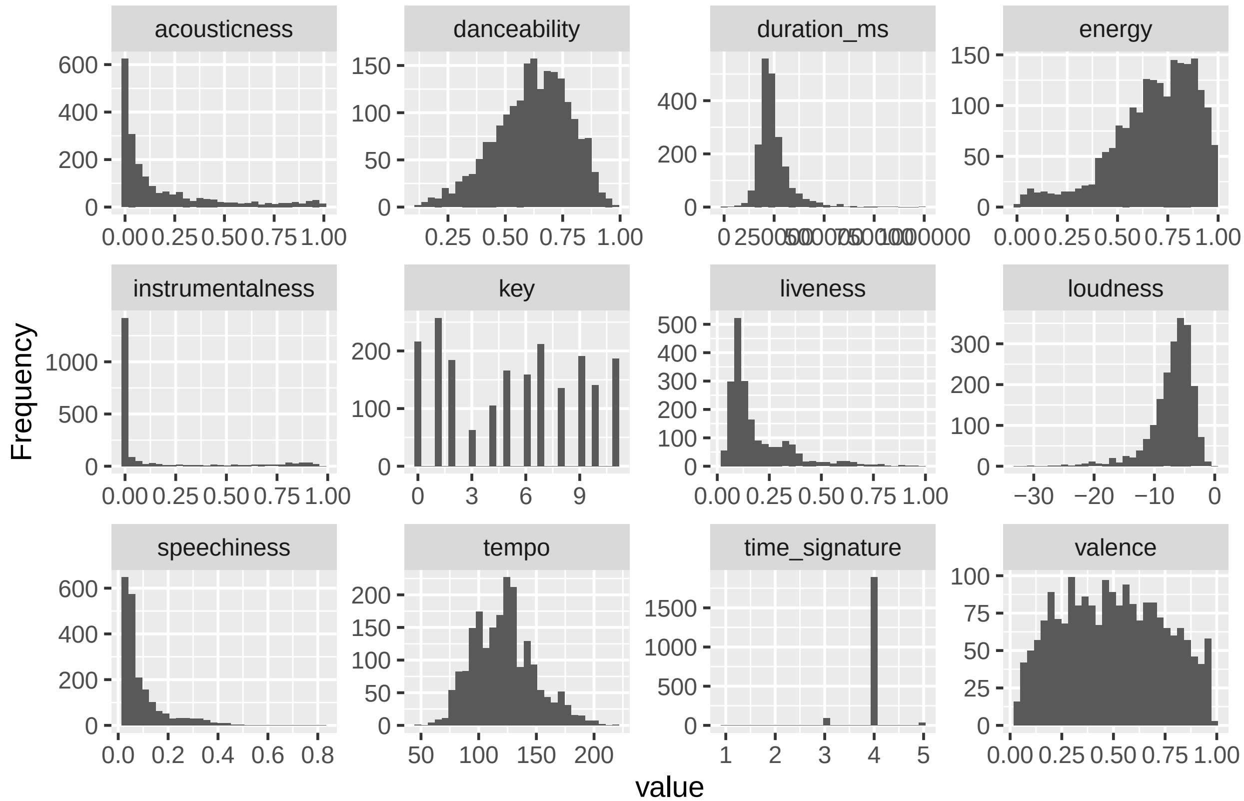
```
## 2 columns ignored with more than 50 categories.
## song_title: 1956 categories
## artist: 1343 categories
```




```r
plot_str(spotify_data1)

plot_intro(spotify_data1)
```

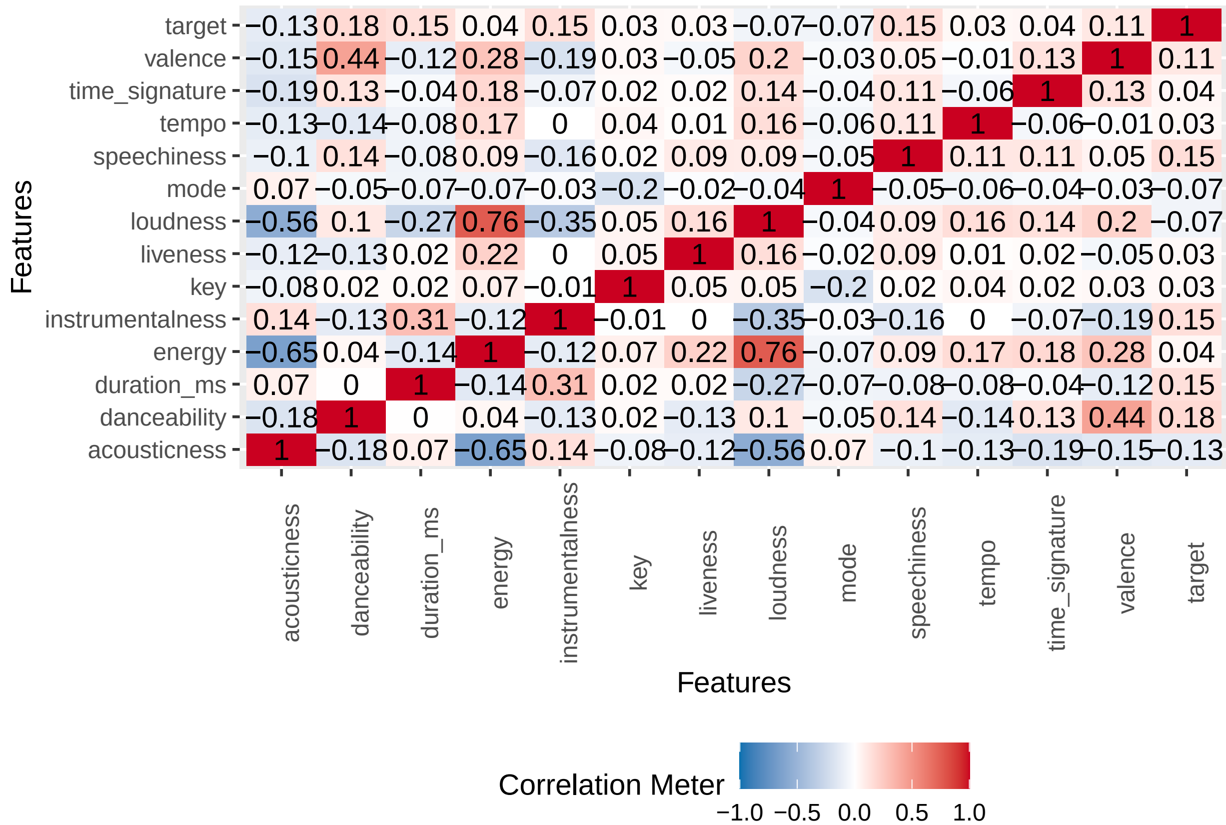



```r
plot_histogram(spotify_data1)
```

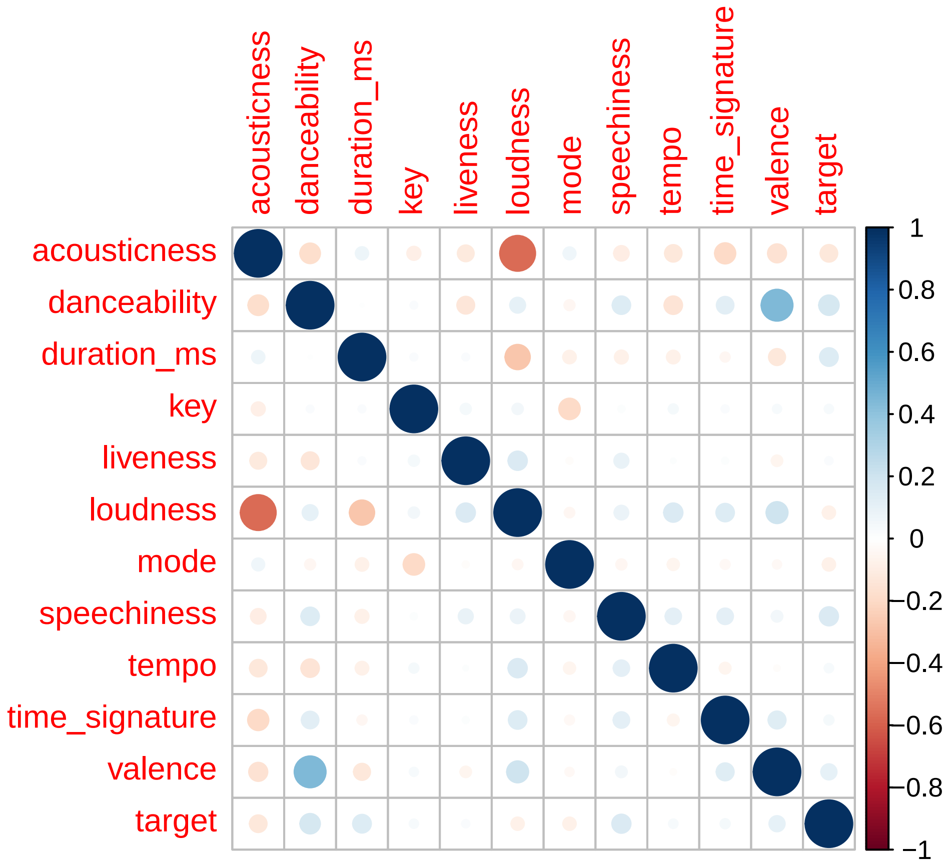



```r
plot_correlation(spotify_data1, maxcat = 5L)
```

```
## Warning in dummify(data, maxcat = maxcat): Ignored all discrete features since
## ‘maxcat‘ set to 5 categories!
```




```
L1_data <- spotify_data1[c(1,2,3,5,6,7,8,9,10,11,12,13,14,15,16)]
corrplot(cor(L1_data[c(1,2,3,5,6,7,8,9,10,11,12,13)]))
```




```
set.seed(42)
rows1 <- sample(nrow(L1_data))
d1 <- L1_data[rows1, ]

split <- round(nrow(d1) * 0.8)
train.df1 <- d1[1:split, ]
test.df1 <- d1[(split+1):nrow(d1), ]
round(nrow(train.df1)/nrow(d1), digits = 3)
```

```
## [1] 0.8
```

```
spot.lm <- lm(target ~ acousticness + danceability + tempo + time_signature + speechiness + mode + key +

summary(spot.lm)
```

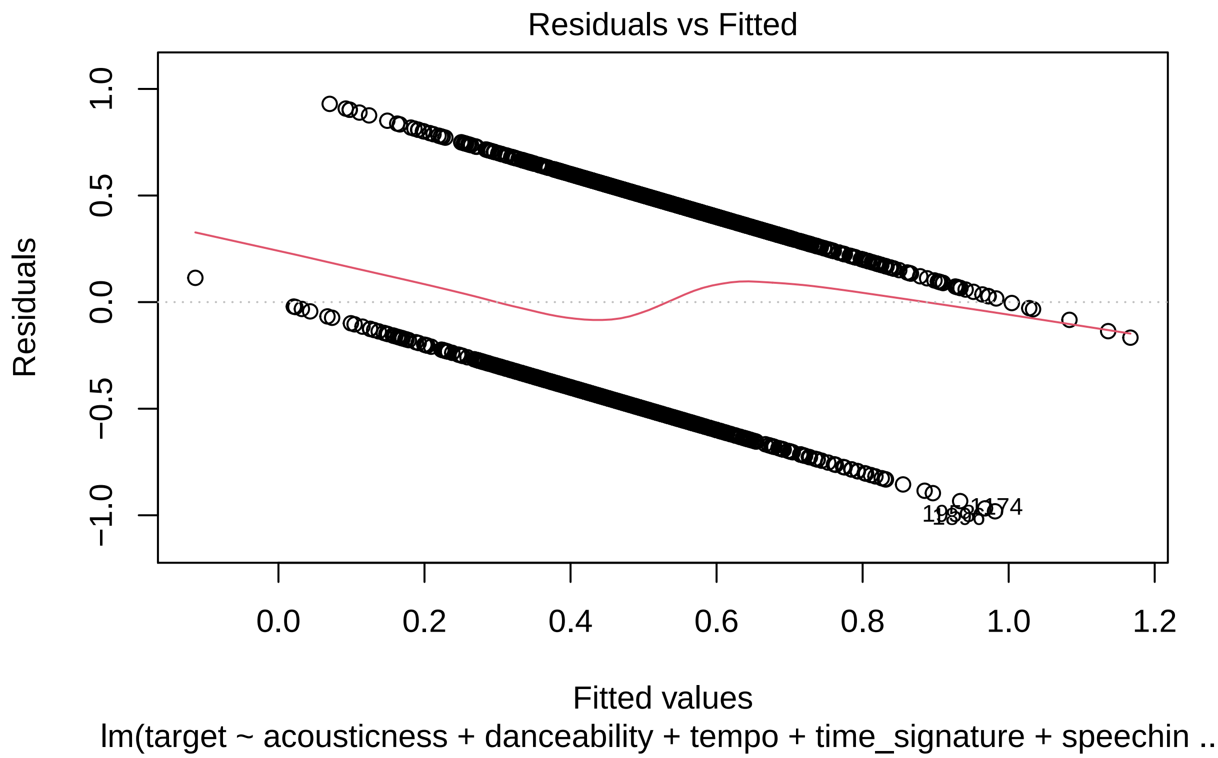
```
##
## Call:
## lm(formula = target ~ acousticness + danceability + tempo + time_signature +
##     speechiness + mode + key + liveness + loudness + valence,
##     data = train.df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98131 -0.45474  0.09155  0.44186  0.92984
```

```
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.0205386  0.2144649  -0.096   0.9237
## acousticness   -0.4153824  0.0566892  -7.327 3.70e-13 ***
## danceability    0.4121728  0.0857233   4.808 1.67e-06 ***
## tempo           0.0007947  0.0004637   1.714   0.0868 .
## time_signature -0.0285370  0.0475552  -0.600   0.5485
## speechiness     0.6455653  0.1344950   4.800 1.74e-06 ***
## mode           -0.0380477  0.0248396  -1.532   0.1258
## key            -0.0010817  0.0033027  -0.328   0.7433
## liveness        0.1696894  0.0780691   2.174   0.0299 *
## loudness       -0.0353806  0.0040336  -8.772  < 2e-16 ***
## valence         0.1143162  0.0544974   2.098   0.0361 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4754 on 1603 degrees of freedom
## Multiple R-squared:  0.1017, Adjusted R-squared:  0.09609
## F-statistic: 18.15 on 10 and 1603 DF,  p-value: < 2.2e-16
```

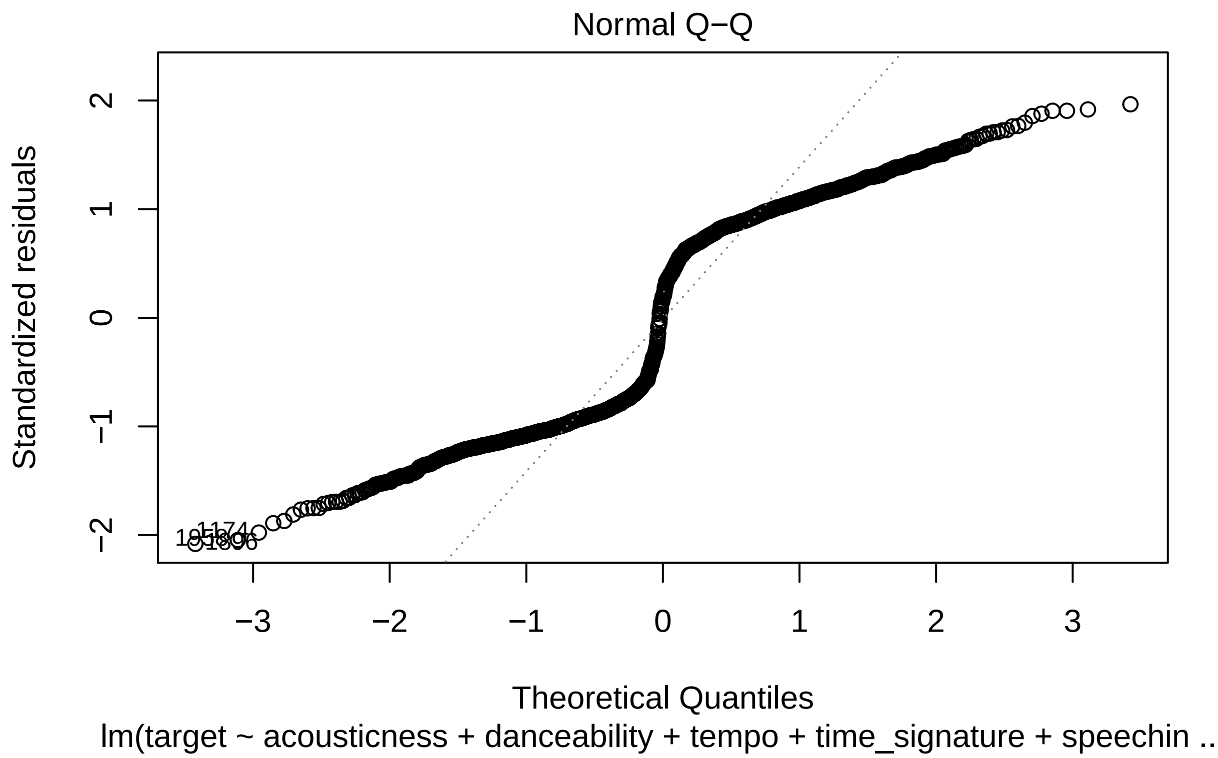
```
spot.pred <- predict(spot.lm,test.df1)

pred_cutoff_50<- ifelse(spot.pred > 0.5, 1, 0)

accuracy(spot.pred, test.df1$target)
```

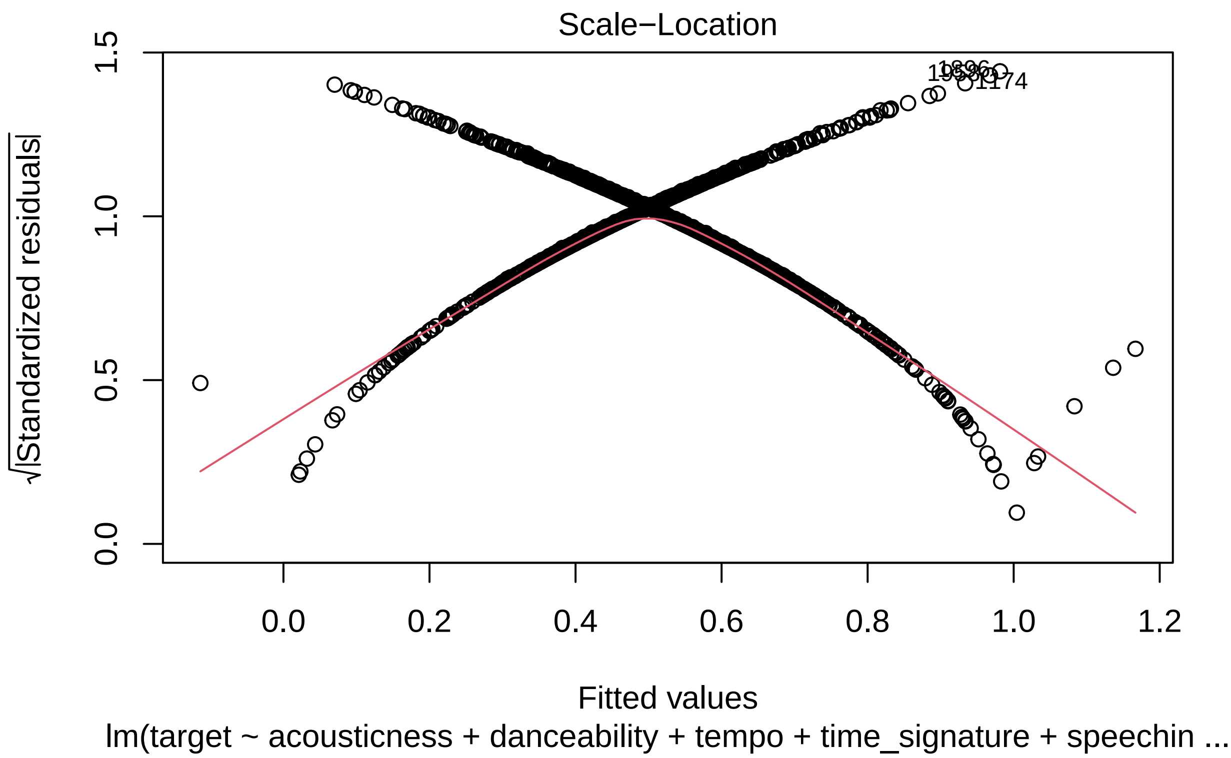
```
##                   ME      RMSE       MAE  MPE MAPE
## Test set -0.02374216 0.4793428 0.4563709 -Inf  Inf
```

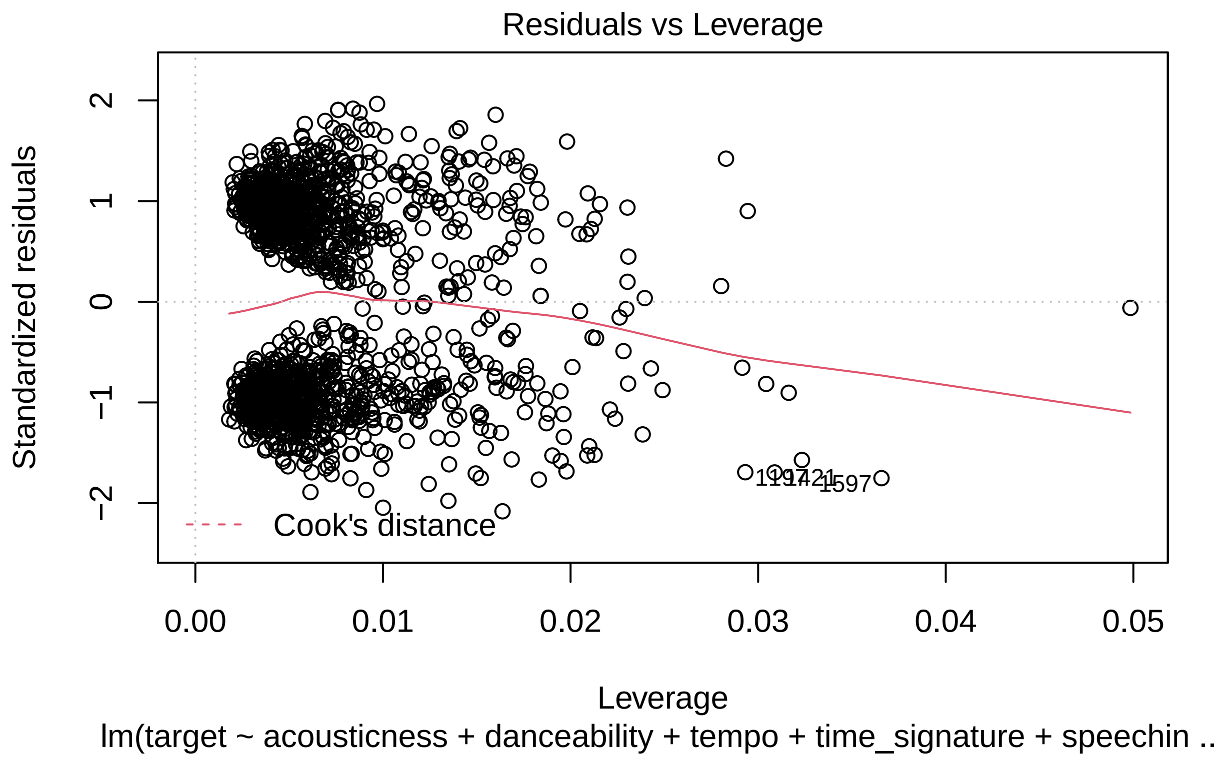
```
plot(spot.lm)
```



Residuals vs Fitted
Residuals
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1174
Fitted values
lm(target ~ acousticness + danceability + tempo + time_signature + speechin ...


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(target ~ acousticness + danceability + tempo + time_signature + speechin ...


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Fitted values
lm(target ~ acousticness + danceability + tempo + time_signature + speechin ...
1174




```r
conf.matrix <- table(Actual = test.df1$target, Predicted = pred_cutoff_50)
conf.matrix
```

```
##       Predicted
## Actual   0   1
##      0 140  70
##      1  75 118
```

```r
accuracy_Testlm <- sum(diag(conf.matrix)) / sum(conf.matrix)
print(paste('Accuracy for test', accuracy_Testlm))
```

```
## [1] "Accuracy for test 0.640198511166253"
```

```r
logit.reg <- glm(target ~ ., data = train.df1[,c(1:13)], family = "binomial")

summary(logit.reg)
```

```
##
## Call:
## glm(formula = target ~ ., family = "binomial", data = train.df1[,
##     c(1:13)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.0481  -1.0581   0.4511   1.0461   2.1721
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.947e+00  1.008e+00  -2.924  0.00345 **
## acousticness     -1.778e+00  2.751e-01  -6.461 1.04e-10 ***
## danceability      1.886e+00  3.990e-01   4.727 2.27e-06 ***
## duration_ms       2.461e-06  7.716e-07   3.190  0.00142 **
## instrumentalness  1.417e+00  2.381e-01   5.953 2.63e-09 ***
## key              -4.509e-03  1.495e-02  -0.302  0.76296
## liveness          5.458e-01  3.582e-01   1.524  0.12756
## loudness         -1.119e-01  2.055e-02  -5.446 5.16e-08 ***
## mode             -1.220e-01  1.127e-01  -1.083  0.27881
## speechiness       3.844e+00  6.640e-01   5.789 7.07e-09 ***
## tempo             3.420e-03  2.109e-03   1.621  0.10500
## time_signature   -1.464e-01  2.204e-01  -0.664  0.50649
## valence           7.943e-01  2.519e-01   3.153  0.00161 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2236.5  on 1613  degrees of freedom
## Residual deviance: 1998.0  on 1601  degrees of freedom
## AIC: 2024
##
## Number of Fisher Scoring iterations: 3
```

```
# Generate odds-ratios
exp(coef(logit.reg))
```

```
##      (Intercept)     acousticness     danceability      duration_ms
##       0.05249596       0.16902355       6.59445251       1.00000246
## instrumentalness              key         liveness         loudness
##       4.12647251       0.99550081       1.72605966       0.89412382
##             mode      speechiness            tempo   time_signature
##       0.88512959      46.71803127       1.00342540       0.86381953
##          valence
##       2.21280623
```

```
logit.reg.pred <- predict(logit.reg, test.df1[,c(1:13)], type = "response")


pred_cutoff_lr<- ifelse(logit.reg.pred > 0.5, 1, 0)
accuracy(logit.reg.pred,test.df1$target)
```

```
##                   ME      RMSE       MAE  MPE MAPE
## Test set -0.02575393 0.4722529 0.4392831 -Inf  Inf
```

```
conf.matrix2 <- table(Actual = test.df1$target, Predicted = pred_cutoff_lr)
conf.matrix2
```

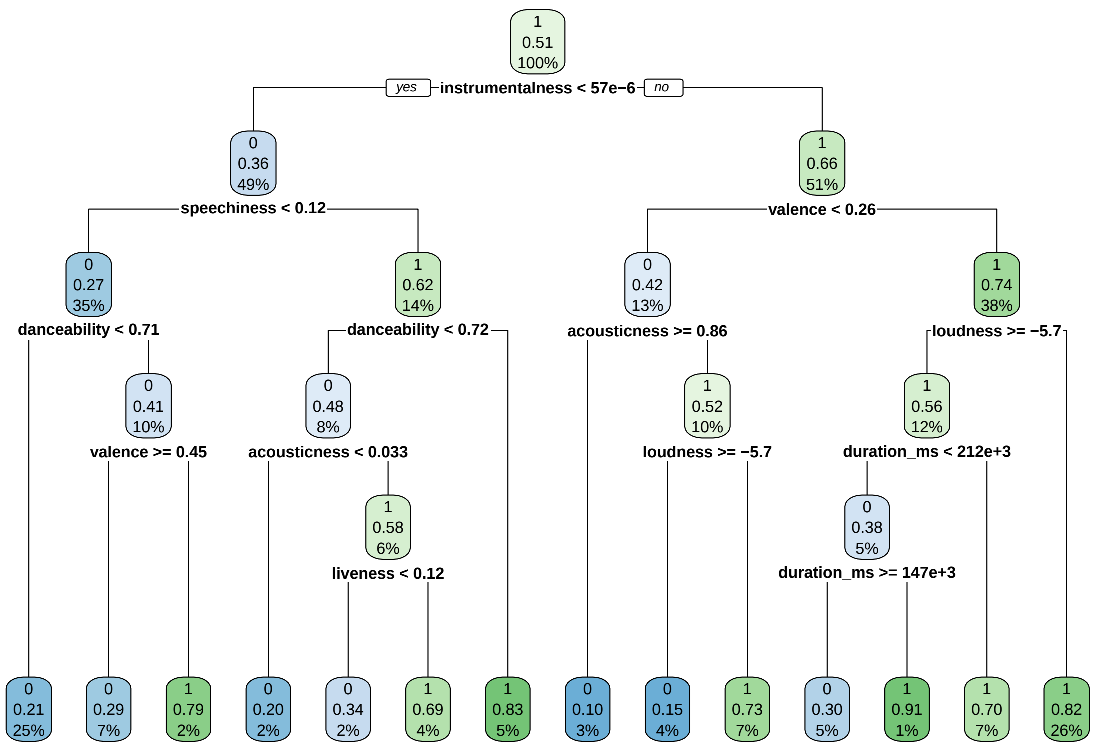
```
##        Predicted
## Actual   0   1
##      0 146  64
##      1  71 122
```

```
accuracy_Test_lr <- sum(diag(conf.matrix2)) / sum(conf.matrix2)
    accuracy_Test_lr
```

```
## [1] 0.6650124
```

```
dt.fit <- rpart(target~., data = train.df1[,c(1:13)], method = 'class')
rpart.plot(dt.fit, extra = 106)
```

```
## Warning: Bad 'data' field in model 'call' (expected a data.frame or a matrix).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```




```
predict_dt <-predict(dt.fit, test.df1, type = 'class')
table_mat <- table(test.df1$target, predict_dt)
table_mat
```

```
##    predict_dt
```

```
##        0   1
##    0 159  51
##    1  58 135
```

```
accuracy_Testdt <- sum(diag(table_mat)) / sum(table_mat)
print(paste('Accuracy for test', accuracy_Testdt))
```

```
## [1] "Accuracy for test 0.729528535980149"
```

```
accuracy_tune <- function(fit) {
    predict_unseen <- predict(dt.fit, test.df1, type = 'class')
    table_mat <- table(test.df1$target, predict_unseen)
    accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
    accuracy_Test
}
```

```
control <- rpart.control(minsplit = 3,
    minbucket = round(3 / 3),
    maxdepth = 3,
    cp = 0.01)
tune_fit <- rpart(target~., data = train.df1[,c(1:13)], method = 'class', control = control)
accuracy_tune(tune_fit)
```

```
## [1] 0.7295285
```

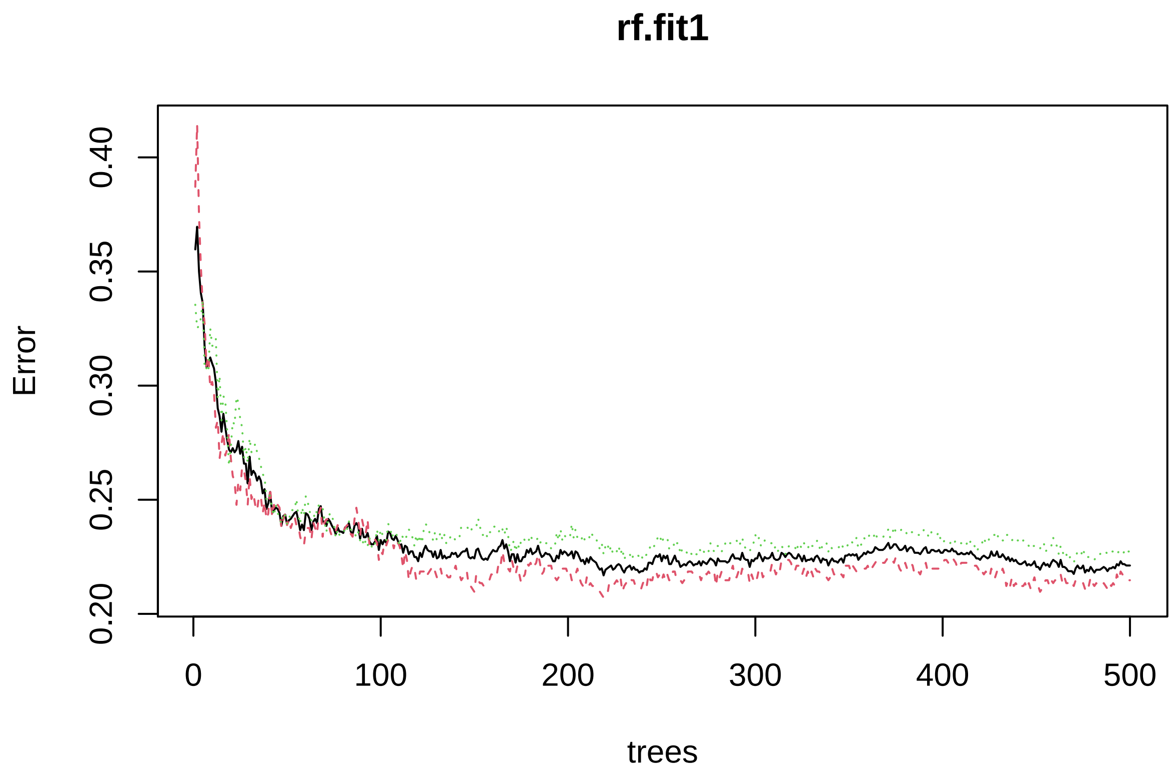
```
print(paste('Accuracy for test', accuracy_tune(tune_fit)))
```

```
## [1] "Accuracy for test 0.729528535980149"
```

```
train.df1$target <- as.factor(train.df1$target)
test.df1$target <- as.factor(test.df1$target)
rf.fit1 <- randomForest(target ~ ., data = train.df1[,c(1:13)], importance = TRUE)
rf.fit1
```

```
##
## Call:
##  randomForest(formula = target ~ ., data = train.df1[, c(1:13)],      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 22.12%
## Confusion matrix:
##     0   1 class.error
## 0 618 169   0.2147395
## 1 188 639   0.2273277
```

```
plot(rf.fit1)
```



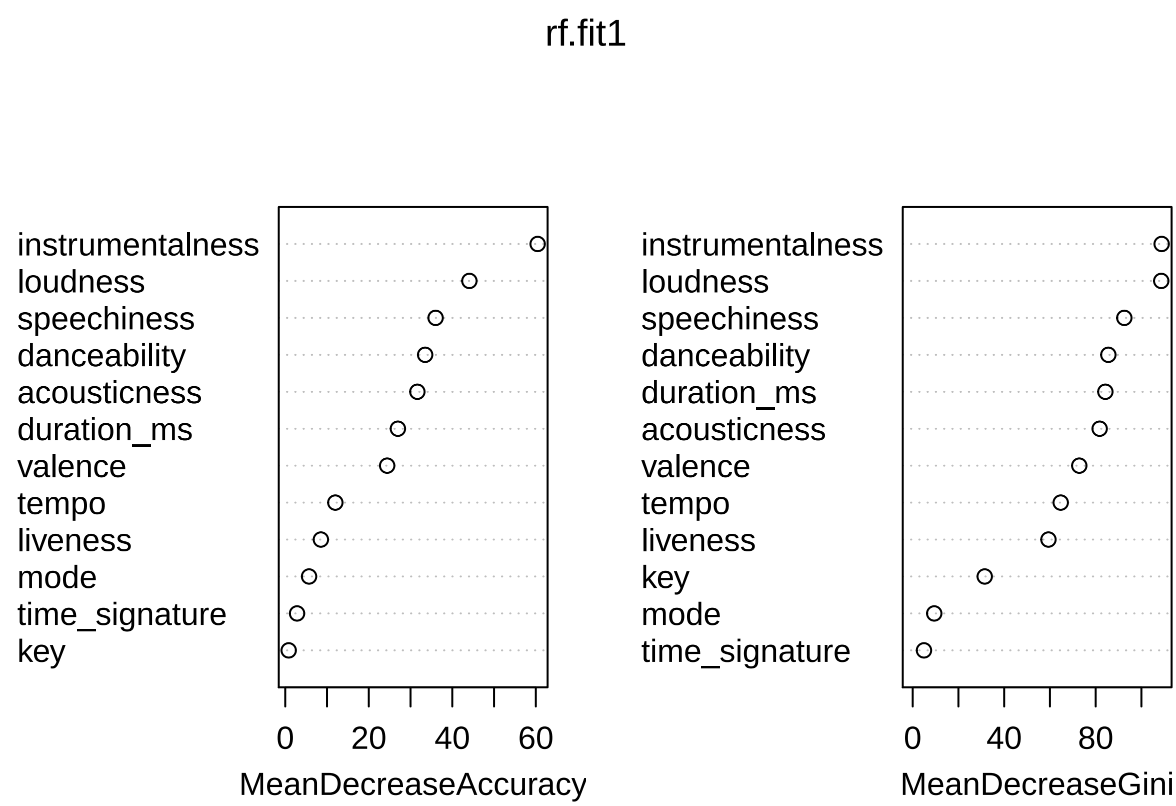

```
prediction <-predict(rf.fit1, test.df1)
pred<- ifelse(prediction == 1, 1, 0)
conf.matrix_rf <- table(Actual = test.df1$target, Predicted = pred)
conf.matrix_rf
```

```
##        Predicted
## Actual   0   1
##      0 171  39
##      1  53 140
```

```
accuracy_Testrf <- sum(diag(conf.matrix_rf)) / sum(conf.matrix_rf)
print(paste('Accuracy for test', accuracy_Testrf))
```

```
## [1] "Accuracy for test 0.771712158808933"
```

```
varImpPlot(rf.fit1)
```




```r
varImp(rf.fit1)
```

```
##                          0          1
## acousticness     18.4249471 18.4249471
## danceability     23.0386170 23.0386170
## duration_ms      18.7861100 18.7861100
## instrumentalness 46.6640409 46.6640409
## key               0.5899723  0.5899723
## liveness          5.9606104  5.9606104
## loudness         30.3980687 30.3980687
## mode              3.8625484  3.8625484
## speechiness      26.6821391 26.6821391
## tempo             8.2099632  8.2099632
## time_signature    2.0762823  2.0762823
## valence          15.7661310 15.7661310
```

```r
dataset_of_10s <- read_csv("dataset-of-10s.csv")
```

```
## Parsed with column specification:
## cols(
##   track = col_character(),
##   artist = col_character(),
##   uri = col_character(),
##   danceability = col_double(),
##   energy = col_double(),
```

```
##   key = col_double(),
##   loudness = col_double(),
##   mode = col_double(),
##   speechiness = col_double(),
##   acousticness = col_double(),
##   instrumentalness = col_double(),
##   liveness = col_double(),
##   valence = col_double(),
##   tempo = col_double(),
##   duration_ms = col_double(),
##   time_signature = col_double(),
##   chorus_hit = col_double(),
##   sections = col_double(),
##   target = col_double()
## )
```

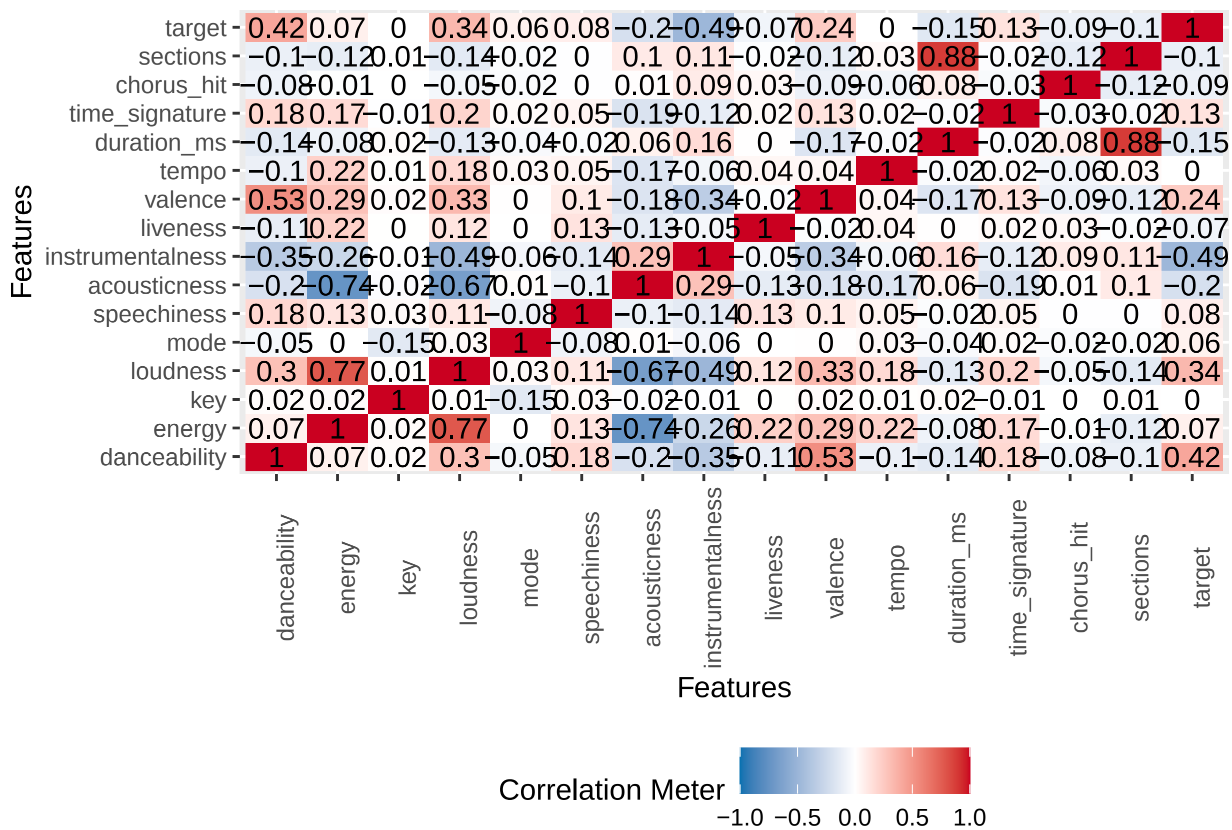
```r
dataset_of_00s <- read_csv("dataset-of-00s.csv")
```

```
## Parsed with column specification:
## cols(
##   track = col_character(),
##   artist = col_character(),
##   uri = col_character(),
##   danceability = col_double(),
##   energy = col_double(),
##   key = col_double(),
##   loudness = col_double(),
##   mode = col_double(),
##   speechiness = col_double(),
##   acousticness = col_double(),
##   instrumentalness = col_double(),
##   liveness = col_double(),
##   valence = col_double(),
##   tempo = col_double(),
##   duration_ms = col_double(),
##   time_signature = col_double(),
##   chorus_hit = col_double(),
##   sections = col_double(),
##   target = col_double()
## )
```

```r
D1 <- rbind(dataset_of_10s,dataset_of_00s)
str(D1)
```

```r
plot_correlation(D1,maxcat = 5L)
```

```
## Warning in dummify(data, maxcat = maxcat): Ignored all discrete features since
## `maxcat` set to 5 categories!
```




```r
summary(D1)
```

```
##      track              artist              uri             danceability
##  Length:12270       Length:12270       Length:12270       Min.   :0.0588
##  Class :character   Class :character   Class :character   1st Qu.:0.4320
##  Mode  :character   Mode  :character   Mode  :character   Median :0.5730
##                                                           Mean   :0.5561
##                                                           3rd Qu.:0.6970
##                                                           Max.   :0.9860
##      energy              key            loudness             mode
##  Min.   :0.000251   Min.   : 0.00   Min.   :-47.327   Min.   :0.0000
##  1st Qu.:0.548000   1st Qu.: 2.00   1st Qu.: -8.375   1st Qu.:0.0000
##  Median :0.727000   Median : 5.00   Median : -6.069   Median :1.0000
##  Mean   :0.680560   Mean   : 5.28   Mean   : -7.523   Mean   :0.6453
##  3rd Qu.:0.871000   3rd Qu.: 8.00   3rd Qu.: -4.585   3rd Qu.:1.0000
##  Max.   :0.999000   Max.   :11.00   Max.   :  1.137   Max.   :1.0000
##   speechiness       acousticness      instrumentalness       liveness
##  Min.   :0.02240   Min.   :0.000000   Min.   :0.000000   Min.   :0.0167
##  1st Qu.:0.03732   1st Qu.:0.006072   1st Qu.:0.000000   1st Qu.:0.0953
##  Median :0.05510   Median :0.063200   Median :0.000019   Median :0.1280
##  Mean   :0.09531   Mean   :0.215706   Mean   :0.158413   Mean   :0.1964
##  3rd Qu.:0.10900   3rd Qu.:0.312000   3rd Qu.:0.051150   3rd Qu.:0.2570
##  Max.   :0.95600   Max.   :0.996000   Max.   :0.998000   Max.   :0.9870
##     valence            tempo          duration_ms       time_signature
##  Min.   :0.0000   Min.   : 39.37   Min.   :  15920   Min.   :0.000
```

```
##  1st Qu.:0.2580   1st Qu.: 97.70   1st Qu.: 199387   1st Qu.:4.000
##  Median :0.4550   Median :120.08   Median : 228846   Median :4.000
##  Mean   :0.4622   Mean   :122.00   Mean   : 246977   Mean   :3.923
##  3rd Qu.:0.6590   3rd Qu.:141.32   3rd Qu.: 269217   3rd Qu.:4.000
##  Max.   :0.9820   Max.   :213.23   Max.   :4170227   Max.   :5.000
##    chorus_hit         sections          target
##  Min.   :  0.00   Min.   :  1.00   Min.   :0.0
##  1st Qu.: 27.82   1st Qu.:  8.00   1st Qu.:0.0
##  Median : 36.18   Median : 10.00   Median :0.5
##  Mean   : 40.89   Mean   : 10.67   Mean   :0.5
##  3rd Qu.: 48.16   3rd Qu.: 12.00   3rd Qu.:1.0
##  Max.   :262.62   Max.   :169.00   Max.   :1.0
```

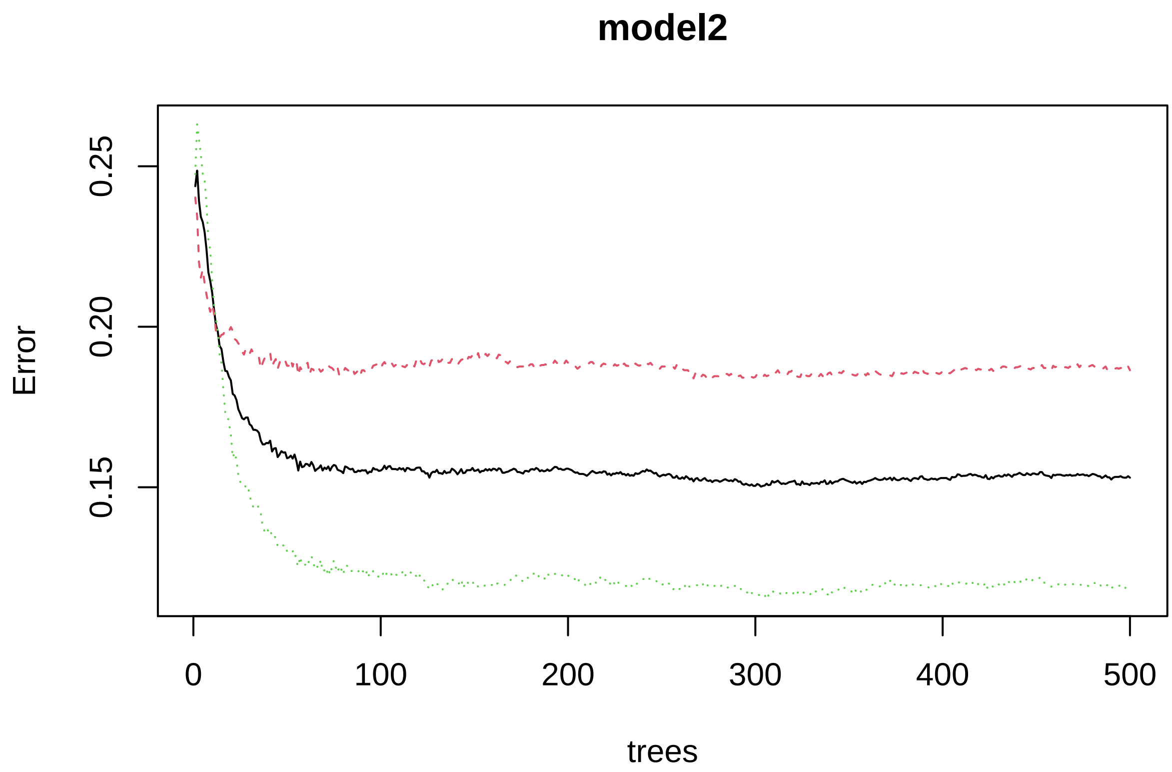
```r
D1 <- subset(D1, select = -c(uri,sections,chorus_hit))
```

```r
set.seed(42)
train.rf <- sample(nrow(D1), 0.7*nrow(D1), replace = FALSE)
TrainSet <- D1[train.rf,]
ValidSet <- D1[-train.rf,]
summary(TrainSet)
summary(ValidSet)

TrainSet <- subset(TrainSet, select = -c(track,artist))
ValidSet <- subset(ValidSet, select = -c(track,artist))
```

```r
TrainSet$target <- as.factor(TrainSet$target)
ValidSet$target <- as.factor(ValidSet$target)

model2 <- randomForest(target ~ ., data = TrainSet, importance = TRUE)
model2
```

```
##
## Call:
##  randomForest(formula = target ~ ., data = TrainSet, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 15.3%
## Confusion matrix:
##      0    1 class.error
## 0 3508  804   0.1864564
## 1  510 3767   0.1192425
```

```r
plot(model2)
```




```r
prediction_model2 <-predict(model2, ValidSet)

confusionMatrix(prediction_model2, ValidSet$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1499  194
##          1  324 1664
##
##                Accuracy : 0.8593
##                  95% CI : (0.8476, 0.8704)
##     No Information Rate : 0.5048
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7183
##
##  Mcnemar's Test P-Value : 1.445e-08
##
##             Sensitivity : 0.8223
##             Specificity : 0.8956
##          Pos Pred Value : 0.8854
##          Neg Pred Value : 0.8370
##              Prevalence : 0.4952
```

```
##          Detection Rate : 0.4072
##    Detection Prevalence : 0.4599
##       Balanced Accuracy : 0.8589
##
##        'Positive' Class : 0
##
```

```
trControl <- trainControl(method = "cv",
    number = 10,
search = "grid")
```

```
rf_default <- train(target~.,
    data = TrainSet,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl)
# Print the results
print(rf_default)
```

```
## Random Forest
##
## 8589 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7730, 7730, 7731, 7731, 7730, 7730, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8473634  0.6948170
##    7    0.8468984  0.6938806
##   13    0.8441045  0.6882843
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(2:10))
rf_mtry <- train(target~.,
    data = TrainSet,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = trControl,
    importance = TRUE,
    nodesize = 14,
    ntree = 300)
print(rf_mtry)
```

```
## Random Forest
```

```
##
## 8589 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7730, 7730, 7729, 7731, 7729, 7731, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8445687  0.6892168
##    3    0.8439860  0.6880440
##    4    0.8437548  0.6875841
##    5    0.8445717  0.6892164
##    6    0.8456180  0.6913077
##    7    0.8444538  0.6889849
##    8    0.8449193  0.6899118
##    9    0.8450385  0.6901486
##   10    0.8450349  0.6901430
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

```r
best_mtry <- rf_mtry$bestTune$mtry
best_mtry
```

```
## [1] 6
```

```r
store_maxnode <- list()
tuneGrid <- expand.grid(.mtry = 2)
for (maxnodes in c(5: 20)) {
    set.seed(1234)
    rf_maxnode <- train(target~.,
        data = TrainSet,
        method = "rf",
        metric = "Accuracy",
        tuneGrid = tuneGrid,
        trControl = trControl,
        importance = TRUE,
        nodesize = 14,
        maxnodes = maxnodes,
        ntree = 300)
    current_iteration <- toString(maxnodes)
    store_maxnode[[current_iteration]] <- rf_maxnode
}
results_mtry <- resamples(store_maxnode)
summary(results_mtry)
```

```
##
## Call:
## summary.resamples(object = results_mtry)
##
```

```
## Models: 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
## Number of resamples: 10
##
## Accuracy
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 5  0.7342657 0.7489843 0.7571321 0.7612052 0.7782305 0.7904540    0
## 6  0.7540793 0.7595930 0.7740257 0.7744780 0.7878347 0.7974389    0
## 7  0.7564103 0.7716946 0.7852154 0.7850744 0.8005819 0.8079162    0
## 8  0.7785548 0.7819138 0.7899965 0.7926424 0.8050058 0.8067520    0
## 9  0.7832168 0.7894576 0.7916182 0.7948539 0.8002906 0.8114086    0
## 10 0.7802326 0.7847604 0.7986030 0.7961370 0.8061700 0.8100233    0
## 11 0.7797203 0.7905139 0.7981387 0.7984617 0.8092065 0.8125728    0
## 12 0.7843823 0.7974978 0.8016284 0.8028850 0.8122268 0.8207218    0
## 13 0.7878788 0.7990697 0.8026793 0.8028852 0.8099534 0.8160652    0
## 14 0.7913753 0.7970930 0.8048946 0.8054480 0.8140279 0.8195576    0
## 15 0.7925408 0.8022691 0.8132611 0.8110348 0.8203782 0.8242142    0
## 16 0.7925408 0.8109920 0.8144279 0.8130135 0.8200868 0.8265425    0
## 17 0.7902098 0.8077947 0.8144266 0.8138284 0.8194009 0.8381839    0
## 18 0.7913753 0.8041327 0.8149035 0.8121967 0.8204832 0.8300349    0
## 19 0.7902098 0.8024427 0.8155894 0.8124315 0.8207218 0.8323632    0
## 20 0.7937063 0.8066326 0.8179170 0.8158080 0.8253275 0.8335274    0
##
## Kappa
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 5  0.4695690 0.4989035 0.5151158 0.5232043 0.5570610 0.5814271    0
## 6  0.5090175 0.5200600 0.5487458 0.5496246 0.5761994 0.5953006    0
## 7  0.5136292 0.5440992 0.5709329 0.5707275 0.6016189 0.6161991    0
## 8  0.5577056 0.5643444 0.5805219 0.5857686 0.6103980 0.6138636    0
## 9  0.5670170 0.5794003 0.5836646 0.5901785 0.6010073 0.6231497    0
## 10 0.5610017 0.5701310 0.5976168 0.5927452 0.6127021 0.6205373    0
## 11 0.5599904 0.5814934 0.5966981 0.5973687 0.6188710 0.6254851    0
## 12 0.5692935 0.5954104 0.6036274 0.6061691 0.6248200 0.6417188    0
## 13 0.5762457 0.5985602 0.6058436 0.6061656 0.6202334 0.6324367    0
## 14 0.5832352 0.5946332 0.6101948 0.6112670 0.6283468 0.6394070    0
## 15 0.5854150 0.6049984 0.6268238 0.6224026 0.6410239 0.6487012    0
## 16 0.5854330 0.6223640 0.6291431 0.6263414 0.6404704 0.6533373    0
## 17 0.5808659 0.6159664 0.6291300 0.6279653 0.6391792 0.6765296    0
## 18 0.5830813 0.6085930 0.6301800 0.6246983 0.6412202 0.6602646    0
## 19 0.5807112 0.6052435 0.6314449 0.6251632 0.6417057 0.6649294    0
## 20 0.5878112 0.6136896 0.6361178 0.6319114 0.6508729 0.6672049    0
```

```r
store_maxtrees <- list()
for (ntree in c(250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000)) {
    set.seed(5678)
    rf_maxtrees <- train(target~.,
        data = TrainSet,
        method = "rf",
        metric = "Accuracy",
        tuneGrid = tuneGrid,
        trControl = trControl,
        importance = TRUE,
        nodesize = 14,
        maxnodes = 17,
        ntree = ntree)
```

```r
    key <- toString(ntree)
    store_maxtrees[[key]] <- rf_maxtrees
}
results_tree <- resamples(store_maxtrees)
summary(results_tree)
```

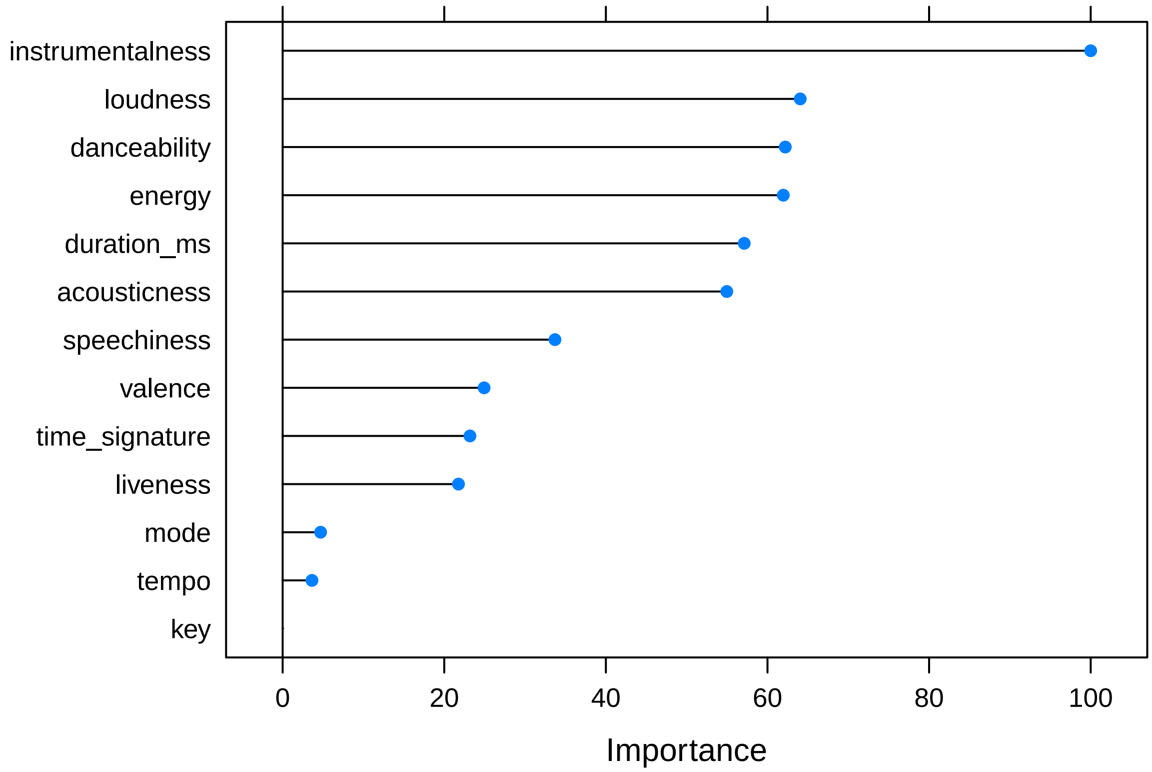
```
##
## Call:
## summary.resamples(object = results_tree)
##
## Models: 250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 250  0.7892899 0.7968002 0.8150060 0.8127846 0.8236321 0.8381839    0
## 300  0.7904540 0.7979646 0.8160652 0.8133670 0.8205871 0.8381839    0
## 350  0.7904540 0.7991291 0.8172293 0.8138326 0.8216470 0.8393481    0
## 400  0.7892899 0.7953453 0.8154831 0.8122023 0.8216470 0.8370198    0
## 450  0.7869616 0.7994205 0.8132611 0.8120868 0.8192666 0.8370198    0
## 500  0.7892899 0.8005850 0.8172293 0.8137161 0.8208781 0.8393481    0
## 550  0.7881257 0.7979657 0.8160652 0.8134834 0.8217502 0.8405122    0
## 600  0.7892899 0.7997129 0.8161715 0.8137169 0.8227590 0.8370198    0
## 800  0.7857974 0.7962194 0.8183935 0.8132513 0.8245580 0.8381839    0
## 1000 0.7892899 0.7985474 0.8183935 0.8144151 0.8236839 0.8391608    0
## 2000 0.7869616 0.7979646 0.8172293 0.8137164 0.8233932 0.8379953    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 250  0.5788451 0.5939104 0.6303464 0.6258635 0.6474835 0.6766418    0
## 300  0.5811754 0.5962711 0.6323868 0.6270256 0.6414893 0.6766505    0
## 350  0.5812434 0.5985577 0.6347102 0.6279557 0.6435541 0.6789637    0
## 400  0.5789136 0.5910358 0.6312130 0.6247035 0.6435672 0.6743199    0
## 450  0.5742676 0.5991715 0.6268396 0.6244741 0.6387572 0.6743199    0
## 500  0.5789136 0.6014860 0.6347074 0.6277273 0.6420645 0.6789724    0
## 550  0.5766112 0.5962629 0.6324248 0.6272682 0.6437868 0.6812944    0
## 600  0.5789410 0.5997488 0.6326336 0.6277337 0.6457439 0.6743375    0
## 800  0.5719586 0.5927467 0.6370776 0.6267993 0.6493421 0.6766330    0
## 1000 0.5789273 0.5973909 0.6370776 0.6291209 0.6476032 0.6785802    0
## 2000 0.5742815 0.5962222 0.6347512 0.6277282 0.6470194 0.6762616    0
```

```r
fit_rf <- train(target~.,
    TrainSet,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = trControl,
    importance = TRUE,
    nodesize = 14,
    ntree = 550,
    maxnodes = 17)
```

```r
prediction <-predict(fit_rf, ValidSet)

confusionMatrix(prediction, ValidSet$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1306  124
##          1  517 1734
##
##                Accuracy : 0.8259
##                  95% CI : (0.8132, 0.838)
##     No Information Rate : 0.5048
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.651
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7164
##             Specificity : 0.9333
##          Pos Pred Value : 0.9133
##          Neg Pred Value : 0.7703
##              Prevalence : 0.4952
##          Detection Rate : 0.3548
##    Detection Prevalence : 0.3885
##       Balanced Accuracy : 0.8248
##
##        'Positive' Class : 0
##
```

```r
plot(varImp(fit_rf))
```




```r
set.seed(42)
Datatrain <- spotify_data1

Datatest <- D1[,c(1,2,9,3,14,4,10,5,11,6,7,8,13,15,12,16)]

Datatest.rf <- sample(nrow(Datatest), 0.5*nrow(Datatest), replace = FALSE)

Datatest.rf1 <- Datatest[Datatest.rf,]
```

```r
Datatrain$target <- as.factor(Datatrain$target)
older_songs_pred <- randomForest(target ~ ., data = Datatrain[,c(1:14)], importance = TRUE)


old_song_list <-predict(older_songs_pred, Datatest.rf1)
old_pred<- ifelse(old_song_list==1, 1, 0)


conf.matrix.old <- table(Actual = Datatest.rf1$target, Predicted = old_pred )
conf.matrix.old
```

```
##         Predicted
## Actual     0    1
##        0 1593 1516
##        1 2058  968
```

```r
old_recommend <- cbind(Datatest.rf1[,c(1,2)],old_pred)
list <- old_recommend %>%  filter(old_pred==1)
head(list)
```

```
##                               track       artist old_pred
## 1    Father Stretch My Hands Pt. 1   Kanye West        1
## 2                 Wu-Tang Forever        Drake        1
## 3   T.R.U.C.E. - Feat. Noyz Narcos Metal Carter        1
## 4                             Texas Cass McCombs        1
## 5 Airwave [Mix Cut] - Original Mix       Rank 1        1
## 6     Victoriae & Triumphi Dominus      Turisas        1
```