Spotify song popularity reggresion Analysis

kite-luva

2025-06-09

```
knitr::opts_chunk$set(echo = FALSE)
#================================#
## RF, XGBoost, and GBM PREDICTIVE CODES ##
#================================#

library(skimr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
```

```
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(ggplot2)
library(writexl)
library(openxlsx)
library(rlang)
```

```
##
## Attaching package: 'rlang'
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(grid)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(gbm)
```

```
## Loaded gbm 2.2.2
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```
library(recipes)
```

```
##
## Attaching package: 'recipes'
##
## The following object is masked from 'package:stringr':
```

```
##
##     fixed
##
## The following object is masked from 'package:stats':
##
##     step
```

```
library(lubridate)

## Load the dataset
spotify_charts_2024 <- read_csv("~/school docs/universal_top_spotify_songs.new.csv")
```

```
## Rows: 1750032 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (5): spotify_id, name, artists, country, album_name
## dbl  (17): daily_rank, daily_movement, weekly_movement, popularity, duration...
## lgl   (1): is_explicit
## date  (2): snapshot_date, album_release_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
view(spotify_charts_2024)

###--------------------------------
##DATA CLEANING
###--------------------------------

# market counts for each song
spotify_charts_2024 <- spotify_charts_2024 %>%
  group_by(spotify_id) %>%
  mutate(
    market_count = n_distinct(country, na.rm = TRUE)
  ) %>%
  ungroup()

# remove all duplicates, pick the most popular song(country)
spotify_charts_2024 <- spotify_charts_2024 %>%
  group_by(spotify_id) %>%
  mutate(
    other_charted_countries = paste(country[!duplicated(country)], collapse = ", ")
  ) %>%
  slice_max(order_by = popularity, n = 1, with_ties = FALSE) %>%
  ungroup()

# Function to count the number of artists in each row
spotify_charts_2024$artist_count <- sapply(strsplit(spotify_charts_2024$artists, ","), length)

# Convert character date columns to Date objects using mdy()
spotify_charts_2024 <- spotify_charts_2024 %>%
  mutate(
    snapshot_date = ymd(snapshot_date),
```

```
    album_release_date = ymd(album_release_date),
    days_out = as.numeric(snapshot_date - album_release_date))

# change boolean into integers
spotify_charts_2024$is_explicit<- as.integer(spotify_charts_2024$is_explicit)

# Standardize duration_ms (convert to minutes)
spotify_charts_2024$duration_ms <- spotify_charts_2024$duration_ms / 60000
colnames(spotify_charts_2024)[colnames(spotify_charts_2024) == "duration_ms"] <- "duration_min"

# Remove unneeded columns
spotify_modelling <- spotify_charts_2024 %>%
  select(-country, -other_charted_countries, -snapshot_date, -name, -artists, -album_name, -album_releas

# check for missing values
colSums(is.na(spotify_modelling))
```

```
##       daily_rank   daily_movement  weekly_movement       popularity
##                0               0                0                0
##      is_explicit     duration_min     danceability           energy
##                0               0                0                0
##              key         loudness             mode      speechiness
##                0               0                0                0
##     acousticness instrumentalness         liveness          valence
##                0               0                0                0
##            tempo   time_signature     market_count     artist_count
##                0               0                0                0
##         days_out
##               13
```

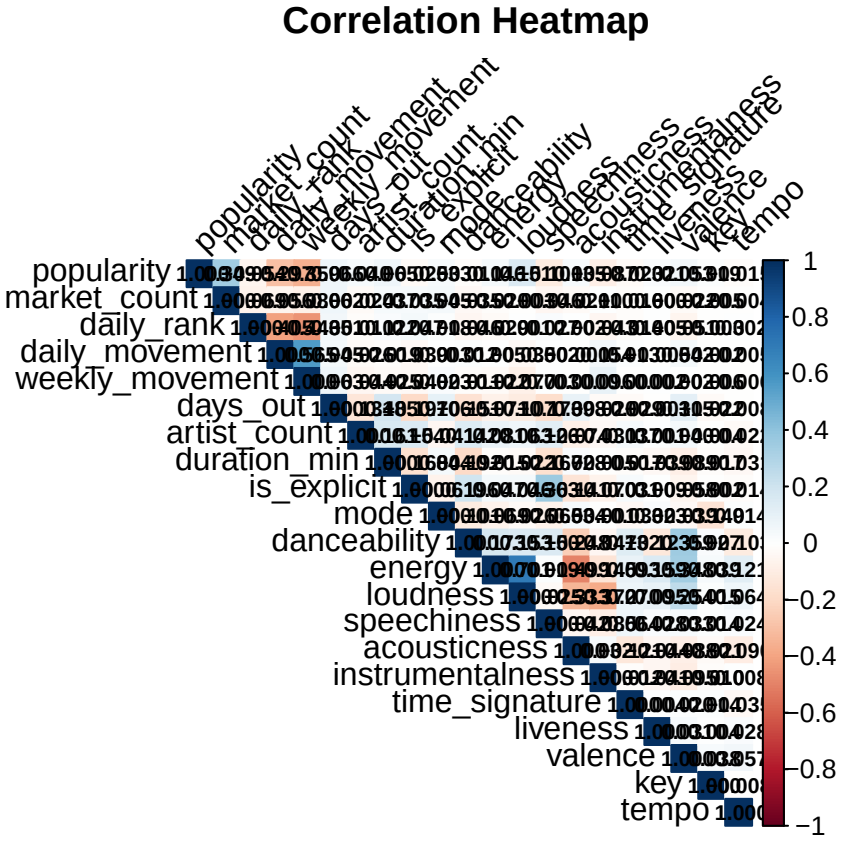
```
#Handle missing values
spotify_modelling <- spotify_modelling %>%
  mutate_all(~ifelse(is.na(.), mean(., na.rm = TRUE), .))

# arrange the columns
spotify_modelling <- spotify_modelling %>%
  select(popularity, days_out, artist_count, market_count, daily_rank, daily_movement,
         weekly_movement, duration_min, is_explicit, mode, danceability, energy, loudness,
         speechiness, acousticness, instrumentalness, liveness, valence, tempo, key, time_signature )

#remove popularity 0
spotify_modelling <- spotify_modelling %>%
  filter(popularity != 0) %>%
  arrange(desc(popularity))

#------------------------------
### DESCRIPTIVE STATISTICS
#-------------------------------------

# Function to compute basic statistics
spotify_stats <- function(column) {
  stats <- c(
    Mean = mean(column, na.rm = TRUE),
```

```
    Median = median(column, na.rm = TRUE),
    SD = sd(column, na.rm = TRUE),
    Variance = var(column, na.rm = TRUE),
    IQR = IQR(column, na.rm = TRUE)
  )
  return(stats)
}

# Loop through columns and compute statistics
stats_results <- lapply(spotify_modelling, spotify_stats)
names(stats_results) <- colnames(spotify_modelling)

# Convert the list of statistics to a data frame for better printing
stats_table <- as.data.frame(stats_results)
print(stats_table)
```

```
##          popularity     days_out artist_count market_count daily_rank
## Mean       58.21686     947.9794     1.631360     2.201072   35.10250
## Median     60.00000      46.0000     1.000000     1.000000   40.00000
## SD         17.93296    2719.6112     1.000970     4.725443   14.35924
## Variance  321.59096 7396285.1214     1.001941    22.329815  206.18770
## IQR        23.00000     233.5000     1.000000     0.000000   21.00000
##          daily_movement weekly_movement duration_min is_explicit      mode
## Mean           2.316623        3.296669    3.1840513   0.3280389 0.5115898
## Median         0.000000        1.000000    3.0280833   0.0000000 1.0000000
## SD            12.077978       14.798967    0.9853391   0.4695107 0.4998781
## Variance     145.877557      219.009422    0.9708932   0.2204403 0.2498781
## IQR            7.000000       13.000000    0.9951083   1.0000000 1.0000000
##          danceability     energy  loudness speechiness acousticness
## Mean       0.67312286 0.65018782 -7.076078  0.12091624   0.27773925
## Median     0.69100000 0.66800000 -6.582000  0.06950000   0.20100000
## SD         0.14019752 0.17081241  3.195613  0.11258703   0.25132402
## Variance   0.01965534 0.02917688 10.211945  0.01267584   0.06316376
## IQR        0.19100000 0.23100000  3.206000  0.12260000   0.37435000
##          instrumentalness   liveness    valence     tempo       key
## Mean           0.02733874 0.18292358 0.53476607 122.33293  5.371619
## Median         0.00000102 0.12700000 0.53400000 121.03000  6.000000
## SD             0.12895952 0.13937591 0.22941330  27.86529  3.600955
## Variance       0.01663056 0.01942564 0.05263046 776.47437 12.966876
## IQR            0.00013700 0.12660000 0.35600000  40.05050  7.000000
##          time_signature
## Mean          3.9455998
## Median        4.0000000
## SD            0.3490409
## Variance      0.1218295
## IQR           0.0000000
```

```
#----------------------------
### CORRELATION
#-------------------------
# Select the audio feature columns
audio_features <- spotify_modelling %>%
  select(popularity, market_count, daily_rank, daily_movement, weekly_movement, days_out, artist_count,
```

```
        duration_min, is_explicit, mode, danceability, energy, loudness, speechiness,
        acousticness, instrumentalness, time_signature, liveness, valence, key, tempo)
# Compute correlation matrix
correlation_matrix <- cor(audio_features, use = "complete.obs")
# Plot heatmap with correlation values
corrplot(correlation_matrix,
        method = "color",
        tl.col = "black",
        tl.srt = 45,
        type = "upper",
        addCoef.col = "black",
        number.cex = 0.7,
        number.digits = 3,
        main = "Correlation Heatmap",
        mar = c(0, 0, 2, 0))
```




```
#-------------------------
### ANOVA FOR VARIANCE IMPORTANCE
#--------------------------

# Convert relevant columns to factors if they are not already
spotify_modelling$mode <- as.factor(spotify_modelling$mode)
spotify_modelling$is_explicit <- as.factor(spotify_modelling$is_explicit)
spotify_modelling$key <- as.factor(spotify_modelling$key)
spotify_modelling$time_signature <- as.factor(spotify_modelling$time_signature)
```

```r
# Perform ANOVA for each predictor against popularity
anova_results <- lapply(names(spotify_modelling)[-which(names(spotify_modelling) == "popularity")], fun
  x <- spotify_modelling[[col]]
  if (is.factor(x) && nlevels(x) > 1) {
    # Check if the factor has more than one level
    aov_result <- aov(popularity ~ x, data = spotify_modelling)
    aov_summary <- summary(aov_result)[[1]]
    print(paste("ANOVA Summary for column:", col))
    print(str(aov_summary)) # Print the structure of aov_summary
    if (is.matrix(aov_summary) || is.data.frame(aov_summary)) {
      f_value <- aov_summary["F value"][1, 1]
      p_value <- aov_summary["Pr(>F)"][1, 1]
      return(list(
        Feature = col, # Include the feature name in the result
        F_Value = f_value,
        P_Value = p_value
      )) # Return F and P-value
    } else {
      return(list(Feature = col, F_Value = NA, P_Value = NA))
    }
  } else if (!is.factor(x)) {
    lm_result <- lm(popularity ~ x, data = spotify_modelling)
    print(paste("LM Summary for column:", col))
    print(str(summary(lm_result)))
    if (length(lm_result$coefficients) > 1) { # check if the model was properly fitted
      return(list(
        Feature = col, # Include the feature name in the result
        F_Value = summary(lm_result)$fstatistic[1],
        P_Value = summary(lm_result)$coefficients[2, "Pr(>|t|)"]
      ))
    } else {
      return(list(Feature = col, F_Value = NA, P_Value = NA))
    }
  } else {
    return(list(Feature = col, F_Value = NA, P_Value = NA)) # Return NA for factors with only one level
  }
})
```

```
## [1] "LM Summary for column: days_out"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e481f568>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
```

```
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 37.6 41.5 42.2 42.1 42.2 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 5.78e+01 4.35e-04 1.34e-01 4.64e-05 4.33e+02 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00436
##  $ adj.r.squared: num 0.00431
##  $ fstatistic   : Named num [1:3] 88.2 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 5.57e-05 -6.36e-09 -6.36e-09 6.71e-12
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: artist_count"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e2513388>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 42.2 42.2 42.2 41.5 42.2 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 57.05 0.715 0.241 0.126 236.333 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00159
##  $ adj.r.squared: num 0.00154
##  $ fstatistic   : Named num [1:3] 32.2 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 1.81e-04 -8.08e-05 -8.08e-05 4.95e-05
```

```
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: market_count"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e4a34378>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] -38.84 -9.67 -32.21 -46.8 -36.19 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 55.2979 1.3261 0.1306 0.0251 423.4396 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 16.8
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.122
##  $ adj.r.squared: num 0.122
##  $ fstatistic   : Named num [1:3] 2802 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 6.04e-05 -4.89e-06 -4.89e-06 2.22e-06
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: daily_rank"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
```

```
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e35a7a50>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 40.4 39.8 40 39.5 39.6 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 60.56473 -0.06689 0.33323 0.00879 181.74916 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00287
##  $ adj.r.squared: num 0.00282
##  $ fstatistic   : Named num [1:3] 57.9 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 3.46e-04 -8.45e-06 -8.45e-06 2.41e-07
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: daily_movement"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e2067d68>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 36.3 40.8 40.3 40.8 40.8 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 59.2394 -0.44139 0.12283 0.00999 482.28066 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.1
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.0884
##  $ adj.r.squared: num 0.0883
```

```
##  $ fstatistic   : Named num [1:3] 1953 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 5.15e-05 -7.88e-07 -7.88e-07 3.40e-07
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: weekly_movement"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e0a28a58>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 35.6 39.9 38.6 40.3 40.3 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 59.65115 -0.43507 0.12081 0.00797 493.75609 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 16.7
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.129
##  $ adj.r.squared: num 0.129
##  $ fstatistic   : Named num [1:3] 2981 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 5.21e-05 -7.47e-07 -7.47e-07 2.27e-07
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: duration_min"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
```

```
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254df87d490>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 40.8 42 42.7 40.6 42.1 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 54.465 1.178 0.426 0.128 127.705 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00419
##  $ adj.r.squared: num 0.00414
##  $ fstatistic   : Named num [1:3] 84.8 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 5.68e-04 -1.63e-04 -1.63e-04 5.11e-05
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "ANOVA Summary for column: is_explicit"
## Classes 'anova' and 'data.frame':    2 obs. of  5 variables:
##  $ Df     : num  1 20145
##  $ Sum Sq : num  4093 6474679
##  $ Mean Sq: num  4093 321
##  $ F value: num  12.7 NA
##  $ Pr(>F) : num  0.00036 NA
## NULL
## [1] "ANOVA Summary for column: mode"
## Classes 'anova' and 'data.frame':    2 obs. of  5 variables:
##  $ Df     : num  1 20145
##  $ Sum Sq : num  7004 6471768
##  $ Mean Sq: num  7004 321
##  $ F value: num  21.8 NA
##  $ Pr(>F) : num  3.04e-06 NA
## NULL
## [1] "LM Summary for column: danceability"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
```

```
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254da97f208>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 41.3 41.6 41.7 41.6 41.8 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 59.186 -1.44 0.62 0.901 95.522 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.000127
##  $ adj.r.squared: num 7.7e-05
##  $ fstatistic   : Named num [1:3] 2.55 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 0.00119 -0.0017 -0.0017 0.00253
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: energy"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254d975dbe0>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 41.9 41.5 40.3 42.1 41.2 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 55.06 4.855 0.497 0.739 110.846 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
```

```
##  $ r.squared    : num 0.00214
##  $ adj.r.squared: num 0.00209
##  $ fstatistic   : Named num [1:3] 43.2 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 0.000769 -0.001106 -0.001106 0.001701
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: loudness"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x0000025481cbf5f8>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 42.1 40.6 39.4 42.4 40.4 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 64.2325 0.8501 0.3034 0.0391 211.6839 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.7
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.023
##  $ adj.r.squared: num 0.0229
##  $ fstatistic   : Named num [1:3] 473 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 2.93e-04 3.44e-05 3.44e-05 4.86e-06
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: speechiness"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
```

```
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x0000025480c39568>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 40.5 42.4 40.6 40.3 40.3 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 60.165 -16.108 0.184 1.116 326.17 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.8
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.0102
##  $ adj.r.squared: num 0.0102
##  $ fstatistic   : Named num [1:3] 208 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 0.000107 -0.000473 -0.000473 0.003916
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: acousticness"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e1f5fe10>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 42.1 42.2 42.5 41.7 42.2 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 57.52 2.509 0.188 0.502 305.65 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
```

```
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00124
##  $ adj.r.squared: num 0.00119
##  $ fstatistic   : Named num [1:3] 24.9 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 0.00011 -0.000218 -0.000218 0.000786
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: instrumentalness"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254dfd8fbd8>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 41.5 41.5 41.6 41.5 41.5 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 58.548 -12.102 0.129 0.976 455.048 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00757
##  $ adj.r.squared: num 0.00752
##  $ fstatistic   : Named num [1:3] 154 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 5.19e-05 -8.16e-05 -8.16e-05 2.98e-03
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: liveness"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
```

```
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e7292d50>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 42.1 42 42 41.9 41.8 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 57.733 2.645 0.208 0.906 276.992 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.000423
##  $ adj.r.squared: num 0.000373
##  $ fstatistic   : Named num [1:3] 8.52 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 0.000135 -0.000467 -0.000467 0.002555
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: valence"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e5e195a0>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 42.6 41.7 40.9 41.8 41.1 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 55.98 4.18 0.32 0.55 174.94 ...
##   ..- attr(*, "dimnames")=List of 2
```

```
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.00286
##  $ adj.r.squared: num 0.00281
##  $ fstatistic   : Named num [1:3] 57.7 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 0.000319 -0.000504 -0.000504 0.000943
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "LM Summary for column: tempo"
## List of 11
##  $ call         : language lm(formula = popularity ~ x, data = spotify_modelling)
##  $ terms        :Classes 'terms', 'formula'  language popularity ~ x
##   .. ..- attr(*, "variables")= language list(popularity, x)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "popularity" "x"
##   .. .. .. ..$ : chr "x"
##   .. ..- attr(*, "term.labels")= chr "x"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: 0x00000254e48868f0>
##   .. ..- attr(*, "predvars")= language list(popularity, x)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "popularity" "x"
##  $ residuals    : Named num [1:20147] 42 42.2 42.1 42.1 41.6 ...
##   ..- attr(*, "names")= chr [1:20147] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] 59.3638 -0.00938 0.56883 0.00453 104.3605 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
##  $ aliased      : Named logi [1:2] FALSE FALSE
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
##  $ sigma        : num 17.9
##  $ df           : int [1:3] 2 20145 2
##  $ r.squared    : num 0.000212
##  $ adj.r.squared: num 0.000163
##  $ fstatistic   : Named num [1:3] 4.28 1 20145
##   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
##  $ cov.unscaled : num [1:2, 1:2] 1.01e-03 -7.82e-06 -7.82e-06 6.39e-08
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##   .. ..$ : chr [1:2] "(Intercept)" "x"
##  - attr(*, "class")= chr "summary.lm"
## NULL
## [1] "ANOVA Summary for column: key"
```

```
## Classes 'anova' and 'data.frame':    2 obs. of  5 variables:
##  $ Df     : num  11 20135
##  $ Sum Sq : num  5207 6473565
##  $ Mean Sq: num  473 322
##  $ F value: num  1.47 NA
##  $ Pr(>F) : num  0.134 NA
## NULL
## [1] "ANOVA Summary for column: time_signature"
## Classes 'anova' and 'data.frame':    2 obs. of  5 variables:
##  $ Df     : num  4 20142
##  $ Sum Sq : num  7346 6471425
##  $ Mean Sq: num  1837 321
##  $ F value: num  5.72 NA
##  $ Pr(>F) : num  0.000135 NA
## NULL
```

```r
# Extract F-values and p-values
f_values <- sapply(anova_results, function(x) {
  if (is.null(x) || is.na(x["F_Value"])) {
    return(NA)
  }
  return(x[["F_Value"]])
})

p_values <- sapply(anova_results, function(x) {
  if (is.null(x) || is.na(x["P_Value"])) {
    return(NA)
  }
  return(x[["P_Value"]])
})

# Print F and P values
for (i in seq_along(anova_results)) {
  cat("Feature: ", anova_results[[i]]$Feature, "\n")
  cat("  F-Value: ", f_values[i], "\n")
  cat("  P-Value: ", p_values[i], "\n\n")
}
```

```
## Feature:  days_out
##   F-Value:  88.15287
##   P-Value:  6.684649e-21
##
## Feature:  artist_count
##   F-Value:  32.15626
##   P-Value:  1.442066e-08
##
## Feature:  market_count
##   F-Value:  2802.119
##   P-Value:  0
##
## Feature:  daily_rank
##   F-Value:  57.94914
##   P-Value:  2.808079e-14
##
```

```
## Feature:  daily_movement
##   F-Value:  1952.948
##   P-Value:  0
##
## Feature:  weekly_movement
##   F-Value:  2981.192
##   P-Value:  0
##
## Feature:  duration_min
##   F-Value:  84.79697
##   P-Value:  3.621562e-20
##
## Feature:  is_explicit
##   F-Value:  12.7333
##   P-Value:  0.0003600364
##
## Feature:  mode
##   F-Value:  21.80158
##   P-Value:  3.04301e-06
##
## Feature:  danceability
##   F-Value:  2.551796
##   P-Value:  0.1101847
##
## Feature:  energy
##   F-Value:  43.18045
##   P-Value:  5.113799e-11
##
## Feature:  loudness
##   F-Value:  473.1953
##   P-Value:  1.00861e-103
##
## Feature:  speechiness
##   F-Value:  208.1548
##   P-Value:  5.95102e-47
##
## Feature:  acousticness
##   F-Value:  24.94027
##   P-Value:  5.962827e-07
##
## Feature:  instrumentalness
##   F-Value:  153.7396
##   P-Value:  3.548328e-35
##
## Feature:  liveness
##   F-Value:  8.51577
##   P-Value:  0.003524712
##
## Feature:  valence
##   F-Value:  57.7071
##   P-Value:  3.174639e-14
##
## Feature:  tempo
##   F-Value:  4.276451
```

```
##   P-Value:  0.03865669
##
## Feature:  key
##   F-Value:  1.472264
##   P-Value:  0.1341509
##
## Feature:  time_signature
##   F-Value:  5.716177
##   P-Value:  0.0001354902
```

```
# Create a data frame for variance importance based on ANOVA (using F-value as a measure)
anova_importance <- data.frame(
  Feature = sapply(anova_results, function(x) x$Feature), # Extract Feature names.
  F_Value = as.numeric(f_values),
  P_Value = as.numeric(p_values)
)

# Sort by F-value (higher F-value suggests higher variance explained)
anova_importance <- anova_importance %>% arrange(desc(F_Value))

# Print the table
print("ANOVA Variance Importance (Combined):")
```

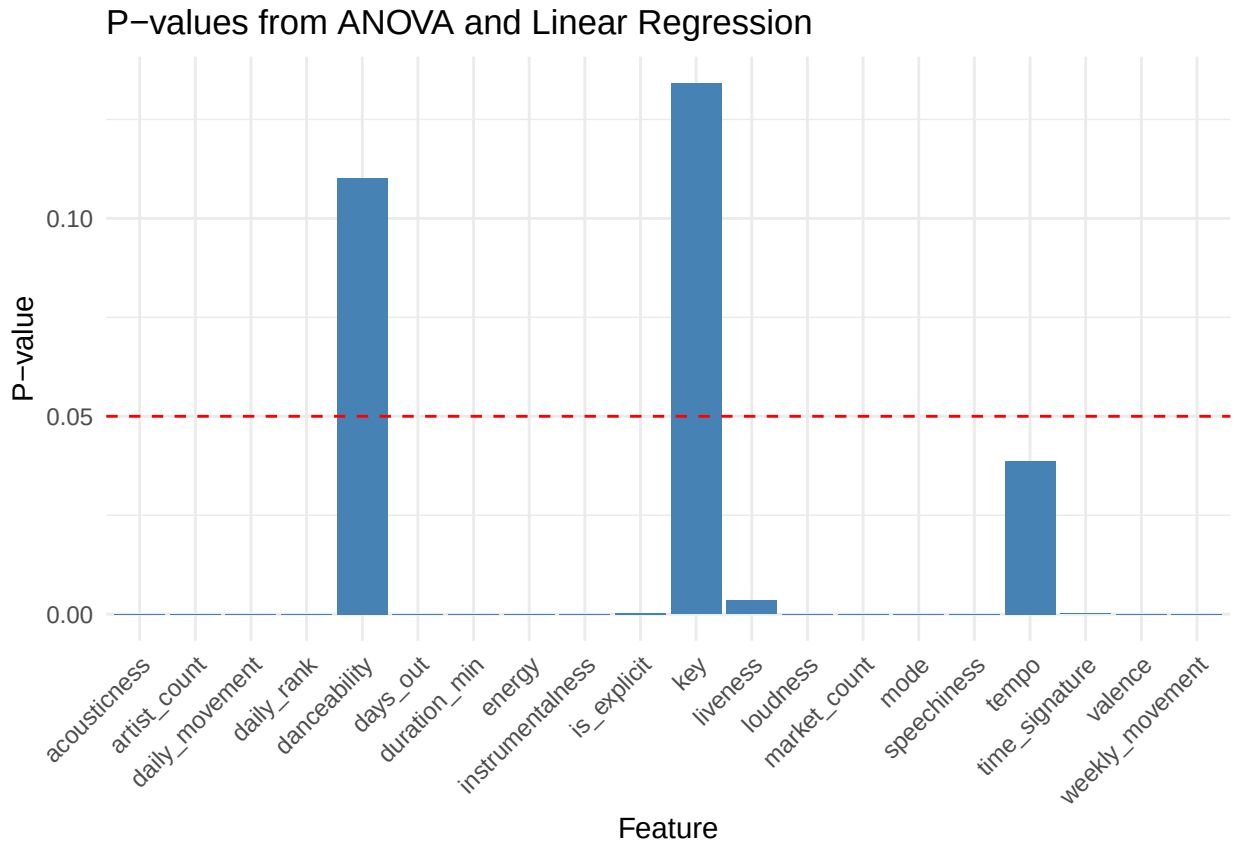
```
## [1] "ANOVA Variance Importance (Combined):"
```

```
print(anova_importance)
```

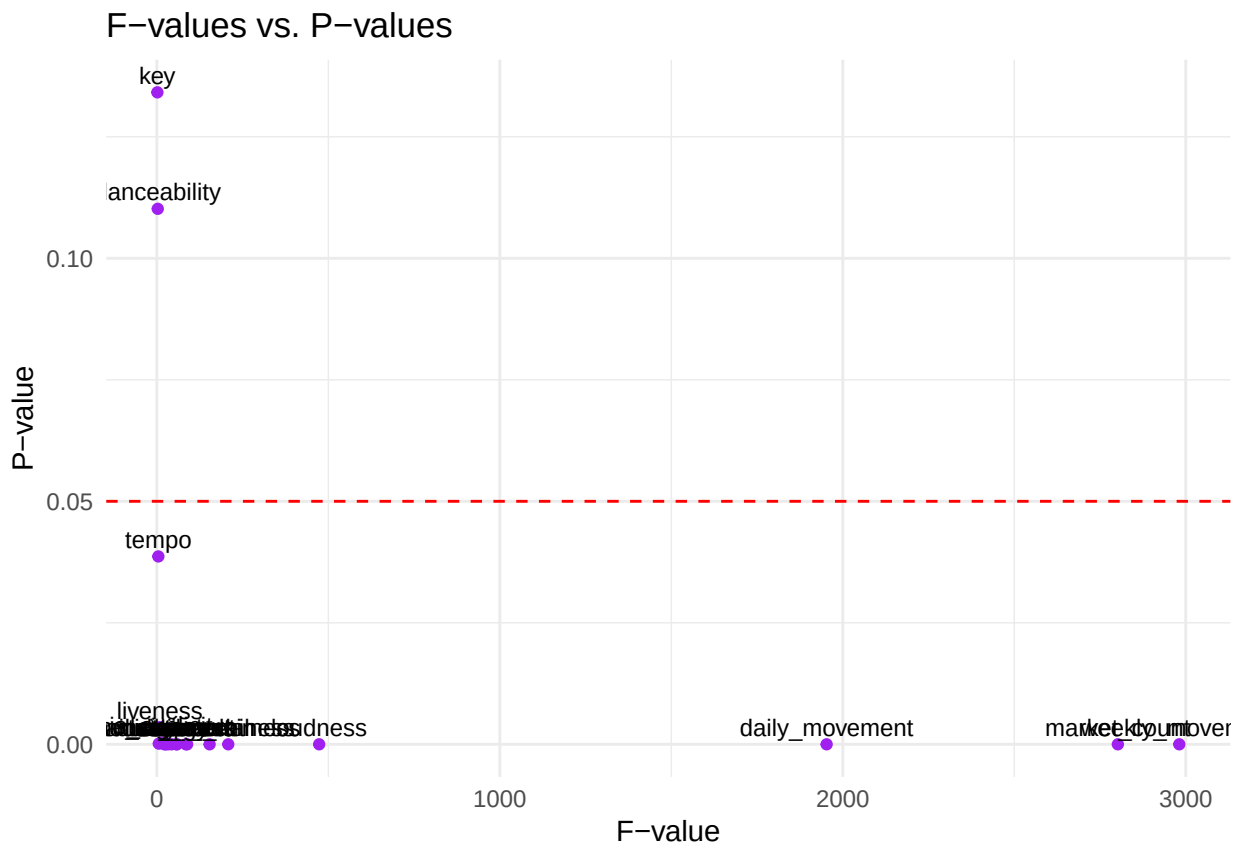
```
##             Feature     F_Value       P_Value
## 1   weekly_movement 2981.192255  0.000000e+00
## 2      market_count 2802.118963  0.000000e+00
## 3    daily_movement 1952.948151  0.000000e+00
## 4          loudness  473.195310 1.008610e-103
## 5       speechiness  208.154832  5.951020e-47
## 6  instrumentalness  153.739611  3.548328e-35
## 7          days_out   88.152871  6.684649e-21
## 8      duration_min   84.796971  3.621562e-20
## 9        daily_rank   57.949135  2.808079e-14
## 10          valence   57.707096  3.174639e-14
## 11           energy   43.180449  5.113799e-11
## 12     artist_count   32.156257  1.442066e-08
## 13     acousticness   24.940274  5.962827e-07
## 14             mode   21.801575  3.043010e-06
## 15      is_explicit   12.733300  3.600364e-04
## 16         liveness    8.515770  3.524712e-03
## 17   time_signature    5.716177  1.354902e-04
## 18            tempo    4.276451  3.865669e-02
## 19     danceability    2.551796  1.101847e-01
## 20              key    1.472264  1.341509e-01
```

```
# --- Plotting ---

# 1. Bar plot of p-values
```

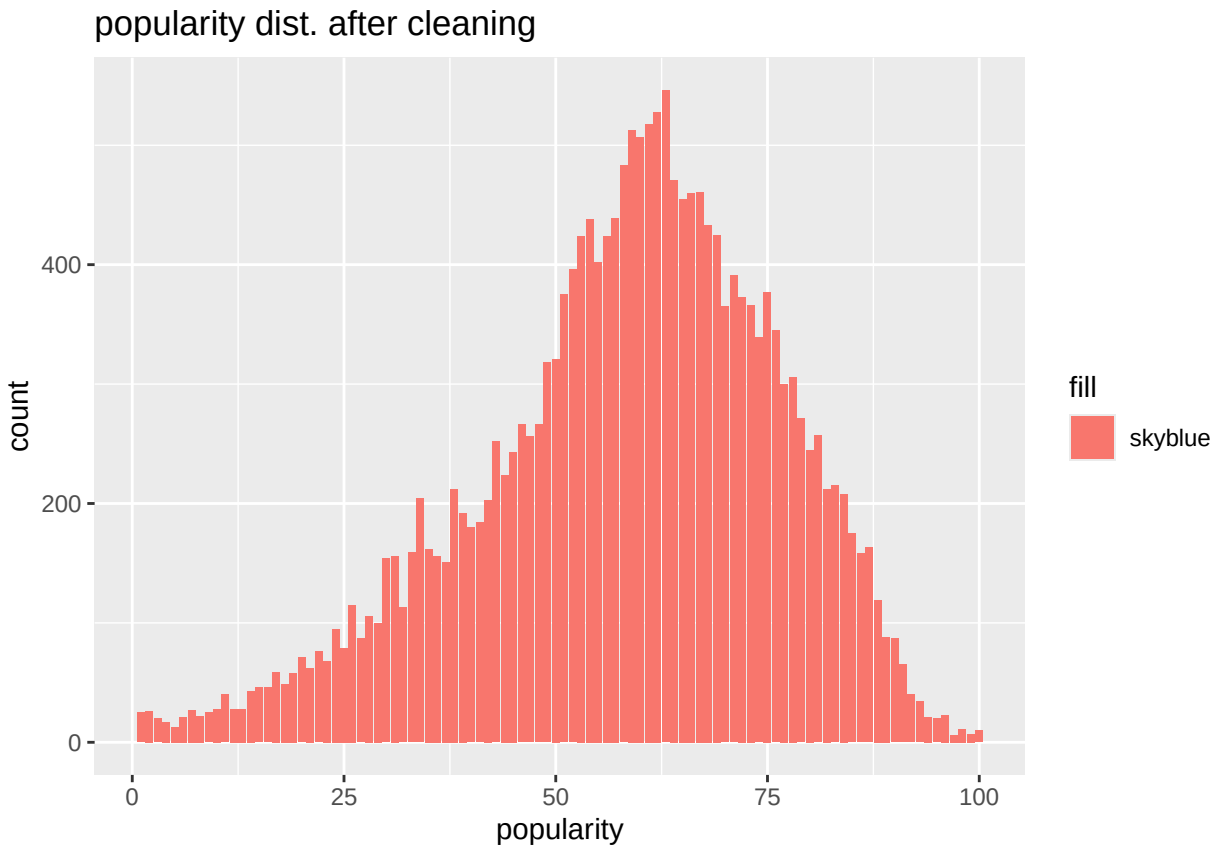
```r
print(ggplot(anova_importance, aes(x = Feature, y = P_Value)) +
        geom_bar(stat = "identity", fill = "steelblue") +
        geom_hline(yintercept = 0.05, color = "red", linetype = "dashed") + # Add significance line
        labs(title = "P-values from ANOVA and Linear Regression",
             x = "Feature",
             y = "P-value") +
        theme_minimal() +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))) # Rotate x-axis labels
```




```r
# 2.  Scatter plot of F-values vs. p-values
print(ggplot(anova_importance, aes(x = F_Value, y = P_Value, label = Feature)) +
        geom_point(color = "purple") +
        geom_text(vjust = -0.5, size = 3) +  # Add labels for the features
        geom_hline(yintercept = 0.05, color = "red", linetype = "dashed") +
        labs(title = "F-values vs. P-values",
             x = "F-value",
             y = "P-value") +
        theme_minimal())
```

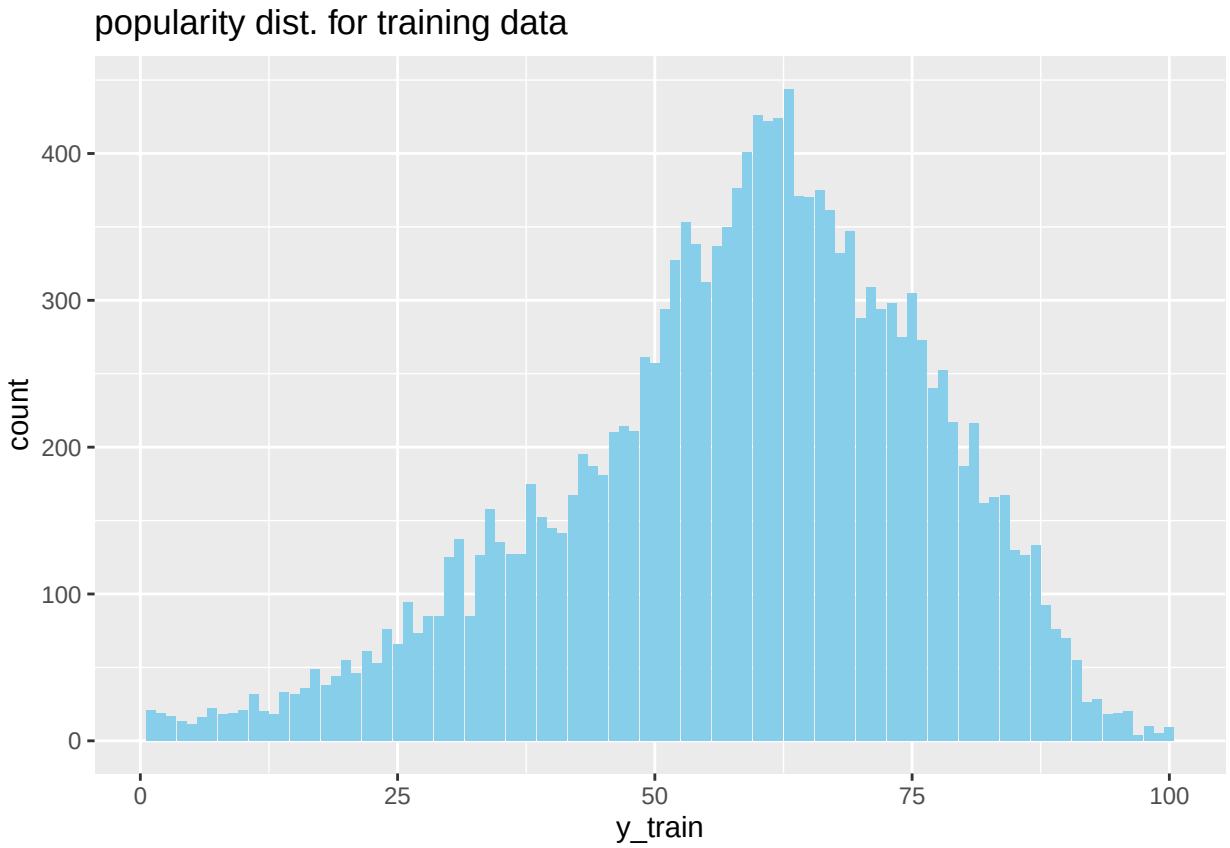



```
# check popularity distribution
popularity_dist_plot <- ggplot(data=spotify_modelling)+
  geom_bar(mapping=aes(x=popularity, fill = "skyblue"))+
  ggtitle('popularity dist. after cleaning')
print(popularity_dist_plot)
```

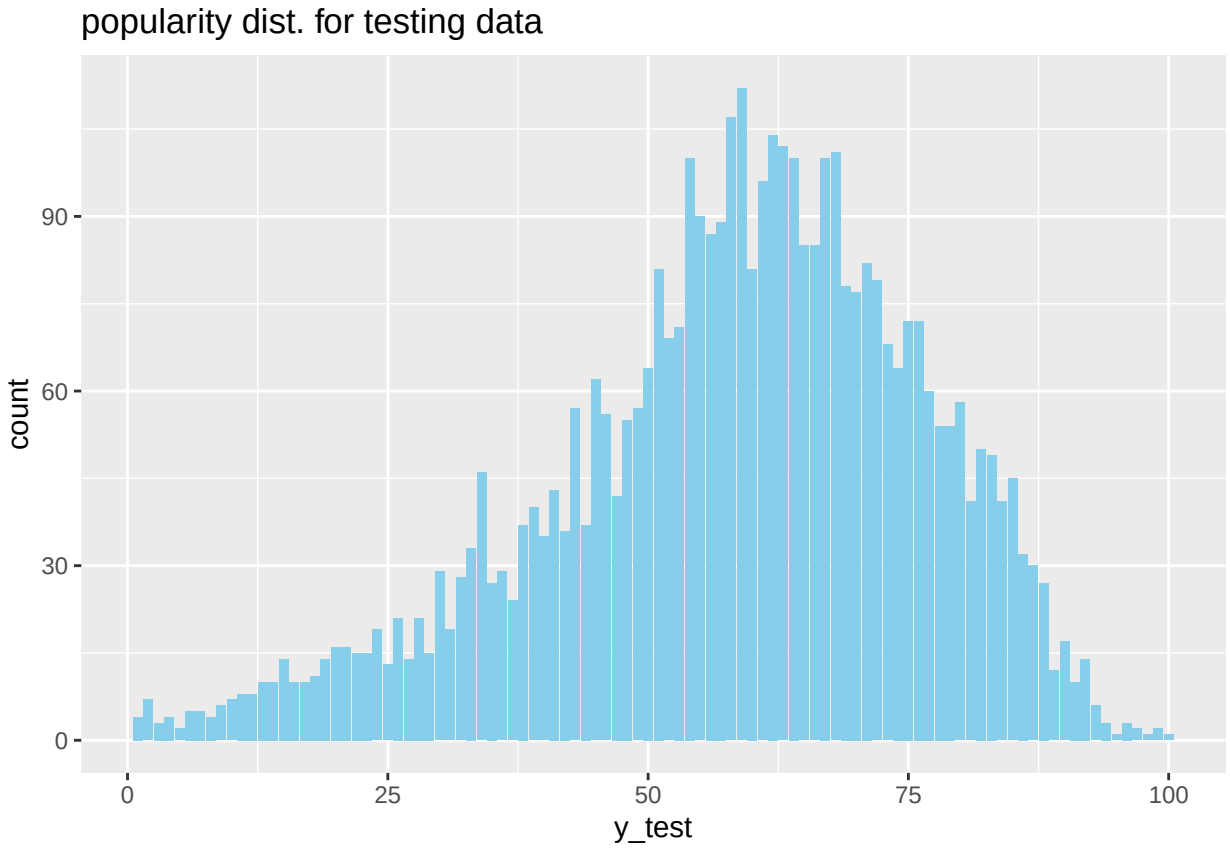



```
#--------------------------
### CARET DATA PARTITION
#--------------------------

# Define feature columns and target variable
target_column <- "popularity"

X <- spotify_modelling %>% select(-popularity)
y <- spotify_modelling$popularity

# Split the data into training and testing sets
set.seed(50)
trainIndex <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[trainIndex, ]
X_test <- X[-trainIndex, ]
y_train <- y[trainIndex]
y_test <- y[-trainIndex]

train.dframe_features <- as.data.frame(X_train)
test.dframe_features <- as.data.frame(X_test)
train.dframe_target <- as.data.frame(y_train)
test.dframe_target <- as.data.frame(y_test)
train.dframe <- cbind(train.dframe_features, y_train)
test.dframe <- cbind(test.dframe_features, y_test)

# testing popularity distribution for training data
```

```
print(ggplot(data=train.dframe)+
        geom_bar(mapping=aes(x=y_train),fill = "skyblue")+
        ggtitle('popularity dist. for training data'))
```




```
# testing popularity distribution for testing data
print(ggplot(data=test.dframe)+
        geom_bar(mapping=aes(x=y_test) ,fill = "skyblue")+
        ggtitle('popularity dist. for testing data'))
```




```
#-----------------------------------
## DATA PREPARATION FOR MODELING
#-----------------------------------

# Create a preprocessing recipe
preprocess_recipe <- recipe(x = train.dframe_features,
                            y = y_train) %>%
  # Center and scale numeric predictors
  step_center(all_numeric(), -all_outcomes()) %>%
  step_scale(all_numeric(), -all_outcomes()) %>%
  # Handle categorical variables
  step_dummy(all_nominal(), -all_outcomes())

# Apply preprocessing
prep_recipe <- prep(preprocess_recipe, training = train.dframe_features, y = y_train)
train_processed <- bake(prep_recipe, new_data = train.dframe_features)
test_processed <- bake(prep_recipe, new_data = test.dframe_features)

cat("\nProcessed Training Data (First few rows):\n")
```

```
##
## Processed Training Data (First few rows):
```

```
print(head(train_processed))
```

```
## # A tibble: 6 x 33
##   days_out artist_count market_count daily_rank daily_movement weekly_movement
##      <dbl>        <dbl>        <dbl>      <dbl>          <dbl>           <dbl>
## 1    3.56        -0.628        12.6       -1.42         -1.02           -0.965
## 2    0.257       -0.628         8.04      -2.12         -0.187          -0.287
## 3   -0.285        0.376        13.8       -2.40         -0.187          -0.219
## 4   -0.312       -0.628        12.2       -2.33         -0.187          -0.219
## 5   -0.134       -0.628         8.66      -1.49         -0.270          -0.626
## 6   -0.288       -0.628         9.28      -2.05         -0.354          -0.219
## # i 27 more variables: duration_min <dbl>, danceability <dbl>, energy <dbl>,
## #   loudness <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   is_explicit_X1 <dbl>, mode_X1 <dbl>, key_X1 <dbl>, key_X2 <dbl>,
## #   key_X3 <dbl>, key_X4 <dbl>, key_X5 <dbl>, key_X6 <dbl>, key_X7 <dbl>,
## #   key_X8 <dbl>, key_X9 <dbl>, key_X10 <dbl>, key_X11 <dbl>,
## #   time_signature_X1 <dbl>, time_signature_X3 <dbl>, ...
```

```
cat("\nProcessed Testing Data (First few rows):\n")
```

```
##
## Processed Testing Data (First few rows):
```

```
print(head(test_processed))
```

```
## # A tibble: 6 x 33
##   days_out artist_count market_count daily_rank daily_movement weekly_movement
##      <dbl>        <dbl>        <dbl>      <dbl>          <dbl>           <dbl>
## 1   -0.326       -0.628        11.6       -1.84         -0.270          -0.490
## 2   -0.309       -0.628         6.80      -1.49         -0.104          -0.355
## 3   -0.330        0.376        13.4       -2.33          3.81            3.04
## 4    7.58        -0.628        11.1       -1.28         -1.27           -0.830
## 5   -0.285       -0.628         6.38      -2.05         -0.187          -0.219
## 6   -0.331        3.39          7.21      -2.05         -0.270          -0.355
## # i 27 more variables: duration_min <dbl>, danceability <dbl>, energy <dbl>,
## #   loudness <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   is_explicit_X1 <dbl>, mode_X1 <dbl>, key_X1 <dbl>, key_X2 <dbl>,
## #   key_X3 <dbl>, key_X4 <dbl>, key_X5 <dbl>, key_X6 <dbl>, key_X7 <dbl>,
## #   key_X8 <dbl>, key_X9 <dbl>, key_X10 <dbl>, key_X11 <dbl>,
## #   time_signature_X1 <dbl>, time_signature_X3 <dbl>, ...
```

```
# Combine processed features with the target variable
train_processed <- cbind(train_processed, y_train)
test_processed <- cbind(test_processed, y_test)
colnames(train_processed)[ncol(train_processed)] <- "y_train"
colnames(test_processed)[ncol(test_processed)] <- "y_test"

#--------------------------
### RANDOM FOREST MODEL
#--------------------------

# Hyperparameter tuning using caret
```

```
control_rf <- trainControl(method = "cv", number = 5)
grid_rf <- expand.grid(mtry = c(5, 7, 9, 11),
                       min.node.size = c(3, 5, 7),
                       splitrule = "variance")
rf_model <- train(y_train ~ .,
                  data = train_processed,
                  method = "ranger",
                  trControl = control_rf,
                  tuneGrid = grid_rf,
                  importance = "permutation",
                  num.trees = 200)
# model results
print(rf_model$results)
```

```
##    mtry min.node.size splitrule     RMSE  Rsquared      MAE    RMSESD
## 1     5             3  variance 11.31741 0.6087047 8.682438 0.1533747
## 2     5             5  variance 11.30941 0.6087791 8.666364 0.1519314
## 3     5             7  variance 11.32941 0.6080022 8.688625 0.1500166
## 4     7             3  variance 11.21107 0.6112726 8.553292 0.1504125
## 5     7             5  variance 11.20971 0.6114507 8.547238 0.1701741
## 6     7             7  variance 11.23612 0.6096016 8.569745 0.1497400
## 7     9             3  variance 11.19606 0.6103843 8.502163 0.1772835
## 8     9             5  variance 11.17621 0.6119373 8.494919 0.1601108
## 9     9             7  variance 11.19028 0.6109119 8.507355 0.1441013
## 10   11             3  variance 11.18819 0.6101414 8.481995 0.1478997
## 11   11             5  variance 11.18911 0.6099901 8.490066 0.1729961
## 12   11             7  variance 11.19160 0.6099564 8.484104 0.1662319
##    RsquaredSD      MAESD
## 1  0.01495114 0.08918882
## 2  0.01429449 0.08350844
## 3  0.01472301 0.08181081
## 4  0.01436682 0.08255941
## 5  0.01594832 0.09283641
## 6  0.01517465 0.08492083
## 7  0.01654380 0.09960232
## 8  0.01579665 0.08570779
## 9  0.01414051 0.07726192
## 10 0.01464249 0.06967750
## 11 0.01628250 0.09467098
## 12 0.01581071 0.09330359
```

```
print(rf_model$bestTune)
```

```
##   mtry splitrule min.node.size
## 8    9  variance             5
```

```
importance_matrix_rf <- varImp(rf_model)

### save RF model
saveRDS(rf_model, "rf_model.rds")

# Predictions on test set
```

```r
rf_pred <- predict(rf_model, newdata= test_processed )

# Evaluate model performance
MAE_rf <- mean(abs(rf_pred - test_processed$y_test))
RMSE_rf <- sqrt(mean((rf_pred - test_processed$y_test)^2))
R_squared_rf <- cor(rf_pred, test_processed$y_test)^2

cat("Random Forest - Mean Absolute Error (MAE):", MAE_rf, "\n")
```

```
## Random Forest - Mean Absolute Error (MAE): 8.720291
```

```r
cat("Random Forest - Root Mean Squared Error (RMSE):", RMSE_rf, "\n")
```

```
## Random Forest - Root Mean Squared Error (RMSE): 11.41999
```

```r
cat("Random Forest - R-squared (R²):", R_squared_rf, "\n")
```

```
## Random Forest - R-squared (R²): 0.6031902
```

```r
# Convert the importance matrix to a data frame for easier handling
importance_df_rf <- data.frame(
  Feature = rownames(importance_matrix_rf$importance),
  Importance = importance_matrix_rf$importance[, 1] # Access the first column of importance
)
# sort by importance:
sorted_importance_df_rf <- importance_df_rf[order(importance_df_rf$Importance, decreasing = TRUE), ]
print(sorted_importance_df_rf)
```

```
##              Feature   Importance
## 1           days_out 100.00000000
## 3       market_count  82.23188596
## 6    weekly_movement  29.05875652
## 5     daily_movement  27.68036883
## 10          loudness  11.77845298
## 4         daily_rank  10.01976626
## 9             energy   4.81855821
## 12      acousticness   3.72502449
## 11       speechiness   3.65388251
## 7       duration_min   3.34145756
## 13  instrumentalness   3.13953705
## 15           valence   3.08574765
## 8       danceability   2.27976589
## 17    is_explicit_X1   2.17363043
## 16             tempo   1.14602062
## 2       artist_count   0.93562604
## 14          liveness   0.71400215
## 18           mode_X1   0.34379854
## 32 time_signature_X4   0.22513353
## 31 time_signature_X3   0.22219356
## 25            key_X7   0.22120696
## 29           key_X11   0.16922841
```

```
## 28           key_X10   0.14791528
## 26            key_X8   0.13631648
## 20            key_X2   0.09424799
## 19            key_X1   0.09337529
## 33 time_signature_X5   0.08423343
## 30 time_signature_X1   0.07821832
## 24            key_X6   0.07543584
## 23            key_X5   0.03869108
## 22            key_X4   0.03100982
## 27            key_X9   0.01553417
## 21            key_X3   0.00000000
```

```r
#--------------------------
### XGBOOST MODEL
#--------------------------

# Prepare data for XGBoost
dtrain <- xgb.DMatrix(data = as.matrix(train_processed %>% select(-y_train)), label = train_processed$y
dtest <- xgb.DMatrix(data = as.matrix(test_processed %>% select(-y_test)), label = test_processed$y_tes

# Hyperparameter tuning using caret
control_xgb <- trainControl(method = "cv", number = 5)
grid_xgb <- expand.grid(nrounds = c(100, 200),
                        max_depth = c(3, 5, 7),
                        eta = c(0.01, 0.05, 0.1),
                        gamma = 0,
                        colsample_bytree = c(0.7, 0.9),
                        min_child_weight = 1,
                        subsample = 0.8)

xgb_model <- train(y_train ~ .,
                   data = train_processed,
                   method = "xgbTree",
                   trControl = control_xgb,
                   tuneGrid = grid_xgb,
                   verbose = FALSE)
```

```
## [20:29:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:18] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:40] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:49] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:29:58] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [20:30:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:22] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:41] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:43] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:30:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:09] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:39] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:44] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:45] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:31:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:04] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:05] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:21] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:23] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:51] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:52] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:54] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:32:57] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:06] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:11] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:12] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:14] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:17] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```
## [20:33:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:38] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:42] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:47] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:48] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:50] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:53] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:33:56] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:00] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:07] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:08] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:10] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:13] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:16] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:20] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
## [20:34:24] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range` instea
```

```r
# Model results
print(xgb_model$results)
```

```
##     eta max_depth gamma colsample_bytree min_child_weight subsample nrounds
## 1  0.01         3     0              0.7                1       0.8     100
## 3  0.01         3     0              0.9                1       0.8     100
## 13 0.05         3     0              0.7                1       0.8     100
## 15 0.05         3     0              0.9                1       0.8     100
## 25 0.10         3     0              0.7                1       0.8     100
## 27 0.10         3     0              0.9                1       0.8     100
## 5  0.01         5     0              0.7                1       0.8     100
## 7  0.01         5     0              0.9                1       0.8     100
## 17 0.05         5     0              0.7                1       0.8     100
## 19 0.05         5     0              0.9                1       0.8     100
## 29 0.10         5     0              0.7                1       0.8     100
## 31 0.10         5     0              0.9                1       0.8     100
## 9  0.01         7     0              0.7                1       0.8     100
## 11 0.01         7     0              0.9                1       0.8     100
## 21 0.05         7     0              0.7                1       0.8     100
## 23 0.05         7     0              0.9                1       0.8     100
## 33 0.10         7     0              0.7                1       0.8     100
## 35 0.10         7     0              0.9                1       0.8     100
## 2  0.01         3     0              0.7                1       0.8     200
## 4  0.01         3     0              0.9                1       0.8     200
## 14 0.05         3     0              0.7                1       0.8     200
## 16 0.05         3     0              0.9                1       0.8     200
## 26 0.10         3     0              0.7                1       0.8     200
## 28 0.10         3     0              0.9                1       0.8     200
## 6  0.01         5     0              0.7                1       0.8     200
## 8  0.01         5     0              0.9                1       0.8     200
## 18 0.05         5     0              0.7                1       0.8     200
## 20 0.05         5     0              0.9                1       0.8     200
## 30 0.10         5     0              0.7                1       0.8     200
## 32 0.10         5     0              0.9                1       0.8     200
```

```
## 10 0.01         7     0              0.7                1       0.8     200
## 12 0.01         7     0              0.9                1       0.8     200
## 22 0.05         7     0              0.7                1       0.8     200
## 24 0.05         7     0              0.9                1       0.8     200
## 34 0.10         7     0              0.7                1       0.8     200
## 36 0.10         7     0              0.9                1       0.8     200
##        RMSE  Rsquared       MAE    RMSESD RsquaredSD     MAESD
## 1  25.08267 0.5614223 22.315173 0.1036646 0.02346227 0.1038213
## 3  24.98913 0.5513985 22.277689 0.1252953 0.02228306 0.1163378
## 13 11.44356 0.5941940  8.777263 0.3265949 0.02196189 0.1937403
## 15 11.44810 0.5935497  8.771578 0.3315982 0.02204443 0.1880701
## 25 11.30509 0.6014146  8.627201 0.3335187 0.02144497 0.1965909
## 27 11.29564 0.6019699  8.606335 0.3157380 0.01976132 0.1806094
## 5  24.88158 0.5915617 22.229773 0.1270046 0.02078002 0.1107946
## 7  24.75780 0.5869052 22.171526 0.1086377 0.02049623 0.1012651
## 17 11.21776 0.6086652  8.564971 0.3141791 0.01932546 0.1824858
## 19 11.20976 0.6089635  8.546383 0.3098237 0.01884367 0.1708822
## 29 11.18269 0.6096980  8.502762 0.3108778 0.01900701 0.1950666
## 31 11.17960 0.6098329  8.483572 0.3195408 0.01976509 0.1992144
## 9  24.88159 0.5970611 22.259851 0.1300552 0.02221747 0.1191139
## 11 24.72605 0.5946238 22.188359 0.1139850 0.02102523 0.1079149
## 21 11.15668 0.6124894  8.493001 0.3128461 0.01904853 0.1843704
## 23 11.16335 0.6117886  8.486219 0.3250357 0.01986318 0.1759743
## 33 11.22588 0.6066246  8.513656 0.3168806 0.01988621 0.1622447
## 35 11.21248 0.6076185  8.494640 0.2880898 0.01772105 0.1578373
## 2  14.40267 0.5753248 11.909699 0.2088112 0.02314080 0.1361745
## 4  14.36855 0.5696173 11.886547 0.2134105 0.02232514 0.1407501
## 14 11.27678 0.6034817  8.601904 0.3348524 0.02124288 0.2025139
## 16 11.27728 0.6033817  8.595171 0.3268395 0.02079185 0.1890250
## 26 11.24943 0.6049799  8.565520 0.3275909 0.02030240 0.1878084
## 28 11.24505 0.6052577  8.551766 0.3245953 0.01999781 0.1894102
## 6  14.08020 0.5987371 11.623243 0.2161956 0.02006518 0.1288560
## 8  14.01272 0.5954853 11.559953 0.2070023 0.01999060 0.1180075
## 18 11.13355 0.6131880  8.464923 0.3111415 0.01879409 0.1830832
## 20 11.13199 0.6132065  8.456999 0.3103938 0.01842150 0.1753197
## 30 11.19047 0.6091669  8.508559 0.3014347 0.01818895 0.1806464
## 32 11.20917 0.6078466  8.511844 0.3242325 0.01960029 0.1995979
## 10 14.02405 0.6048166 11.578212 0.2207384 0.02098946 0.1279066
## 12 13.93497 0.6031786 11.492805 0.2132171 0.02002574 0.1185047
## 22 11.12250 0.6138600  8.430153 0.3142520 0.01934634 0.1859321
## 24 11.12049 0.6139865  8.421239 0.3246579 0.01969744 0.1781350
## 34 11.27727 0.6033044  8.558255 0.3179626 0.01991243 0.1673887
## 36 11.25360 0.6051478  8.533713 0.2806748 0.01749571 0.1572512
```

```
print(xgb_model$bestTune)
```

```
##    nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 24     200         7 0.05     0              0.9                1       0.8
```

```
importance_matrix_xgb <- varImp(xgb_model)

### save XGBoost model
saveRDS(xgb_model, "xgb_model.rds")
```

```r
# Predictions on test set
xgb_pred <- predict(xgb_model, newdata = test_processed %>% select(-y_test))

# Evaluate model performance
MAE_xgb <- mean(abs(xgb_pred - test_processed$y_test))
RMSE_xgb <- sqrt(mean((xgb_pred - test_processed$y_test)^2))
R_squared_xgb <- cor(xgb_pred, test_processed$y_test)^2

cat("XGBoost - Mean Absolute Error (MAE):", MAE_xgb, "\n")
```

```
## XGBoost - Mean Absolute Error (MAE): 8.61725
```

```r
cat("XGBoost - Root Mean Squared Error (RMSE):", RMSE_xgb, "\n")
```

```
## XGBoost - Root Mean Squared Error (RMSE): 11.30276
```

```r
cat("XGBoost - R-squared (R²):", R_squared_xgb, "\n")
```

```
## XGBoost - R-squared (R²): 0.6095372
```

```r
# Convert the importance matrix to a data frame for easier handling
importance_df_xgb <- data.frame(
  Feature = rownames(importance_matrix_xgb$importance),
  Importance = importance_matrix_xgb$importance[, 1] # Access the first column of importance
)
# sort by importance:
sorted_importance_df_xgb <- importance_df_xgb[order(importance_df_xgb$Importance, decreasing = TRUE), ]
print(sorted_importance_df_xgb)
```

```
##              Feature   Importance
## 1           days_out 100.00000000
## 2       market_count  66.67918379
## 3     daily_movement  26.91773616
## 4    weekly_movement  22.28768554
## 5           loudness  10.71855597
## 6         daily_rank   6.82992805
## 7       duration_min   6.68018661
## 8        speechiness   6.16983291
## 9       acousticness   6.16194133
## 10           valence   5.93968632
## 11            energy   5.88869003
## 12  instrumentalness   5.29559598
## 13             tempo   5.18319942
## 14      danceability   5.13177753
## 15          liveness   4.45198530
## 16      artist_count   1.51185259
## 17    is_explicit_X1   1.21928482
## 18           mode_X1   0.49723637
## 19            key_X6   0.26584974
## 20 time_signature_X3   0.21457095
```

```
## 21            key_X1   0.20508764
## 22            key_X7   0.19071879
## 23 time_signature_X5   0.18740539
## 24           key_X10   0.15394100
## 25            key_X4   0.14138807
## 26           key_X11   0.14078185
## 27 time_signature_X4   0.13710628
## 28            key_X5   0.11537559
## 29            key_X2   0.11360914
## 30            key_X3   0.10980091
## 31            key_X8   0.09192684
## 32            key_X9   0.07665649
## 33 time_signature_X1   0.00000000
```

```r
#--------------------------
### GBM MODEL
#--------------------------

# Hyperparameter tuning using caret
control_gbm <- trainControl(method = "cv", number = 5)
grid_gbm <- expand.grid(n.trees = c(100, 200),
                        interaction.depth = c(3, 5),
                        shrinkage = c(0.01, 0.05),
                        n.minobsinnode = c(10, 20))

gbm_model <- train(y_train ~ .,
                   data = train_processed,
                   method = "gbm",
                   trControl = control_gbm,
                   tuneGrid = grid_gbm,
                   verbose = FALSE)

# Model results
print(gbm_model$results)
```

```
##    shrinkage interaction.depth n.minobsinnode n.trees     RMSE  Rsquared
## 1       0.01                 3             10     100 13.69261 0.5183271
## 3       0.01                 3             20     100 13.69658 0.5177864
## 9       0.05                 3             10     100 11.66188 0.5785862
## 11      0.05                 3             20     100 11.66497 0.5781752
## 5       0.01                 5             10     100 13.32411 0.5424023
## 7       0.01                 5             20     100 13.32720 0.5414588
## 13      0.05                 5             10     100 11.48825 0.5899670
## 15      0.05                 5             20     100 11.49732 0.5892322
## 2       0.01                 3             10     200 12.45900 0.5477016
## 4       0.01                 3             20     200 12.46178 0.5477421
## 10      0.05                 3             10     200 11.46033 0.5907193
## 12      0.05                 3             20     200 11.46965 0.5900142
## 6       0.01                 5             10     200 12.12091 0.5663829
## 8       0.01                 5             20     200 12.11770 0.5665512
## 14      0.05                 5             10     200 11.33511 0.5993409
## 16      0.05                 5             20     200 11.34629 0.5985361
##          MAE    RMSESD RsquaredSD      MAESD
## 1  10.819545 0.2445258 0.01597329 0.12941502
```

```
## 3  10.822895 0.2444191 0.01562227 0.12830999
## 9   8.952720 0.2266751 0.01397132 0.12955901
## 11  8.955140 0.2128692 0.01327706 0.11988076
## 5  10.505068 0.2409242 0.01526074 0.13091263
## 7  10.506754 0.2444173 0.01560773 0.13346633
## 13  8.800972 0.2052313 0.01285319 0.11386821
## 15  8.797217 0.2202896 0.01360203 0.10648126
## 2   9.704980 0.2318827 0.01437035 0.12835072
## 4   9.706027 0.2322909 0.01442569 0.12530513
## 10  8.765218 0.2167217 0.01312578 0.11044286
## 12  8.773723 0.2195501 0.01337654 0.10704506
## 6   9.405186 0.2316142 0.01464210 0.12586845
## 8   9.401362 0.2252671 0.01460993 0.11935291
## 14  8.651742 0.2105999 0.01278617 0.10957588
## 16  8.649197 0.2158476 0.01342728 0.09337732
```

```r
print(gbm_model$bestTune)
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 14     200                 5      0.05             10
```

```r
importance_matrix_gbm <- varImp(gbm_model)

### save GBM model
saveRDS(gbm_model, "gbm_model.rds")

# Predictions on test set
gbm_pred <- predict(gbm_model, newdata = test_processed %>% select(-y_test))

# Evaluate model performance
MAE_gbm <- mean(abs(gbm_pred - test_processed$y_test))
RMSE_gbm <- sqrt(mean((gbm_pred - test_processed$y_test)^2))
R_squared_gbm <- cor(gbm_pred, test_processed$y_test)^2

cat("Gradient Boosting Machine - Mean Absolute Error (MAE):", MAE_gbm, "\n")
```

```
## Gradient Boosting Machine - Mean Absolute Error (MAE): 8.795771
```

```r
cat("Gradient Boosting Machine - Root Mean Squared Error (RMSE):", RMSE_gbm, "\n")
```

```
## Gradient Boosting Machine - Root Mean Squared Error (RMSE): 11.51585
```

```r
cat("Gradient Boosting Machine - R-squared (R²):", R_squared_gbm, "\n")
```

```
## Gradient Boosting Machine - R-squared (R²): 0.594686
```

```r
# Convert the importance matrix to a data frame for easier handling
importance_df_gbm <- data.frame(
  Feature = rownames(importance_matrix_gbm$importance),
  Importance = importance_matrix_gbm$importance[, 1] # Access the first column of importance
```

```
)
# sort by importance:
sorted_importance_df_gbm <- importance_df_gbm[order(importance_df_gbm$Importance, decreasing = TRUE), ]
print(sorted_importance_df_gbm)
```

```
##              Feature   Importance
## 1           days_out 100.00000000
## 3       market_count  60.32916781
## 5     daily_movement  24.32500314
## 6    weekly_movement  18.08466822
## 10          loudness   6.44984767
## 4         daily_rank   4.56820668
## 13  instrumentalness   1.89577788
## 12      acousticness   1.81177395
## 9             energy   1.29392017
## 7       duration_min   1.13547119
## 8       danceability   1.12366354
## 11       speechiness   0.88371840
## 14          liveness   0.88184070
## 16             tempo   0.78165057
## 17    is_explicit_X1   0.76786042
## 15           valence   0.64794360
## 2       artist_count   0.35920750
## 18           mode_X1   0.26832853
## 31 time_signature_X3   0.16860377
## 22            key_X4   0.02672843
## 24            key_X6   0.02074546
## 19            key_X1   0.00000000
## 20            key_X2   0.00000000
## 21            key_X3   0.00000000
## 23            key_X5   0.00000000
## 25            key_X7   0.00000000
## 26            key_X8   0.00000000
## 27            key_X9   0.00000000
## 28           key_X10   0.00000000
## 29           key_X11   0.00000000
## 30 time_signature_X1   0.00000000
## 32 time_signature_X4   0.00000000
## 33 time_signature_X5   0.00000000
```

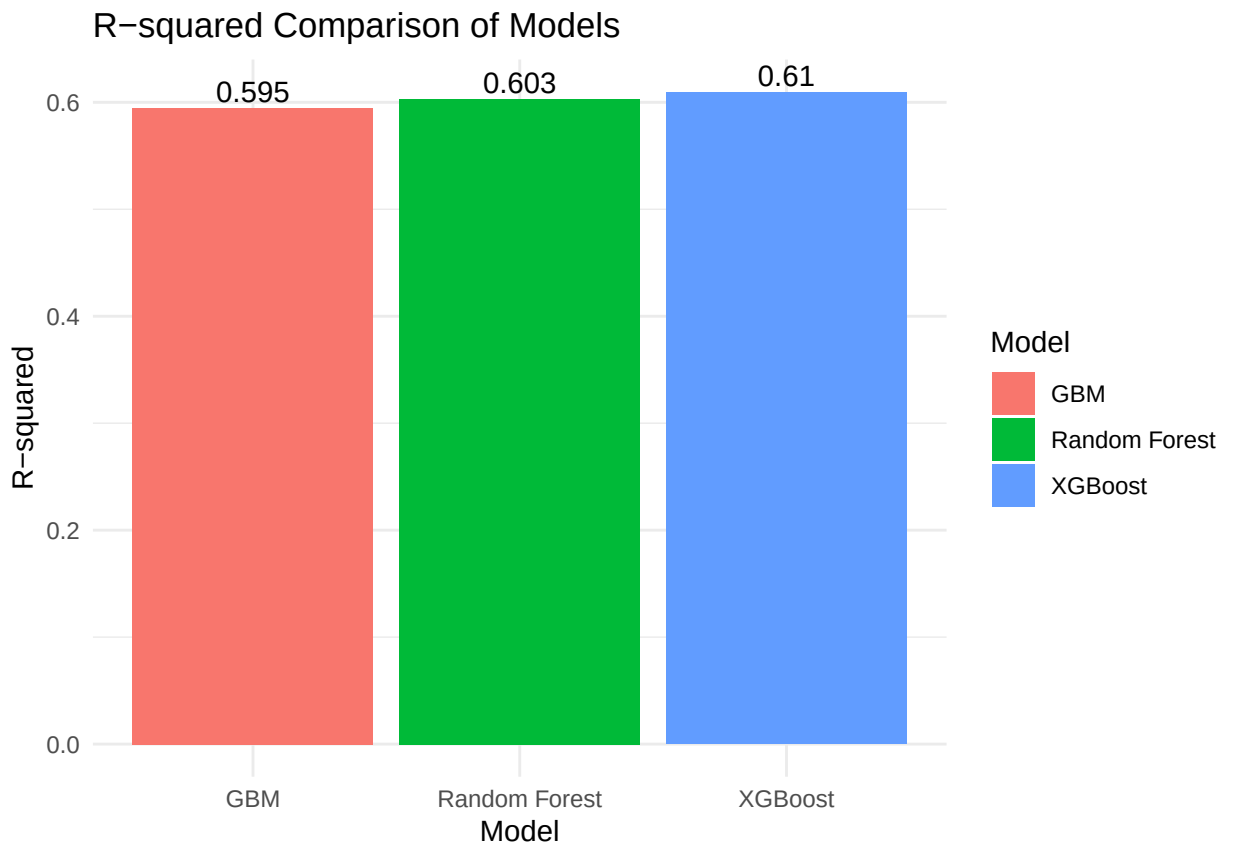
```
#------------------------
## MODEL COMPARISON:
#-------------------------

# Create a data frame to compare model performance
model_comparison <- data.frame(
  Model = c("Random Forest", "XGBoost", "GBM"),
  MAE = c(MAE_rf, MAE_xgb, MAE_gbm),
  RMSE = c(RMSE_rf, RMSE_xgb, RMSE_gbm),
  R_squared = c(R_squared_rf, R_squared_xgb, R_squared_gbm)
)

print("Model Performance Comparison:")
```

```
## [1] "Model Performance Comparison:"
```

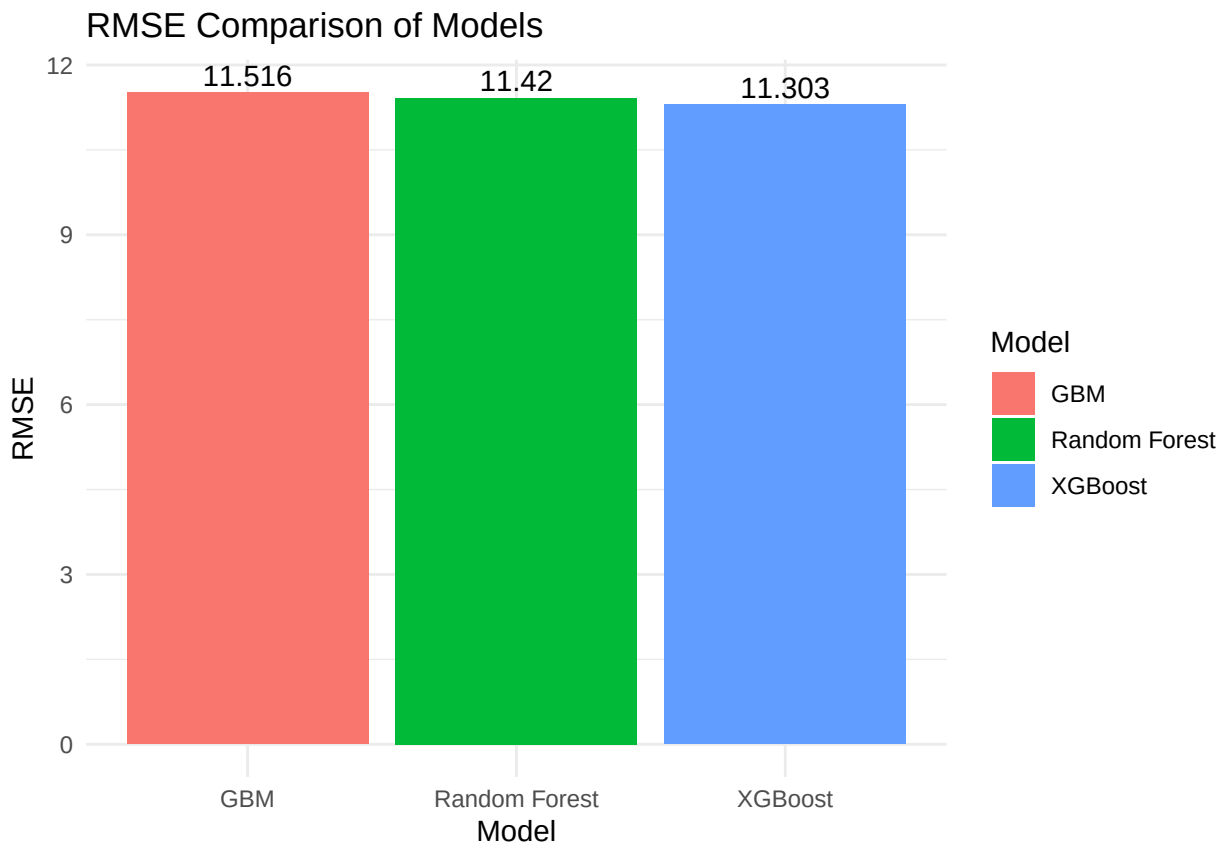
```
print(model_comparison)
```

```
##           Model      MAE     RMSE R_squared
## 1 Random Forest 8.720291 11.41999 0.6031902
## 2       XGBoost 8.617250 11.30276 0.6095372
## 3           GBM 8.795771 11.51585 0.5946860
```

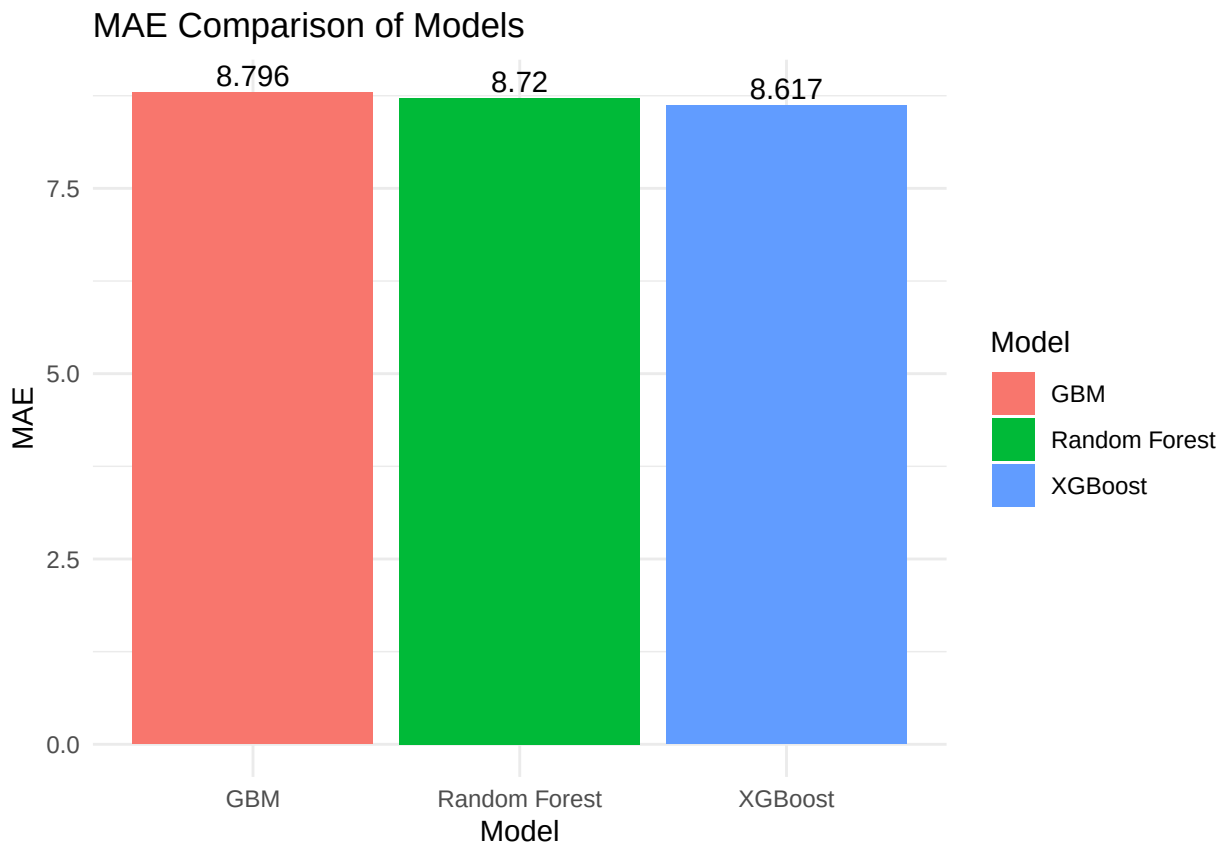
```
# Create a bar plot for R-squared comparison
r_squared_plot <- ggplot(model_comparison, aes(x = Model, y = R_squared, fill = Model)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(R_squared, 3)), vjust = -0.3) +
  labs(title = "R-squared Comparison of Models", y = "R-squared") +
  theme_minimal()
print(r_squared_plot)
```




```
# Create a bar plot for RMSE comparison
rmse_plot <- ggplot(model_comparison, aes(x = Model, y = RMSE, fill = Model)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(RMSE, 3)), vjust = -0.3) +
  labs(title = "RMSE Comparison of Models", y = "RMSE") +
  theme_minimal()
print(rmse_plot)
```

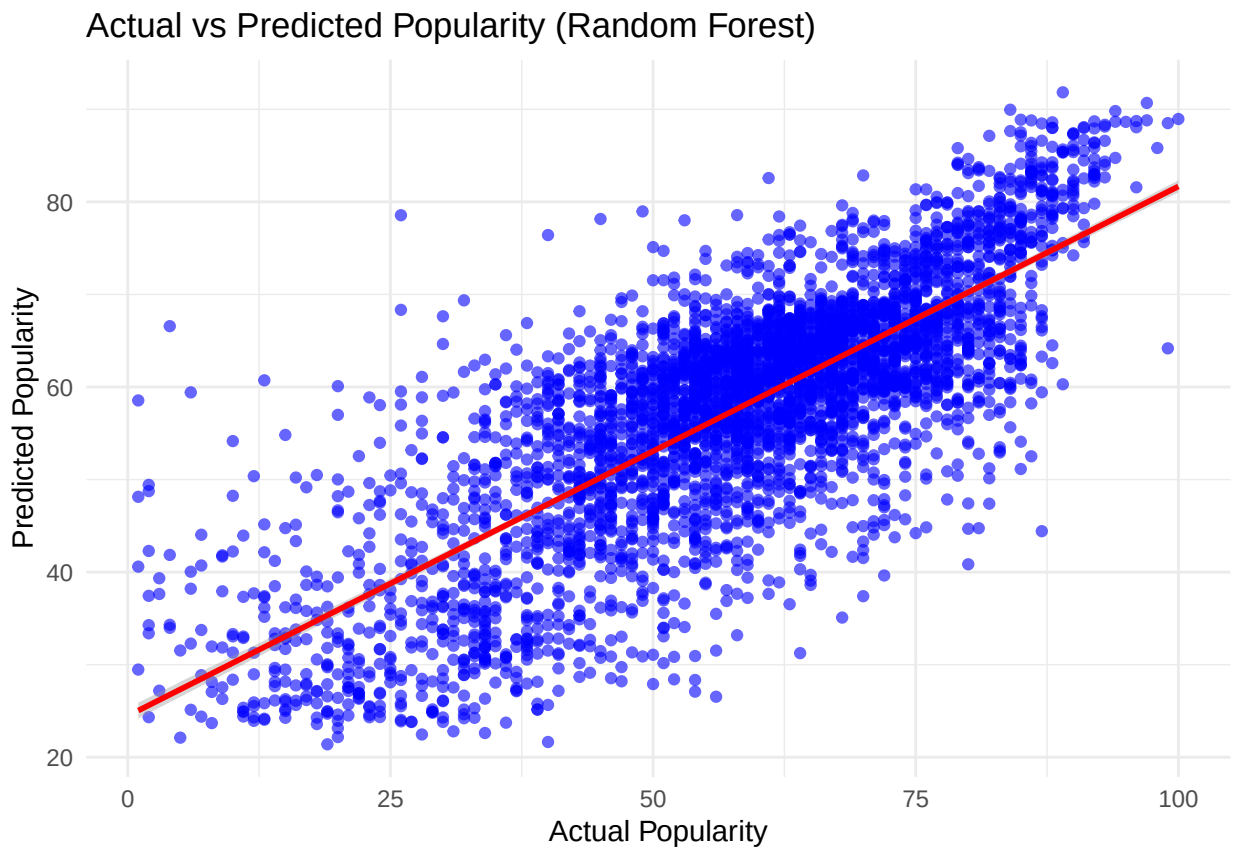



```
# Create a bar plot for MAE comparison
mae_plot <- ggplot(model_comparison, aes(x = Model, y = MAE, fill = Model)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(MAE, 3)), vjust = -0.3) +
  labs(title = "MAE Comparison of Models", y = "MAE") +
  theme_minimal()
print(mae_plot)
```



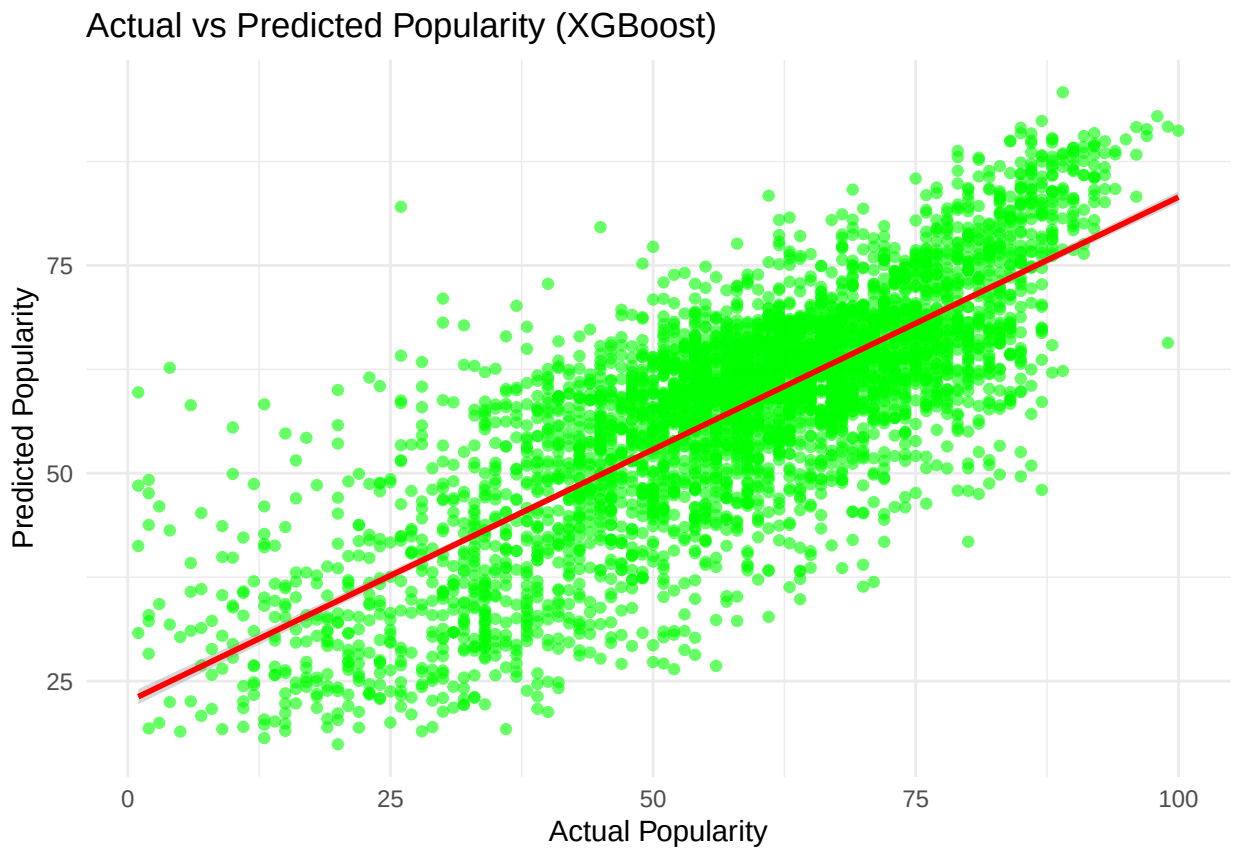

```
#-----------------------------------
## ACTUAL VS PREDICTED PLOTS:
#-----------------------------------

# Random Forest
plot_data_rf <- data.frame(Actual = test_processed$y_test, Predicted = rf_pred)
actual_predicted_plot_rf <- ggplot(plot_data_rf, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = 'lm', col='red') +
  labs(title = "Actual vs Predicted Popularity (Random Forest)",
       x = "Actual Popularity", y = "Predicted Popularity") +
  theme_minimal()
print(actual_predicted_plot_rf)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



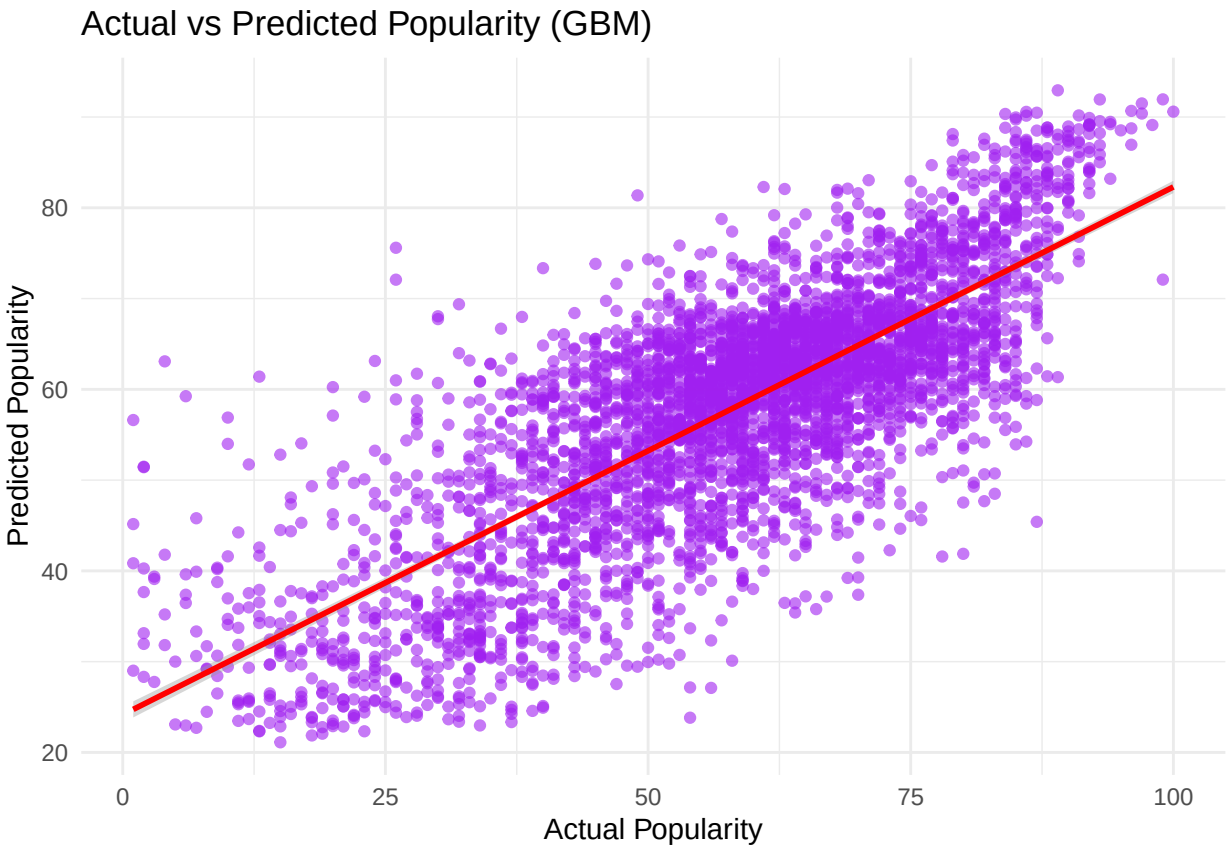

```
# XGBoost
plot_data_xgb <- data.frame(Actual = test_processed$y_test, Predicted = xgb_pred)
actual_predicted_plot_xgb <- ggplot(plot_data_xgb, aes(x = Actual, y = Predicted)) +
  geom_point(color = "green", alpha = 0.6) +
  geom_smooth(method = 'lm', col='red') +
  labs(title = "Actual vs Predicted Popularity (XGBoost)",
       x = "Actual Popularity", y = "Predicted Popularity") +
  theme_minimal()
print(actual_predicted_plot_xgb)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# GBM
plot_data_gbm <- data.frame(Actual = test_processed$y_test, Predicted = gbm_pred)
actual_predicted_plot_gbm <- ggplot(plot_data_gbm, aes(x = Actual, y = Predicted)) +
  geom_point(color = "purple", alpha = 0.6) +
  geom_smooth(method = 'lm', col='red') +
  labs(title = "Actual vs Predicted Popularity (GBM)",
       x = "Actual Popularity", y = "Predicted Popularity") +
  theme_minimal()
print(actual_predicted_plot_gbm)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```


Actual vs Predicted Popularity (GBM)
Predicted Popularity
Actual Popularity
20
40
60
80
0
25
50
75
100