A Quantitative Analysis on Musical Genre

by Derian Lee

Part 1: Introduction

Machine learning has developed tremendously over the last decade. In fact, it's gotten to the point where even art can be made by technology! This is a concept that some, including myself, find uncomfortable at times. But as someone who composes and studies music, I do see the potential and I believe ML can offer deep insights that humans cannot detect! These insights might be valuable for someone who is open improving their art, or music.

DATASETS

For this analysis, we will look at the 'spotify_songs' dataset, which was taken from TidyTuesday on GitHub. This dataset contains a tremendous amount of numerical data for different songs on Spotify, including tempo, duration, key, and much more.

Because there is so much data, this project hopes to focus on three specific musical genres: R&B, LoFi, and Tropical music. This exploratory project in ML is for the purpose of improving clustering and classification.

Specifically, I am curious about what sets these three genres apart. In addition, I am curious to see if a logistic regression model can accurately detect music that is above average in the variable *track_popularity*.

PREDICTIONS

1. Out of three genres, only two will be recognized as clusters
2. In a principle component plot, LoFi will be distinguished from other genres.
3. Loudness will be a defining variable in creating PC1!

First let's install the packages we need:

```
# Install necessary packages
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.0.4     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.0
```

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WB
a
```

```
library(cluster)
library(GGally)
```

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

```
library(ade4)
library(plotROC)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

Next, let's install our data.

```
# Install the data
spotify_songs <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidy
tuesday/master/data/2020/2020-01-21/spotify_songs.csv')
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   .default = col_double(),
##   track_id = col_character(),
##   track_name = col_character(),
##   track_artist = col_character(),
##   track_album_id = col_character(),
##   track_album_name = col_character(),
##   track_album_release_date = col_character(),
##   playlist_name = col_character(),
##   playlist_id = col_character(),
##   playlist_genre = col_character(),
##   playlist_subgenre = col_character()
## )
## ℹ Use `spec()` for the full column specifications.
```

The data is already tidy, but a little wrangling will be done before we begin.

```
# Store genres
vec <- c("Lo-Fi Beats", "New R&B   ", "Tropical Vibes")

# Extract genres
music <- filter(spotify_songs, playlist_name %in% vec)

# Rename for interpretability
music <- music %>%
  rename("genre" = playlist_name)

# Take a look at the dataset:
head(music)
```

```
## # A tibble: 6 x 23
##   track_id track_name track_artist track_popularity track_album_id
##   <chr>    <chr>      <chr>                   <dbl> <chr>
## 1 0rNpm25… With U     SwuM                       64 5YuMyydKScBvK…
## 2 4c2TiYo… sekao      Delayde                    63 2yTJ6fdaX9ZYG…
## 3 2tVHXIM… Aconcagua  Slumberville               59 480XGKz77Nqgc…
## 4 2pZi2b9… thinking … mommy                      64 6mOgPOHFf2JTZ…
## 5 66z8EZX… Crossroads Hanz                       59 3u1k0CIc2zQdX…
## 6 74t6wFV… memories … Rook1e                     53 7EIbQnhNrS9Tb…
## # … with 18 more variables: track_album_name <chr>,
## #   track_album_release_date <chr>, genre <chr>, playlist_id <chr>,
## #   playlist_genre <chr>, playlist_subgenre <chr>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, duration_ms <dbl>
```

Let’s begin!

Part 2: Exploratory Data Analysis

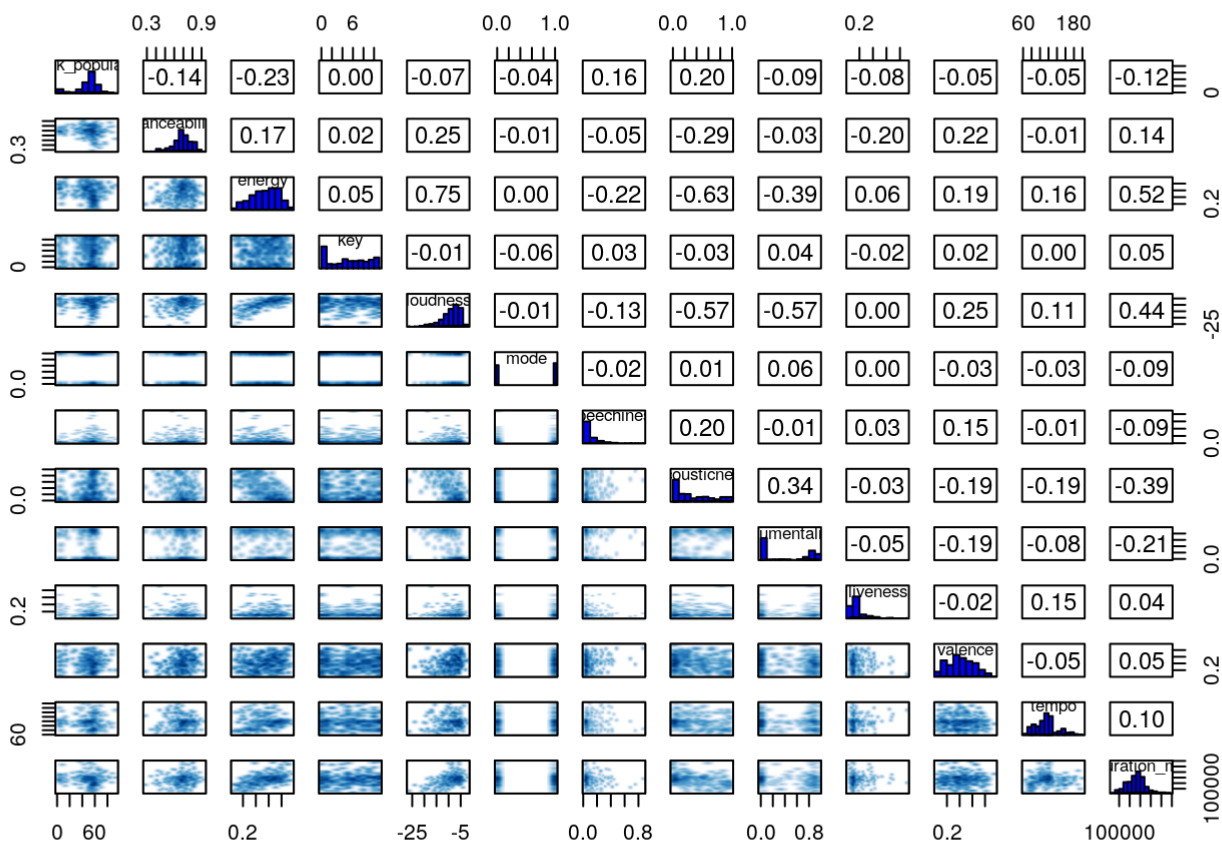
```
# Select numeric variables only
music_num <- music %>%
  select_if(is.numeric)

# Build correlation matrix
cor(music_num, use = "pairwise.complete.obs")
```

```
##                  track_popularity danceability      energy          key
## track_popularity     1.0000000000 -0.143703388 -0.232999776  0.0003771264
## danceability        -0.1437033881  1.000000000  0.171528630  0.0226569550
## energy              -0.2329997764  0.171528630  1.000000000  0.0469002086
## key                  0.0003771264  0.022656955  0.046900209  1.0000000000
## loudness            -0.0684976785  0.254583655  0.749146318 -0.0109955218
## mode                -0.0392236948 -0.009881735  0.001939268 -0.0599477851
## speechiness          0.1601168475 -0.047590930 -0.221038122  0.0269599901
## acousticness         0.1980195166 -0.286350591 -0.634296626 -0.0312489259
## instrumentalness    -0.0856024993 -0.027847584 -0.389714012  0.0372633463
## liveness            -0.0791915358 -0.197776845  0.058633065 -0.0189272822
## valence             -0.0504273567  0.218739723  0.194792673  0.0173884955
## tempo               -0.0505088408 -0.014258534  0.158216734  0.0016630832
## duration_ms         -0.1200615797  0.136781979  0.518059114  0.0501108968
##                      loudness         mode  speechiness acousticness
## track_popularity -0.068497678 -0.039223695  0.160116848  0.198019517
## danceability      0.254583655 -0.009881735 -0.047590930 -0.286350591
## energy            0.749146318  0.001939268 -0.221038122 -0.634296626
## key              -0.010995522 -0.059947785  0.026959990 -0.031248926
## loudness          1.000000000 -0.007908898 -0.130331064 -0.567390042
## mode             -0.007908898  1.000000000 -0.020873881  0.006008775
## speechiness      -0.130331064 -0.020873881  1.000000000  0.196722191
## acousticness     -0.567390042  0.006008775  0.196722191  1.000000000
## instrumentalness -0.573919331  0.057716136 -0.007321569  0.335854672
## liveness          0.003518848  0.003860230  0.033159794 -0.027625152
## valence           0.250976766 -0.026430986  0.152848724 -0.190174481
## tempo             0.109514631 -0.028253607 -0.011682599 -0.194807526
## duration_ms       0.443821695 -0.090963504 -0.094892898 -0.393541478
##                  instrumentalness     liveness     valence        tempo
## track_popularity     -0.085602499 -0.079191536 -0.05042736 -0.050508841
## danceability         -0.027847584 -0.197776845  0.21873972 -0.014258534
## energy               -0.389714012  0.058633065  0.19479267  0.158216734
## key                   0.037263346 -0.018927282  0.01738850  0.001663083
## loudness             -0.573919331  0.003518848  0.25097677  0.109514631
## mode                  0.057716136  0.003860230 -0.02643099 -0.028253607
## speechiness          -0.007321569  0.033159794  0.15284872 -0.011682599
## acousticness          0.335854672 -0.027625152 -0.19017448 -0.194807526
## instrumentalness      1.000000000 -0.053777702 -0.18691047 -0.079121047
## liveness             -0.053777702  1.000000000 -0.02346679  0.152295534
## valence              -0.186910469 -0.023466792  1.00000000 -0.047198260
## tempo                -0.079121047  0.152295534 -0.04719826  1.000000000
## duration_ms          -0.209352772  0.040715957  0.04727652  0.103055162
##                  duration_ms
## track_popularity -0.12006158
## danceability      0.13678198
## energy            0.51805911
## key               0.05011090
## loudness          0.44382170
## mode             -0.09096350
## speechiness      -0.09489290
## acousticness     -0.39354148
## instrumentalness -0.20935277
```

```
## liveness          0.04071596
## valence           0.04727652
## tempo             0.10305516
## duration_ms       1.00000000
```

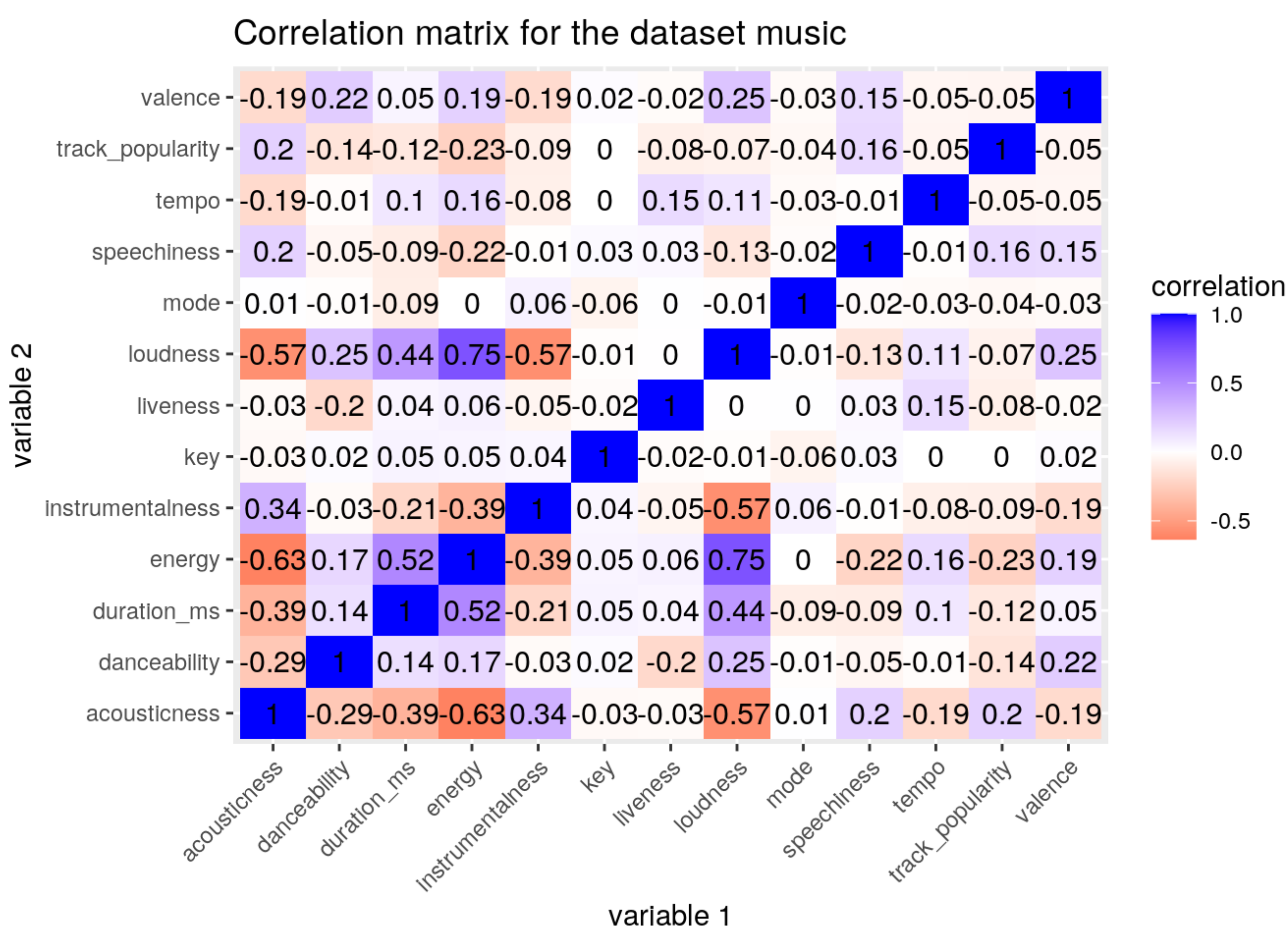
```
# Build variate and bivariate panels
pairs.panels(music_num,
             # correlation method
             method = "pearson",
             # color histogram
             hist.col = "blue",
             # make points transparent
             smoother = TRUE,
             smooth = FALSE,
             density = FALSE,
             ellipses = FALSE,
             # change labels
             cex.labels = 0.8,
             cex.cor = 1.5)
```



Looking at the univariate / bivariate graphs, we can see both correlations and distributions of numeric variables. For example, it looks like there's a strong positive correlation between "energy" and "loudness". Other variables however, such as "mode" and "key" seem disconnected from the rest of the dataset. The distributions along the diagonal are quite varied, and much of the data doesn't appear normally distributed.

Since there is so much numeric data, it is difficult to see what's going on. Let's trying using a heatmap instead.

```
# Create heatmap
cor(music_num, use = "pairwise.complete.obs") %>%
  # Save as data frame
  as.data.frame %>%
  # Convert rows to columns
  rownames_to_column %>%
  # Pivot all but first column
  pivot_longer(-1, names_to = "other_var", values_to = "correlation") %>%
  # Set up graph
  ggplot(aes(rowname, other_var, fill = correlation)) +
  geom_tile() +
  # Change gradient for better visibility
  scale_fill_gradient2(low="red",mid = "white", high = "blue") +
  # Text parameters
  geom_text(aes(label=round(correlation,2)), color = "black", size = 4) +
  # Labels
  labs(title = "Correlation matrix for the dataset music", x = "variable 1", y = "variab
le 2") +
  # Rotate x-axis text
  theme( axis.text.x = element_text( angle = 45, hjust = 1))
```



After creating a heatmap, we can see the relationships much more clearly.

The most notable positive relationships involve energy and loudness (r = .75), energy and duration (r = .52), and loudness and duration (r = .44),

The most notable negative relationships involve acousticness and energy (r = -.63), acousticness and loudness (r = -.57), and instrumentalness and energy (r = -.39).

All in all, it appears that "energy" and "loudness" have strong relationships with other variables in multiple directions. But based on their definitions, there may be some collinearity going on.

Part 3: Clustering

Let's see if we can predict a musical genre based on its numerical qualities.

First, we'll do PAM clustering on Gower's dissimilarity matrix.

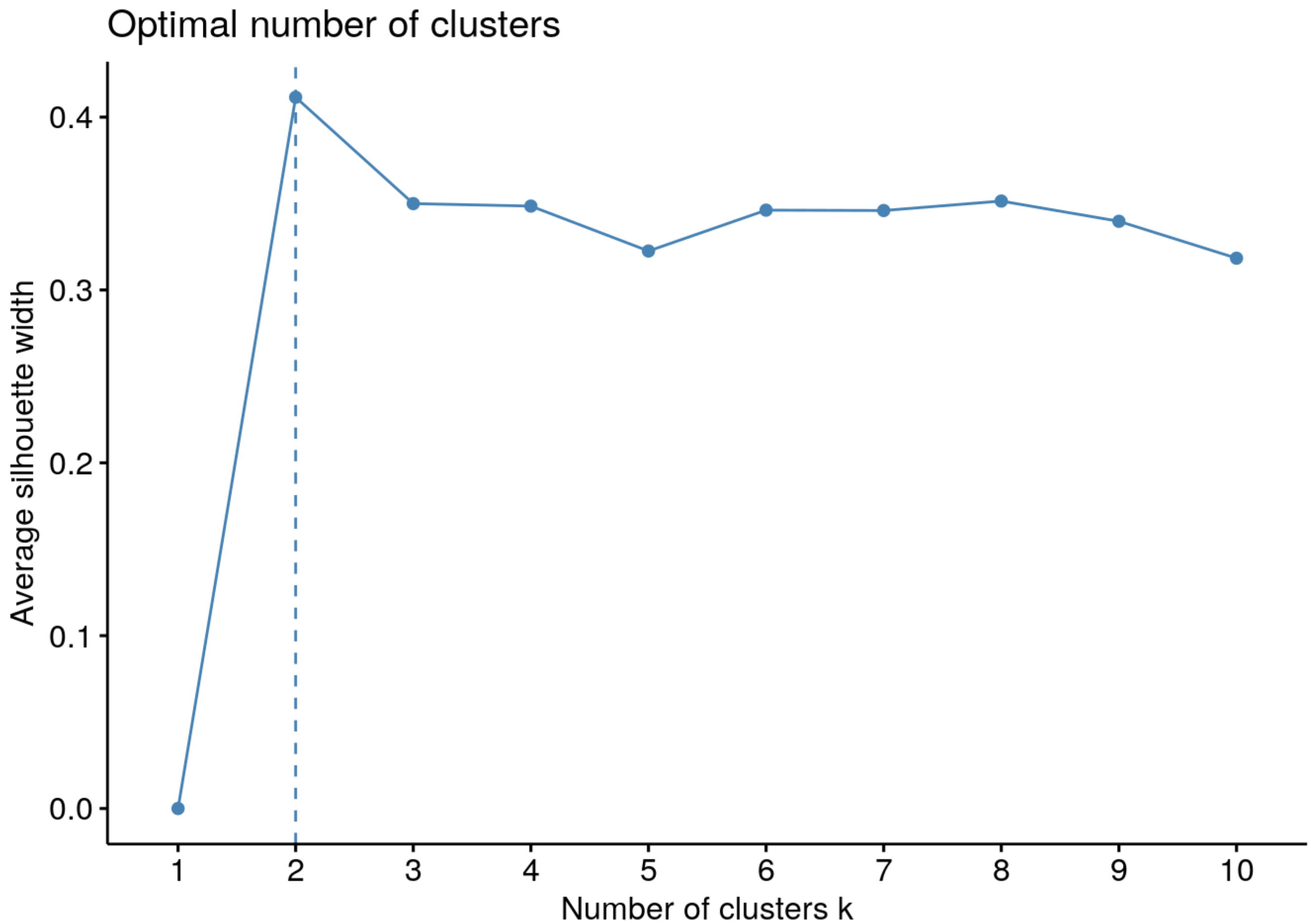
```
# Change genre into factor
music <- music %>%
  mutate_at("genre", as.factor)

# Keep only variables of interest
music_clean <- music %>%
  select(where(is.numeric), genre)

# Store dissimilarity matrix
music_gower <-
  daisy(music_clean, metric = "gower") %>%
  as.matrix
```

```
## Warning in daisy(music_clean, metric = "gower"): binary variable(s) 6 treated as
## interval scaled
```

```
# Use dissimilarity matrix to find number of optimal clusters
fviz_nbclust(music_gower, pam, method = "silhouette")
```



Although 2 clusters was suggested, the average silhouette width isn't much different at 3. Let's attempt PAM with 3 clusters to see if it can detect our 3 genres.

```
# Store PAM with 3 clusters
dis_pam_results <- pam(music_gower, k = 3, diss = TRUE)

# Check strength
dis_pam_results$silinfo$avg.width
```

```
## [1] 0.2698234
```

We got a strength of 0.27. This means our structure is weak and could be artificial.

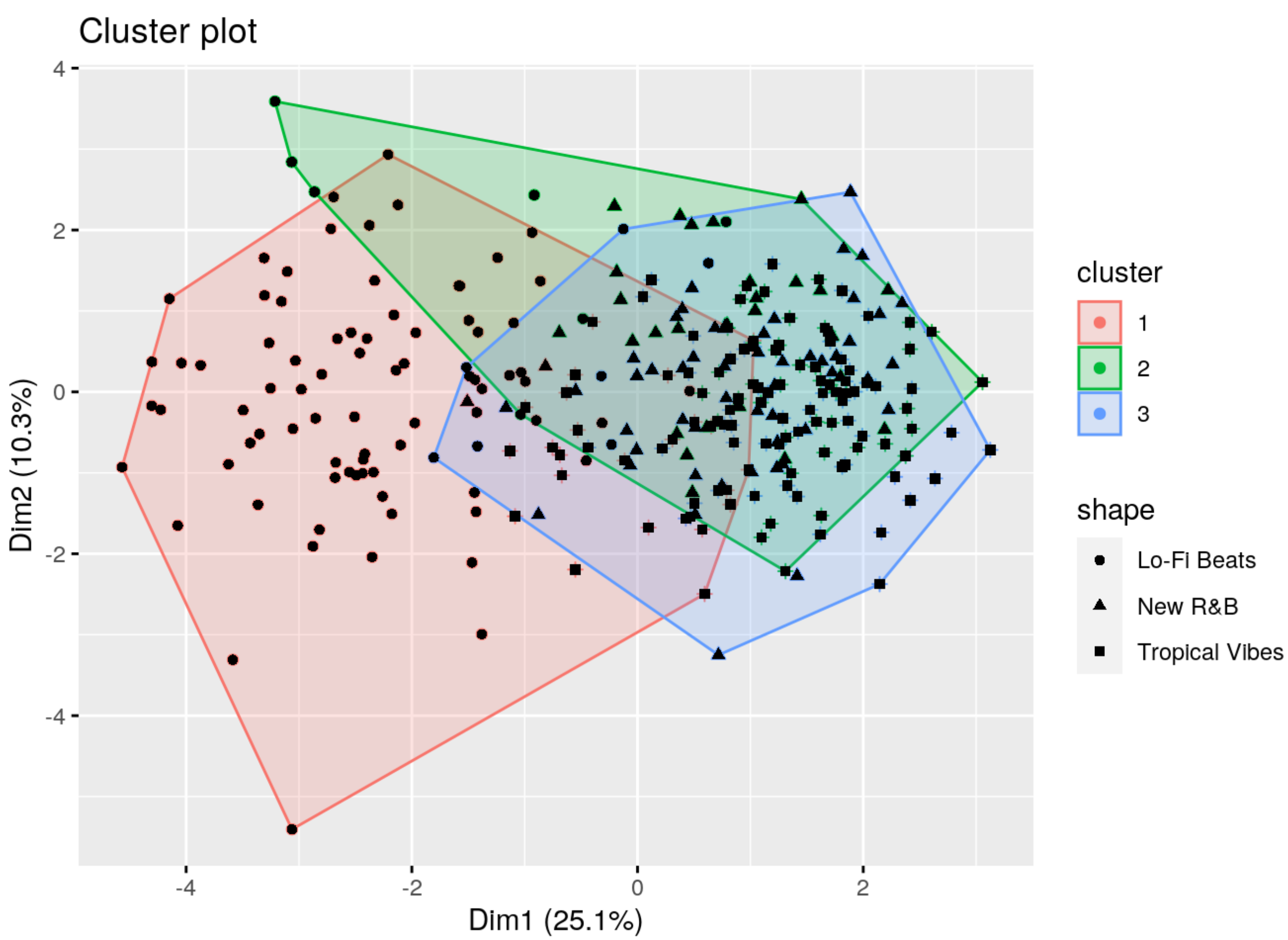
Let's visualize how the clusters measure against the actual genres.

```
# Scale data
music_scaled <- music %>%
  select(where(is.numeric)) %>%
  scale

# Run PAM on 3 clusters
pam_results <- music_scaled %>%
  pam(k = 3)

# Pairwise visualization
fviz_cluster(pam_results, data = music, geom = "point",
             shape = music$genre) +
  geom_point(aes(shape = music$genre)) +
  guides(shape = guide_legend(title = "shape"))
```

```
## Warning in if (shape %in% colnames(data)) {: the condition has length > 1 and
## only the first element will be used
```



This visualization shows the three genres, paired with the cluster they were grouped in. It seems that Lo-Fi Beats is the most distinct out of the three categories, as it is farther away in PC1 than the other clusters. Surprisingly, New R&B and Tropical Vibes have a lot of overlap.

Let’s visualize the proportions of genre by cluster.

```
# Assign clusters to each song
dis_music_pam <- music %>%
  mutate(cluster = as.factor(dis_pam_results$clustering))

# Proportion of genre by cluster
prop.table(table(dis_music_pam$cluster, dis_music_pam$genre), margin = 1)
```

```
##
##      Lo-Fi Beats New R&B    Tropical Vibes
##    1  0.87755102 0.00000000     0.12244898
##    2  0.07692308 0.91025641     0.01282051
##    3  0.00862069 0.09482759     0.89655172
```

Based on the cluster dissimilarities, it seems that the first cluster is mostly Lo-Fi Beats, while the second is New R&B and the third is Tropical Vibes.

Let's get some more information on these clusters!

```
# Simple stats

 # Add clusters as column
music_pam <- music %>%
  mutate(cluster = as.factor(pam_results$clustering))

# Check average numeric value for each cluster
music_pam %>%
  group_by(cluster) %>%
  summarize_if(is.numeric, mean, na.rm = T)
```

```
## # A tibble: 3 x 14
##   cluster track_popularity danceability energy   key loudness  mode speechiness
##   <fct>              <dbl>        <dbl>  <dbl> <dbl>    <dbl> <dbl>       <dbl>
## 1 1                   52.8        0.657  0.370  5.31   -12.8  0.635      0.104
## 2 2                   47.3        0.692  0.654  5.38    -6.79 0.859      0.147
## 3 3                   48.2        0.706  0.625  6.03    -7.41 0.182      0.0852
## # … with 6 more variables: acousticness <dbl>, instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration_ms <dbl>
```

```
# Which songs are most representative of each cluster?
music[pam_results$id.med,]
```

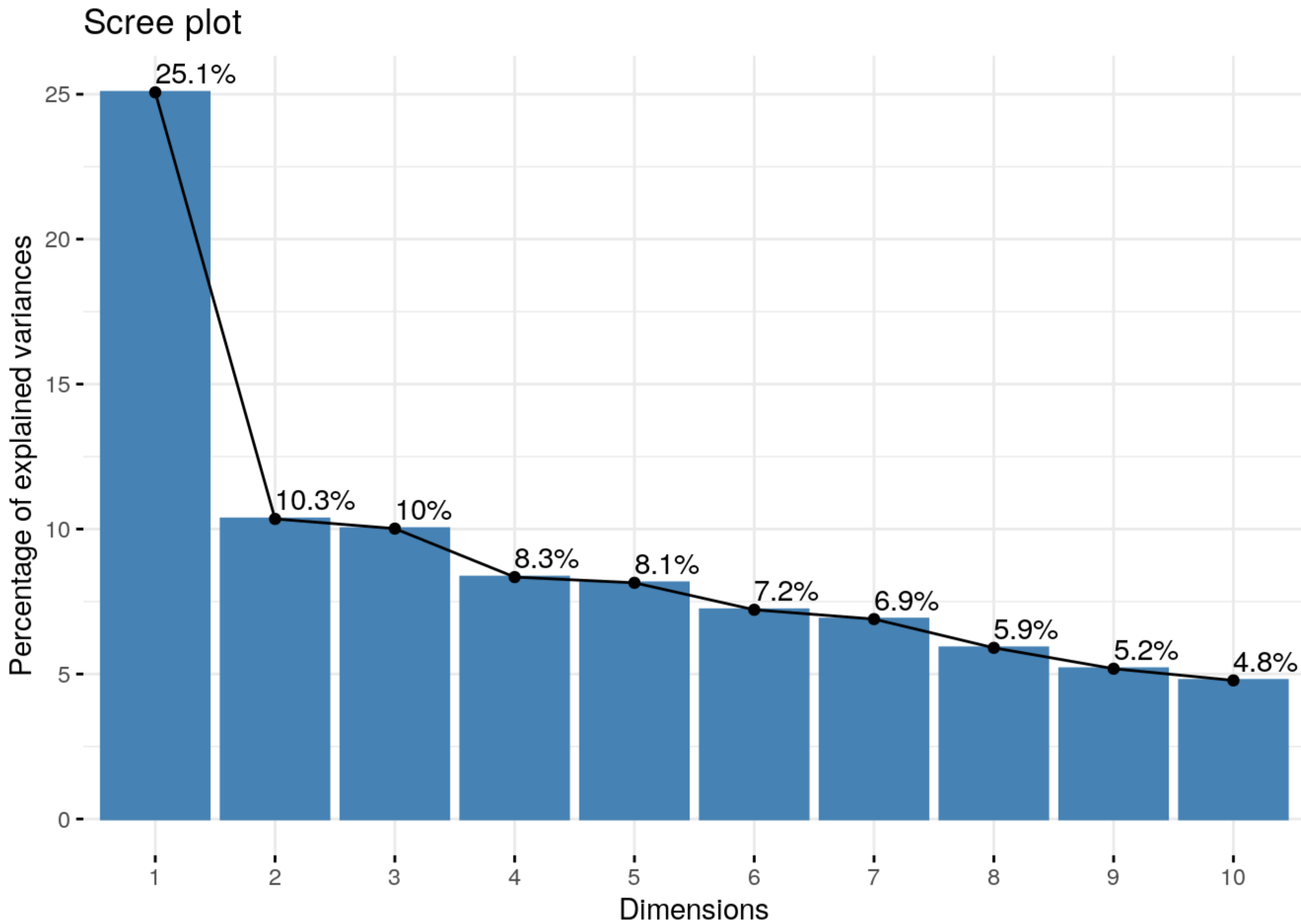
```
## # A tibble: 3 x 23
##   track_id track_name track_artist track_popularity track_album_id
##   <chr>    <chr>      <chr>                   <dbl> <chr>
## 1 69ScUfm… Take Me B… WYS                        60 5O8TzNyhGK7xT…
## 2 5rUTxVx… Wild Ones… Theresa Rex                45 1OczM3udGYiO4…
## 3 5tXg9mq… We Don't … Vivid Color                50 4DYis81pNotnb…
## # … with 18 more variables: track_album_name <chr>,
## #   track_album_release_date <chr>, genre <fct>, playlist_id <chr>,
## #   playlist_genre <chr>, playlist_subgenre <chr>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, duration_ms <dbl>
```

Although our model is pretty good at predicting genres, the overlap between Clusters 2 and 3 are striking. Let's see what goes into PC1 and PC2 to better understand our data.

Part 4: Dimensionality Reduction

```
# Store principle components
pca <- music_scaled %>%
  prcomp()

# Visualize principle components
fviz_eig(pca, addlabels = TRUE)
```



It looks like there's a tremendous drop-off in explained variation from PC1 to PC2. It doesn't seem like there's much added benefit after the first two components, as the explained variation doesn't seem to change much after the drop.

Let's look at total variance explained as a function of these principle components:

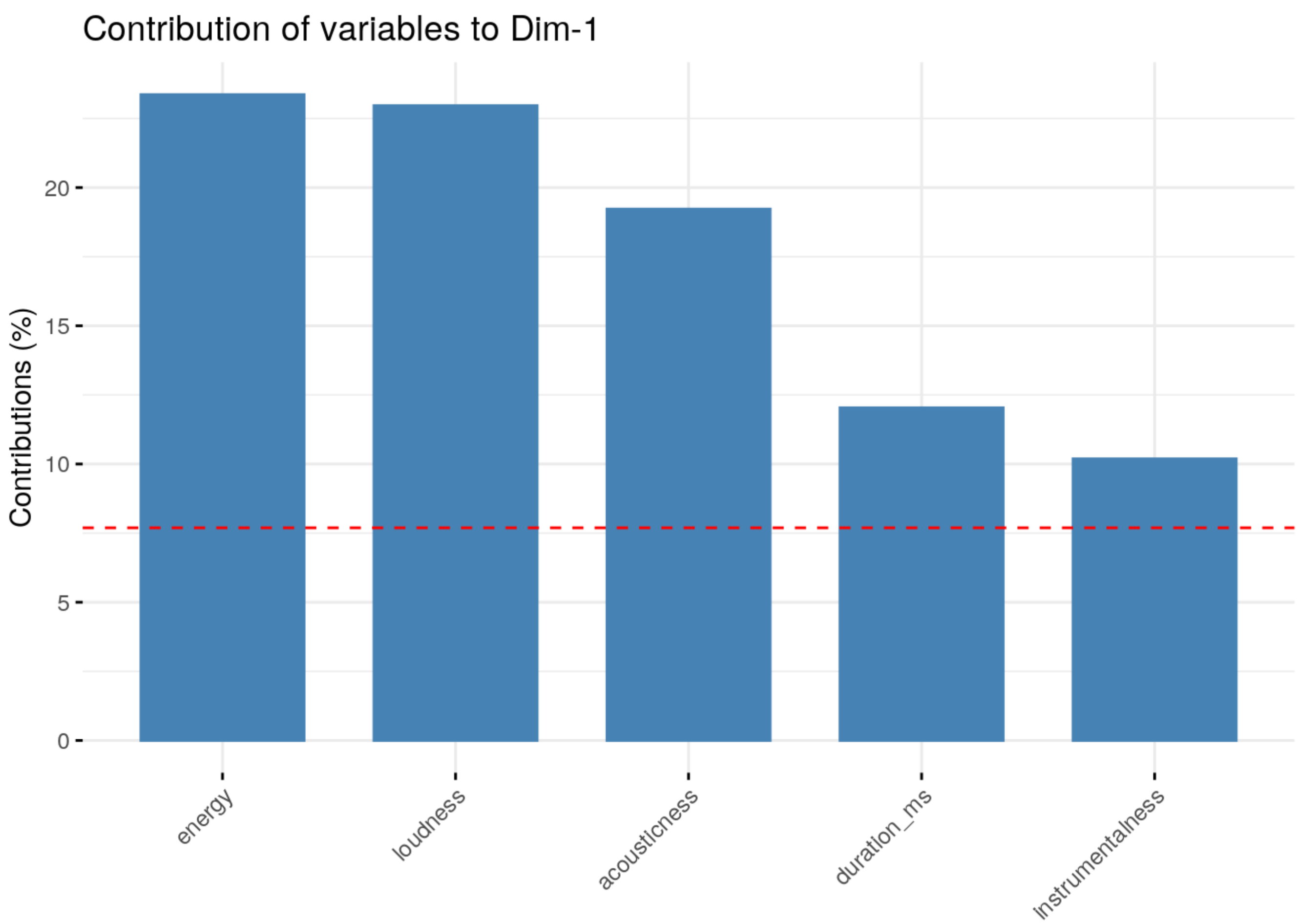
```
# Look at principle components and variation explained.
get_eigenvalue(pca)
```

```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1    3.2584771        25.065208                    25.06521
## Dim.2    1.3454712        10.349779                    35.41499
## Dim.3    1.3019925        10.015327                    45.43031
## Dim.4    1.0846866         8.343743                    53.77406
## Dim.5    1.0590272         8.146363                    61.92042
## Dim.6    0.9380595         7.215843                    69.13626
## Dim.7    0.8963092         6.894686                    76.03095
## Dim.8    0.7674475         5.903442                    81.93439
## Dim.9    0.6737445         5.182650                    87.11704
## Dim.10   0.6213419         4.779553                    91.89660
## Dim.11   0.4779004         3.676157                    95.57275
## Dim.12   0.3858555         2.968119                    98.54087
## Dim.13   0.1896867         1.459129                   100.00000
```

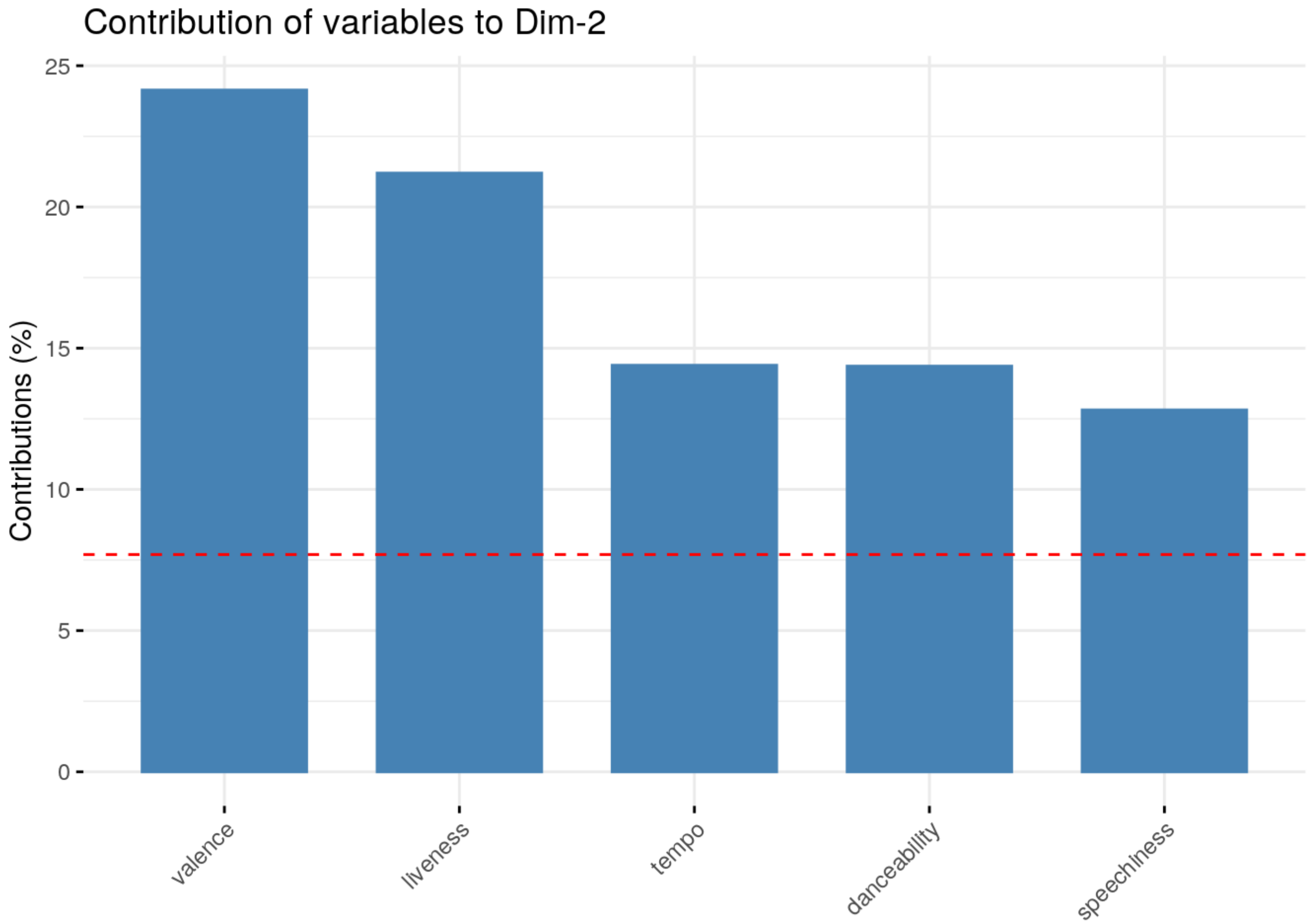
It looks like we need 8 principle components to capture about 80% of the variance in our data. Even after 10 principle components, we're not as close as we'd like to be! Since we have so many numerical measurements in our dataset, it makes sense that the weights on each variable are significantly less.

We will only retain the first two components, as going beyond two doesn't seem very beneficial. Let's look into what goes into building these components.

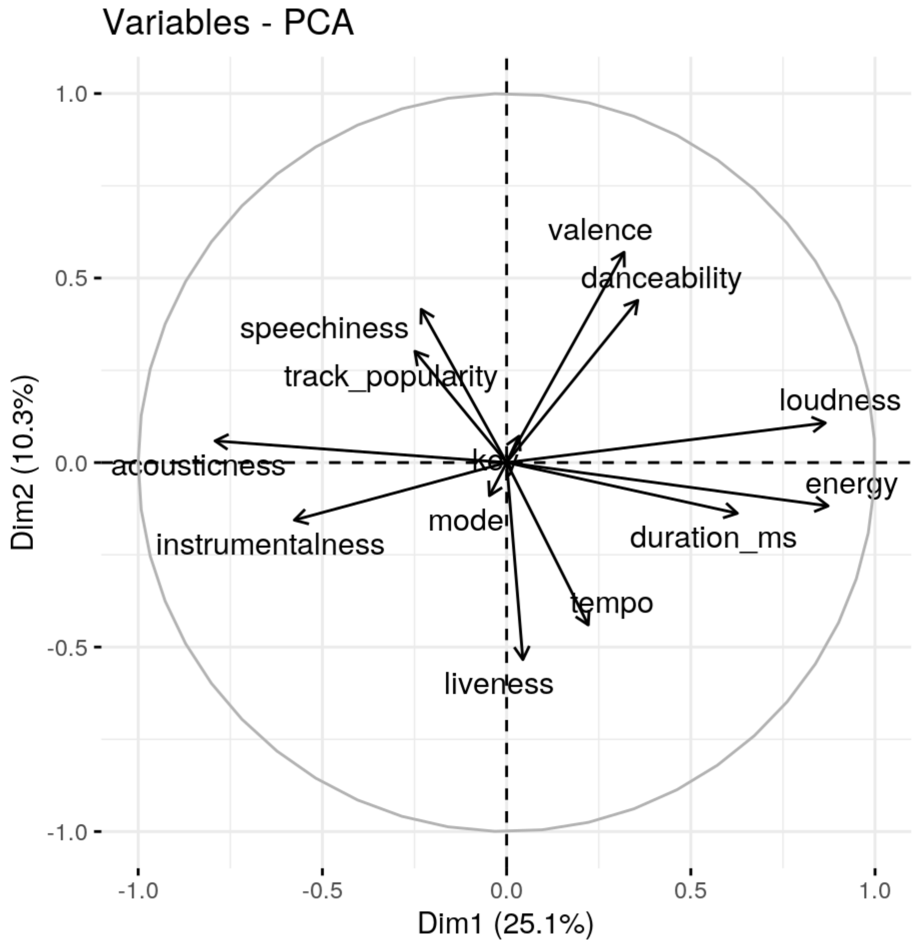
```
# Variable contributions to PC1
fviz_contrib(pca, choice = "var", axes = 1, top = 5)
```




```
# Variable contributions to PC2
fviz_contrib(pca, choice = "var", axes = 2, top = 5)
```




```
# All variable contributions to PC1 and PC2
fviz_pca_var(pca, col.var = "black", repel = TRUE)
```



In the circular plot, we can see how much each variable contributes to a given dimension. It seems like PC1 is some construct for “volume”. The arrows pointing to the right sound like loud music, while the arrows pointing toward the left are aspects that I would imagine softer tracks having. For PC2, it seems that variables that contribute positively are “pop-like”, since it has aspects of vocals, dance, and valence. Variables that contribute negatively to PC2 seem to be to be less lively. For example. liveness is a measure of how much background audio is heard within the track, and is seen to be pointing down in the plot (radio-hosts? podcasts?).For both dimensions, it appears that the *key* and *mode* variables make little to no difference. Although there certainly are some variables that contribute more to PC1 & PC2 than others (as seen in the barcharts above), at first glance, the variables in this high-dimension graph don’t appear overly biased. Based on this, it will probably be difficult to predict popularity with only a few variables.

Part 5: Classification and Cross-validation

Let’s make a logistic regression model to test whether or not a song will be popular.

The variable ‘track_popularity’ in our dataset contains a value from 0-100 on how popular a song is. We will use an arbitrary cut-off point (quartile 3) to determine if a song is popular or not. Everything greater than this will receive a score of 1, otherwise, 0.

```
# Create five number summary of popularity distribution
fivenum(music$track_popularity)
```

```
## [1]  1 44 54 59 93
```

According to the five-number summary, a score of 59 on the track_popularity variable is considered third quartile in the distribution. Let's consider everything greater than or equal to 59 as "excellent"!

```
# Create binary outcome variable
music2 <- music %>%
  mutate(excellent = track_popularity >= 59) %>%
  mutate_at("excellent", as.numeric) %>%
  select(-track_popularity)

# Dataset to be used in the classifier
head(music2)
```

```
## # A tibble: 6 x 23
##   track_id track_name track_artist track_album_id track_album_name
##   <chr>    <chr>      <chr>        <chr>          <chr>
## 1 0rNpm25… With U     SwuM         5YuMyydKScBvK… With U
## 2 4c2TiYo… sekao      Delayde      2yTJ6fdaX9ZYG… running around …
## 3 2tVHXIM… Aconcagua  Slumberville 480XGKz77Nqgc… Aconcagua
## 4 2pZi2b9… thinking … mommy        6mOgPOHFf2JTZ… Inaudible
## 5 66z8EZX… Crossroads Hanz         3u1k0CIc2zQdX… Chillhop Essent…
## 6 74t6wFV… memories … Rook1e       7EIbQnhNrS9Tb… memories of her…
## # … with 18 more variables: track_album_release_date <chr>, genre <fct>,
## #   playlist_id <chr>, playlist_genre <chr>, playlist_subgenre <chr>,
## #   danceability <dbl>, energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>,
## #   speechiness <dbl>, acousticness <dbl>, instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration_ms <dbl>,
## #   excellent <dbl>
```

Now, let's train a logistic regression model!

```
# Randomly take half the dataset
train <- sample_frac(music2, size = 0.5)

# Take the other half
test <- anti_join(music2, train, by = "track_id")

# Store logistic classifier, predicting excellence
# with multiple numeric predictors
fit <- glm(excellent ~ danceability + energy + key + loudness + mode + speechiness + aco
usticness + instrumentalness + liveness, data = train, family = "binomial")

# Create training data
df_train <- data.frame(
  probability = predict(fit, newdata = train, type = "response"),
  excellent = train$excellent,
  data_name = "training")

# Create test data
df_test <- data.frame(
  probability = predict(fit, newdata = test, type = "response"),
  excellent = test$excellent,
  data_name = "test")

# Combine training and test data
df_combined <- rbind(df_train, df_test)
```

Let's check how well the model is doing with the AUC value.

```
# Create ROC
ROC <- ggplot(df_combined) +
  geom_roc(aes(d = excellent, m = probability, color = data_name, n.cuts = 0))
```

```
## Warning: Ignoring unknown aesthetics: n.cuts
```

```
# Calculate AUC
calc_auc(ROC)
```

```
##   PANEL group       AUC
## 1     1     1 0.7709571
## 2     1     2 0.5724057
```

Using the logistic regression classifier, we received an AUC of 71% on the training-dataset. This is considered a fair model! The test dataset on the other hand, was almost a coin-toss. We should verify our findings with k-fold cross validation.

```
# Choose number of folds
k = 10

# Rearrange rows
data <- music2[sample(nrow(music2)), ]

# Create k folds from the dataset
folds <- cut(seq(1:nrow(data)), breaks = k, labels = FALSE)

# Initialize empty vector
diags_k <- NULL

for(i in 1:k){
  # Create training and test sets
  train <- data[folds != i, ]
  test <- data[folds == i, ]

  # Train model on training set (all but fold i)
  fit <- glm(excellent ~ danceability + energy + key + loudness + mode + speechiness + a
cousticness + instrumentalness + liveness, data = train, family = "binomial")

  # Test model on test set (fold i)
  df <- data.frame(
    probability = predict(fit, newdata = test, type = "response"),
    excellent = test$excellent)

  # Create ROC
  ROC <- ggplot(df) +
    geom_roc(aes(d = excellent, m = probability, n.cuts = 0))

  # Store AUCs
  diags_k[i] <- calc_auc(ROC)$AUC
}
```

```
## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts

## Warning: Ignoring unknown aesthetics: n.cuts
```

```
# Take mean of AUCs
mean(diags_k)
```

```
## [1] 0.6634622
```

After k-fold cross validation, we get an average performance of 65% across our k folds. This is a 6% decrease in AUC compared to pre-validation. It is possible that this drop in accuracy was due to overfitting due to our small sample size. Although this is slightly disappointing, this 65% is more reflective of the AUC when exposed to newer environments. I’ve heard of people dropping 30% after validation so it could always be worse! For my first machine learning project, I’ll take it :)