FP

Ma Lok Sum, Zoe (a1819866)

2022-11-12

Load the data frame of spotify songs into R

```
library(spotifyr)
spotify_songs <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
```

```
## Rows: 32833 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...
##
## 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 it
spotify_songs
```

```
## # A tibble: 32,833 x 23
##    track_id      track~1 track~2 track~3 track~4 track~5 track~6 playl~7 playl~8
##    <chr>         <chr>   <chr>     <dbl> <chr>   <chr>   <chr>   <chr>   <chr>
##  1 6f807x0ima9a~ I Don'~ Ed She~      66 2oCs0D~ I Don'~ 2019-0~ Pop Re~ 37i9dQ~
##  2 0r7CVbZTWZgb~ Memori~ Maroon~      67 63rPSO~ Memori~ 2019-1~ Pop Re~ 37i9dQ~
##  3 1z1Hg7Vb0AhH~ All th~ Zara L~      70 1HoSmj~ All th~ 2019-0~ Pop Re~ 37i9dQ~
##  4 75FpbthrwQmz~ Call Y~ The Ch~      60 1nqYsO~ Call Y~ 2019-0~ Pop Re~ 37i9dQ~
##  5 1e8PAfcKUYoK~ Someon~ Lewis ~      69 7m7vv9~ Someon~ 2019-0~ Pop Re~ 37i9dQ~
##  6 7fvUMiyapMsR~ Beauti~ Ed She~      67 2yiy9c~ Beauti~ 2019-0~ Pop Re~ 37i9dQ~
##  7 2OAylPUDDfwR~ Never ~ Katy P~      62 7INHYS~ Never ~ 2019-0~ Pop Re~ 37i9dQ~
##  8 6b1RNvAcJjQH~ Post M~ Sam Fe~      69 6703SR~ Post M~ 2019-0~ Pop Re~ 37i9dQ~
##  9 7bF6tCO3gFb8~ Tough ~ Avicii       68 7CvAfG~ Tough ~ 2019-0~ Pop Re~ 37i9dQ~
## 10 1IXGILkPm0tO~ If I C~ Shawn ~      67 4Qxzbf~ If I C~ 2019-0~ Pop Re~ 37i9dQ~
## # ... with 32,823 more rows, 14 more variables: 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>, and abbreviated variable names 1: track_name,
## #   2: track_artist, 3: track_popularity, 4: track_album_id,
## #   5: track_album_name, 6: track_album_release_date, 7: playlist_name, ...
```

1. Introduction to the data set

discipline area

The discipline area of the data set is music, business, global.

file type

The data set is a CSV file type from the music company Spotify, which enables data to be saved in a tabular style.

variable

```
# view the first 6 row of the variables
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
as_tibble(spotify_songs)%>%head()
```

```
## # A tibble: 6 x 23
##   track_id       track~1 track~2 track~3 track~4 track~5 track~6 playl~7 playl~8
##   <chr>          <chr>   <chr>     <dbl> <chr>   <chr>   <chr>   <chr>   <chr>
## 1 6f807x0ima9a1~ I Don'~ Ed She~      66 2oCs0D~ I Don'~ 2019-0~ Pop Re~ 37i9dQ~
## 2 0r7CVbZTWZgbT~ Memori~ Maroon~      67 63rPS0~ Memori~ 2019-1~ Pop Re~ 37i9dQ~
## 3 1z1Hg7Vb0AhHD~ All th~ Zara L~      70 1HoSmj~ All th~ 2019-0~ Pop Re~ 37i9dQ~
## 4 75FpbthrwQmzH~ Call Y~ The Ch~      60 1nqYsO~ Call Y~ 2019-0~ Pop Re~ 37i9dQ~
## 5 1e8PAfcKUYoKk~ Someon~ Lewis ~      69 7m7vv9~ Someon~ 2019-0~ Pop Re~ 37i9dQ~
## 6 7fvUMiyapMsRR~ Beauti~ Ed She~      67 2yiy9c~ Beauti~ 2019-0~ Pop Re~ 37i9dQ~
## # ... with 14 more variables: 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>, and
## #   abbreviated variable names 1: track_name, 2: track_artist,
## #   3: track_popularity, 4: track_album_id, 5: track_album_name,
## #   6: track_album_release_date, 7: playlist_name, 8: playlist_id
```

There are the description of all the variables in the dataframe:

track_id: A specific song ID.

track_name: The name of a song.

track_artist: The singer.

track_popularity: A popularity score ranging from 0 to 100.

The popularity of the music increases with value.

track_album_id: A distinctive album ID.

track_album_name: Album's title.

track_album_release_date: The released date of album.

playlist_name: The name of the playlist from which the song was taken.

playlist_ID: The playlist's unique ID.

playlist_genre: The playlist's genre. This is the key outcome variable of interest.

playlist_subgenre: The playlist's genre subcategory.

danceability: A number between 0 and 1 that indicates how danceable a song is.

The tune is more danceable the higher the value.

energy: A scale that ranges from 0.0 to 1.0 and serves as a gauge for perceived activity and intensity.

The music is more spirited the higher the value.

key: An integer that represents the song's overall pitch.

Each increase in pitch corresponds to an integer, and an undetectable pitch to -1.

loudness: A number that often falls between -60 and 0 decibels and describes how loud the recording is overall.

mode: A number between 0 and 1 that represents the scale from which the melodic content is formed.

A 0 is performed in minor and a 1 is performed in major.

speechiness: A number between 0 and 1 indicating how "speechy" the song is.

Spoken-word tracks are indicated by higher values.

acousticness: A number between 0 and 1 indicating the track's acoustic quality (More acoustic means higher values).

instrumentalness: A score between 0 and 1 that indicates how instrumental the song is.

Higher values indicate a music with fewer vocals.

liveness: Determines whether a song was recorded in a studio or live and how "live" it is.

valence: A scale from 0 to 1 would indicate how upbeat the song is.

Positive values have higher values.

tempo: The song's beats per minute tempo.

duration_ms: The song's length in milliseconds.

data types

```
# skim the data
library(skimr)
skim(spotify_songs)
```

Table 1: Data summary

Name	spotify_songs
Number of rows	32833
Number of columns	23
_____	
Column type frequency:	
character	10
numeric	13
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
track_id	0	1	22	22	0	28356	0
track_name	5	1	1	144	0	23449	0
track_artist	5	1	2	69	0	10692	0
track_album_id	0	1	22	22	0	22545	0
track_album_name	5	1	1	151	0	19743	0
track_album_release_date	0	1	4	10	0	4530	0
playlist_name	0	1	6	120	0	449	0
playlist_id	0	1	22	22	0	471	0
playlist_genre	0	1	3	5	0	6	0
playlist_subgenre	0	1	4	25	0	24	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
track_popularity	0	1	42.48	24.98	0.00	24.00	45.00	62.00	100.00	
danceability	0	1	0.65	0.15	0.00	0.56	0.67	0.76	0.98	
energy	0	1	0.70	0.18	0.00	0.58	0.72	0.84	1.00	
key	0	1	5.37	3.61	0.00	2.00	6.00	9.00	11.00	
loudness	0	1	-6.72	2.99	-46.45	-8.17	-6.17	-4.64	1.27	
mode	0	1	0.57	0.50	0.00	0.00	1.00	1.00	1.00	
speechiness	0	1	0.11	0.10	0.00	0.04	0.06	0.13	0.92	
acousticness	0	1	0.18	0.22	0.00	0.02	0.08	0.26	0.99	
instrumentalness	0	1	0.08	0.22	0.00	0.00	0.00	0.00	0.99	
liveness	0	1	0.19	0.15	0.00	0.09	0.13	0.25	1.00	
valence	0	1	0.51	0.23	0.00	0.33	0.51	0.69	0.99	
tempo	0	1	120.88	26.90	0.00	99.96	121.98	133.92	239.44	
duration_ms	0	1	225799.81	59834.01	4000.00	187819.00	216000.00	253585.00	517810.00	

There are 10 character variables and 13 numeric variables.

There is one variable (playlist_genre) been read in incorrectly.

There are 15 missing data, which located in the variable of track_name,track_artist and track_album_name.

There are 32833 observations on 23 variables.

Data cleaning and organizating

extract the year

```
details <- str_match(spotify_songs$track_album_release_date, "(\\d+)-(\\d+)-(\\d+)")
spotify_songs$track_album_release_date <- details[, 2]
spotify_songs$track_album_release_date <- as.integer(spotify_songs$track_album_release_date)
spotify_songs
```

```
## # A tibble: 32,833 x 23
##    track_id      track~1 track~2 track~3 track~4 track~5 track~6 playl~7 playl~8
##    <chr>         <chr>   <chr>     <dbl> <chr>   <chr>     <int> <chr>   <chr>
##  1 6f807x0ima9a~ I Don'~ Ed She~      66 2oCs0D~ I Don'~    2019 Pop Re~ 37i9dQ~
##  2 0r7CVbZTWZgb~ Memori~ Maroon~      67 63rPS0~ Memori~    2019 Pop Re~ 37i9dQ~
##  3 1z1Hg7Vb0AhH~ All th~ Zara L~      70 1HoSmj~ All th~    2019 Pop Re~ 37i9dQ~
##  4 75FpbthrwQmz~ Call Y~ The Ch~      60 1nqYsO~ Call Y~    2019 Pop Re~ 37i9dQ~
##  5 1e8PAfcKUYoK~ Someon~ Lewis ~      69 7m7vv9~ Someon~    2019 Pop Re~ 37i9dQ~
##  6 7fvUMiyapMsR~ Beauti~ Ed She~      67 2yiy9c~ Beauti~    2019 Pop Re~ 37i9dQ~
##  7 2OAylPUDDfwR~ Never ~ Katy P~      62 7INHYS~ Never ~    2019 Pop Re~ 37i9dQ~
##  8 6b1RNvAcJjQH~ Post M~ Sam Fe~      69 6703SR~ Post M~    2019 Pop Re~ 37i9dQ~
##  9 7bF6tCO3gFb8~ Tough ~ Avicii       68 7CvAfG~ Tough ~    2019 Pop Re~ 37i9dQ~
## 10 1IXGILkPm0tO~ If I C~ Shawn ~      67 4Qxzbf~ If I C~    2019 Pop Re~ 37i9dQ~
## # ... with 32,823 more rows, 14 more variables: 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>, and abbreviated variable names 1: track_name,
## #   2: track_artist, 3: track_popularity, 4: track_album_id,
## #   5: track_album_name, 6: track_album_release_date, 7: playlist_name, ...
```

The song's period of time is 1960 until 2022.

remove the NA varaibles in the dataset

```
spotify_songs<-drop_na(spotify_songs)

# review it again
spotify_songs
```

```
## # A tibble: 30,942 x 23
##    track_id      track~1 track~2 track~3 track~4 track~5 track~6 playl~7 playl~8
##    <chr>         <chr>   <chr>     <dbl> <chr>   <chr>     <int> <chr>   <chr>
##  1 6f807x0ima9a~ I Don'~ Ed She~      66 2oCs0D~ I Don'~    2019 Pop Re~ 37i9dQ~
##  2 0r7CVbZTWZgb~ Memori~ Maroon~      67 63rPS0~ Memori~    2019 Pop Re~ 37i9dQ~
##  3 1z1Hg7Vb0AhH~ All th~ Zara L~      70 1HoSmj~ All th~    2019 Pop Re~ 37i9dQ~
##  4 75FpbthrwQmz~ Call Y~ The Ch~      60 1nqYsO~ Call Y~    2019 Pop Re~ 37i9dQ~
##  5 1e8PAfcKUYoK~ Someon~ Lewis ~      69 7m7vv9~ Someon~    2019 Pop Re~ 37i9dQ~
##  6 7fvUMiyapMsR~ Beauti~ Ed She~      67 2yiy9c~ Beauti~    2019 Pop Re~ 37i9dQ~
```

```
##  7 2OAylPUDDfwR~ Never ~ Katy P~      62 7INHYS~ Never ~    2019 Pop Re~ 37i9dQ~
##  8 6b1RNvAcJjQH~ Post M~ Sam Fe~      69 6703SR~ Post M~    2019 Pop Re~ 37i9dQ~
##  9 7bF6tCO3gFb8~ Tough ~ Avicii       68 7CvAfG~ Tough ~    2019 Pop Re~ 37i9dQ~
## 10 1IXGILkPm0tO~ If I C~ Shawn ~      67 4Qxzbf~ If I C~    2019 Pop Re~ 37i9dQ~
## # ... with 30,932 more rows, 14 more variables: 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>, and abbreviated variable names 1: track_name,
## #   2: track_artist, 3: track_popularity, 4: track_album_id,
## #   5: track_album_name, 6: track_album_release_date, 7: playlist_name, ...
```

The dataset is huge.

It is not effective for machine to run the coding

Therefore, we sample the dataset from 32833 to 1000 observations.

```
sub_spotify=spotify_songs[sample(nrow(spotify_songs), 1000),]
sub_spotify
```

```
## # A tibble: 1,000 x 23
##    track_id      track~1 track~2 track~3 track~4 track~5 track~6 playl~7 playl~8
##    <chr>         <chr>   <chr>     <dbl> <chr>   <chr>     <int> <chr>   <chr>
##  1 2iIHvI9fUC30~ Matado~ Jory B~      39 5RqPcv~ Greate~    2014 Los Ca~ 2hTs6G~
##  2 5kaiM7WaSYPy~ Don't ~ Soul&R~      33 0Q298f~ Slow J~    2014 Rock B~ 0y8MUl~
##  3 5e1s5JMszwlJ~ Ficar ~ Davi         34 4KDThQ~ Ficar ~    2019 Orgulh~ 37i9dQ~
##  4 6U6XwiK6mYgT~ Glow    Phoebe~      19 5hwIIW~ Glow       2019 Neo So~ 07SNJ4~
##  5 5EaOSrK8cEeb~ Sun Is~ Tomas ~      40 5fV0yY~ Sun Is~    2019 Tropic~ 37i9dQ~
##  6 7pAC4ebO3MVI~ 911     Saint ~      37 6mgsIf~ 911        2019 ELECTR~ 1N5dPU~
##  7 25ZAibhr3bdl~ QUE PR~ J Balv~      86 6ylFfz~ OASIS      2019 Trap 2~ 37DFLy~
##  8 6SxhMVIqpdEO~ Stones  Emmit ~      43 2vTGrJ~ Prolog~    2017 Music&~ 5jROYS~
##  9 6RQzJrr4BrsO~ Hornet~ Tom St~      33 5Rb1xL~ Hornet~    2019 Electr~ 4pVZ70~
## 10 5UbJK7IHuc6B~ I Get ~ 50 Cent       1 4BSvr6~ Curtis     2007 Gangst~ 0ZRwrJ~
## # ... with 990 more rows, 14 more variables: 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>, and abbreviated variable names 1: track_name,
## #   2: track_artist, 3: track_popularity, 4: track_album_id,
## #   5: track_album_name, 6: track_album_release_date, 7: playlist_name, ...
```

Remove the data that is not useful for the prediction.

View between the playlist_genre and playlist_subgenre

```
library(tidyr)
library(stringr)
library(dplyr)
genre <- sub_spotify %>%
  group_by(playlist_genre, playlist_subgenre) %>% summarise(count=n())
```

```
## `summarise()` has grouped output by 'playlist_genre'. You can override using
## the `.groups` argument.
```

```
genre
```

```
## # A tibble: 24 x 3
## # Groups:   playlist_genre [6]
##    playlist_genre playlist_subgenre         count
##    <chr>          <chr>                     <int>
##  1 edm            big room                     41
##  2 edm            electro house                38
##  3 edm            pop edm                      56
##  4 edm            progressive electro house    68
##  5 latin          latin hip hop                42
##  6 latin          latin pop                    40
##  7 latin          reggaeton                    36
##  8 latin          tropical                     49
##  9 pop            dance pop                    41
## 10 pop            electropop                   43
## # ... with 14 more rows
```

The varaible (playlist_subgenre) will not be used

because we aleady know the genure from the playlist_genre after reviewing the playlist_genre and playlist_subgenre.

I decided tp use playlist_subgenre as a variable because it is more specialised the genre.

There are some not useful variables for the prediction,

including track_id, track_album_id and playlist_id, because they are garble.

Besides, the variable track_album_name, playlist_name, track_artist and track_name are the string variables, which are meaningless for the machine learning.

Therefore, all of the mentioned vraiables will be removed from the dataset.

```
spotify3 = select(sub_spotify, -playlist_subgenre, -track_id, -track_album_id, -playlist_id, -track_albu
# View it
spotify3
```

```
## # A tibble: 1,000 x 15
##    track_po~1 track~2 playl~3 dance~4 energy   key loudn~5  mode speec~6 acous~7
##         <dbl>   <int> <chr>     <dbl>  <dbl> <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1         39    2014 latin     0.647  0.724    11   -5.57     0  0.0853 0.162
##  2         33    2014 rock      0.736  0.576     4   -6.35     1  0.0301 0.229
##  3         34    2019 latin     0.788  0.407     7   -7.63     0  0.26   0.209
##  4         19    2019 r&b       0.41   0.587     9   -8.45     0  0.0537 0.136
##  5         40    2019 latin     0.717  0.773     1   -6.26     1  0.0548 0.528
##  6         37    2019 edm       0.746  0.974     8   -2.42     0  0.149  0.00534
##  7         86    2019 rap       0.639  0.791    10   -4.44     0  0.253  0.0275
##  8         43    2017 pop       0.685  0.364     3  -13.0      0  0.053  0.751
##  9         33    2019 edm       0.667  0.975    11   -5.04     1  0.109  0.0268
## 10          1    2007 rap       0.525  0.933     9   -4.67     1  0.356  0.159
## # ... with 990 more rows, 5 more variables: instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration_ms <dbl>, and
## #   abbreviated variable names 1: track_popularity,
## #   2: track_album_release_date, 3: playlist_genre, 4: danceability,
## #   5: loudness, 6: speechiness, 7: acousticness
```

convert the variables into a correct form

```
spotify3$playlist_genre <- as.factor(spotify3$playlist_genre)
# review it
spotify3
```

```
## # A tibble: 1,000 x 15
##    track_po~1 track~2 playl~3 dance~4 energy   key loudn~5  mode speec~6 acous~7
##         <dbl>   <int> <fct>     <dbl>  <dbl> <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1         39    2014 latin     0.647  0.724    11   -5.57     0  0.0853 0.162
##  2         33    2014 rock      0.736  0.576     4   -6.35     1  0.0301 0.229
##  3         34    2019 latin     0.788  0.407     7   -7.63     0  0.26   0.209
##  4         19    2019 r&b       0.41   0.587     9   -8.45     0  0.0537 0.136
##  5         40    2019 latin     0.717  0.773     1   -6.26     1  0.0548 0.528
##  6         37    2019 edm       0.746  0.974     8   -2.42     0  0.149  0.00534
##  7         86    2019 rap       0.639  0.791    10   -4.44     0  0.253  0.0275
##  8         43    2017 pop       0.685  0.364     3  -13.0      0  0.053  0.751
##  9         33    2019 edm       0.667  0.975    11   -5.04     1  0.109  0.0268
## 10          1    2007 rap       0.525  0.933     9   -4.67     1  0.356  0.159
## # ... with 990 more rows, 5 more variables: instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration_ms <dbl>, and
## #   abbreviated variable names 1: track_popularity,
## #   2: track_album_release_date, 3: playlist_genre, 4: danceability,
## #   5: loudness, 6: speechiness, 7: acousticness
```

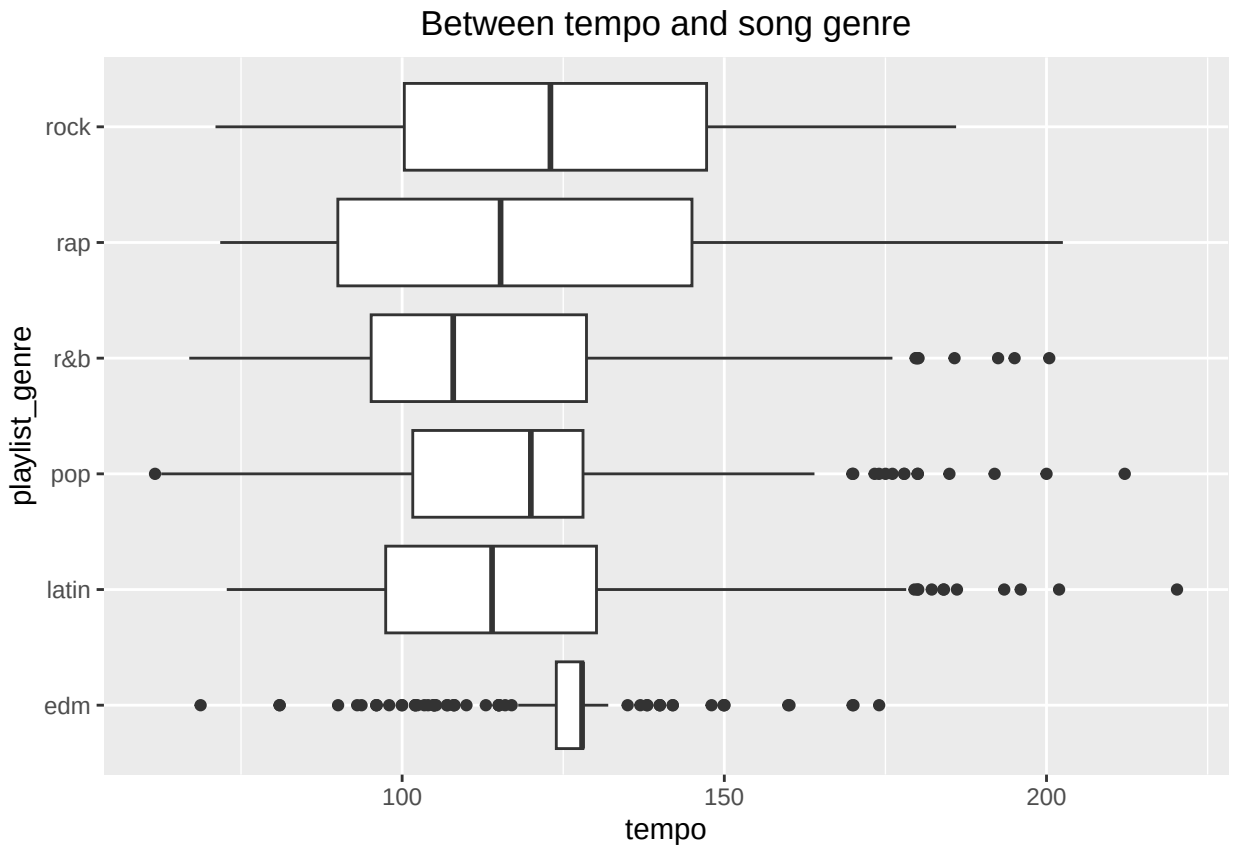
2. Summarize and visualise the data

1st plot

Use the variables of track album release date, speechiness, danceability, and tempo

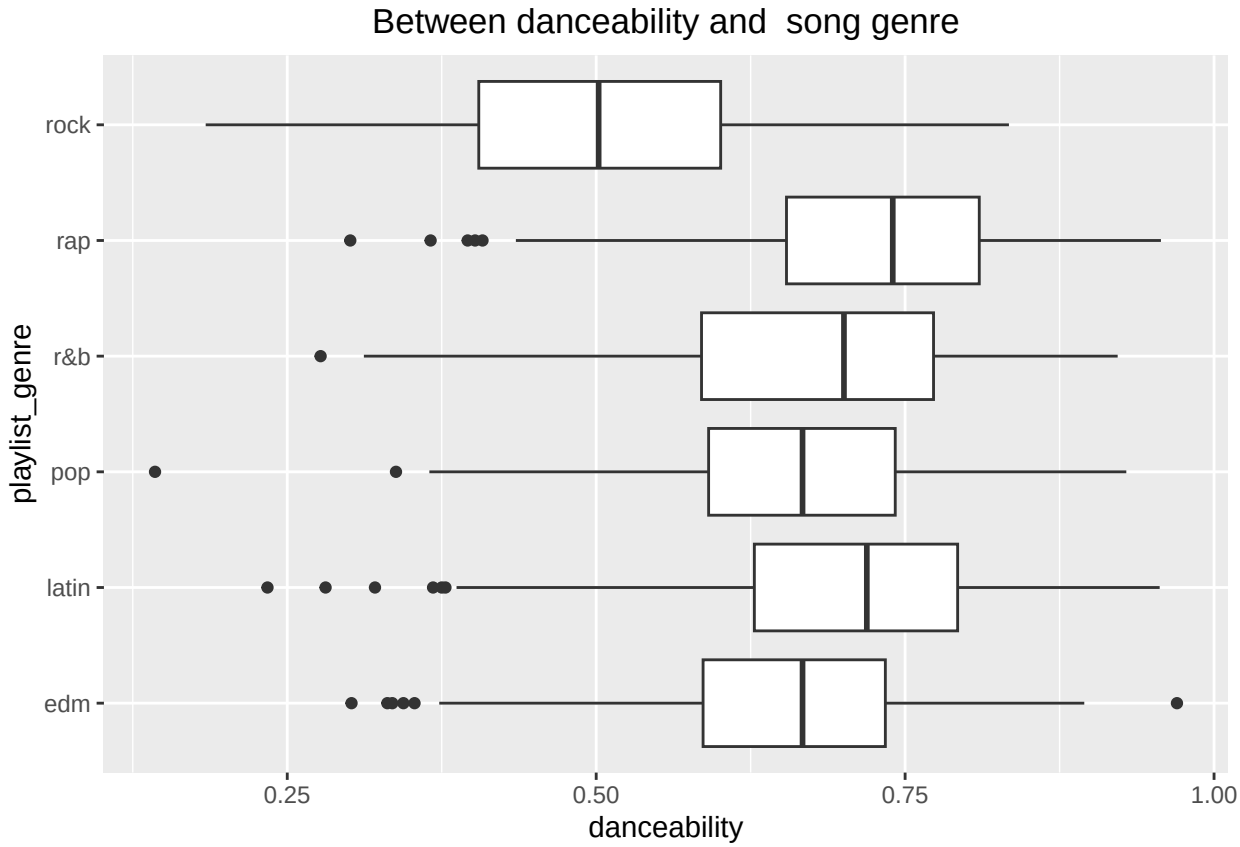
to predict which genre are specialised and not specialised in the particular perspectives

```
library(ggplot2)
ggplot(spotify3, aes(x = tempo , y = playlist_genre)) + geom_boxplot() +
  ggtitle("Between tempo and song genre") +
  theme(plot.title = element_text(hjust = 0.5))
```



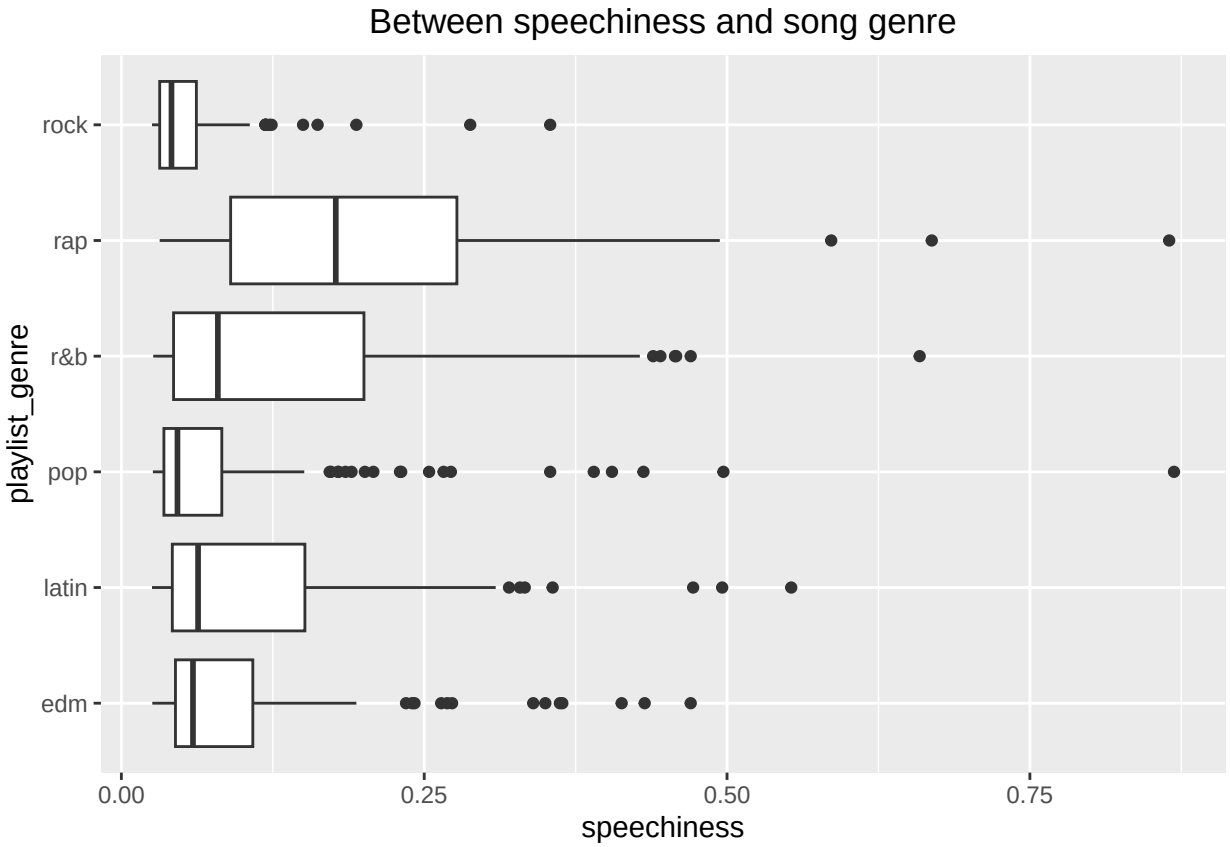
The song genre - egm has the highest tempo. The median of the edm is the highest among all the genres. ### The song genre - latin and r&b have the lowest tempo. The median of the them are the lowest among all the genres.

```
ggplot(spotify3, aes(x = danceability , y = playlist_genre)) + geom_boxplot() +
  ggtitle("Between danceability and  song genre") +
  theme(plot.title = element_text(hjust = 0.5))
```



The song genre - rap has the highest danceability, which the median of the rap is the highest. ### The song genre - rock has the lowest danceability, which the median of the rock is the lowest.

```
ggplot(spotify3, aes(x = speechiness , y = playlist_genre)) +
  geom_boxplot() +
  ggtitle("Between speechiness and song genre") +
  theme(plot.title = element_text(hjust = 0.5))
```

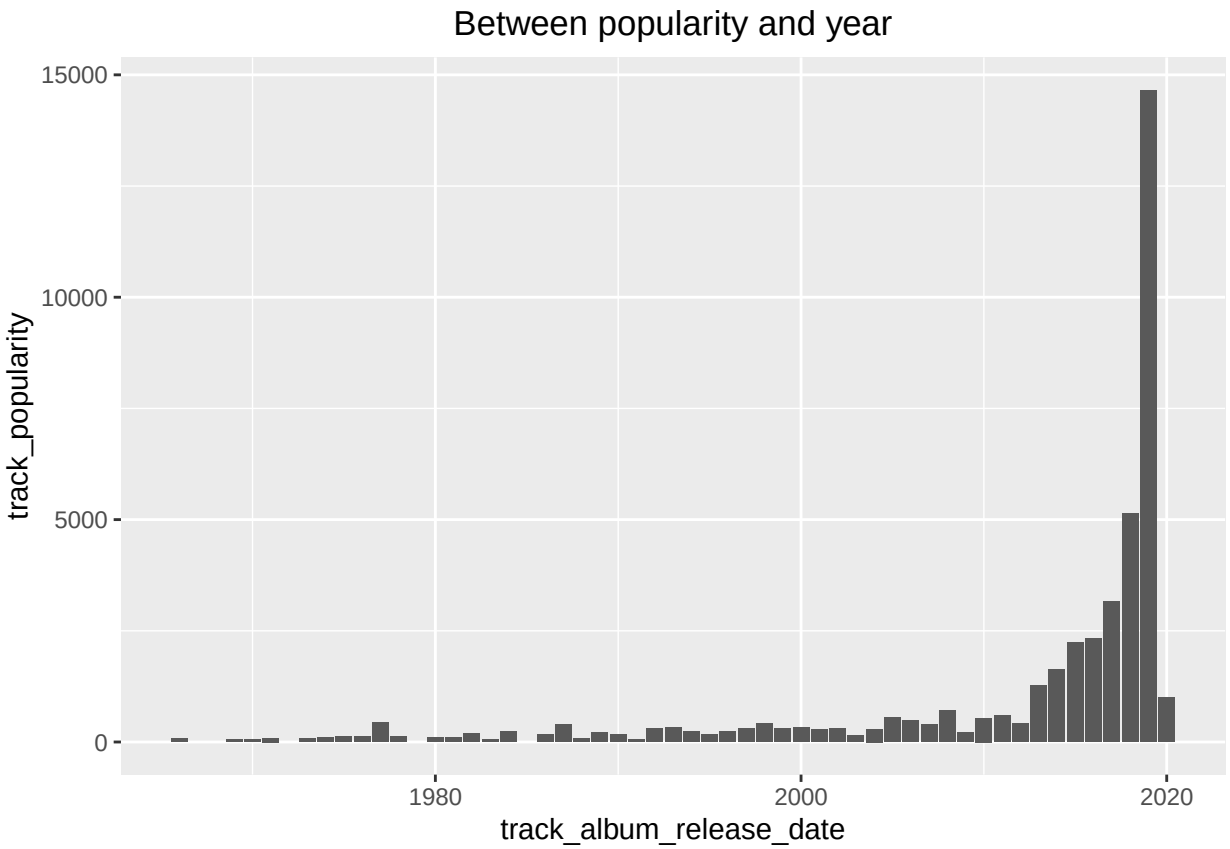


The song genre - rap has the highest speechy, which the median has the highest median. ### The song genre - rock has the lowest speechy, which the median has the lowest median.

2nd plot

Using the variables of year to discover which year of the song is the highest popularity

```
ggplot(spotify3, aes(x=track_album_release_date, y=track_popularity)) +
  geom_col() +
  ggtitle("Between popularity and year") +
  theme(plot.title = element_text(hjust = 0.5))
```

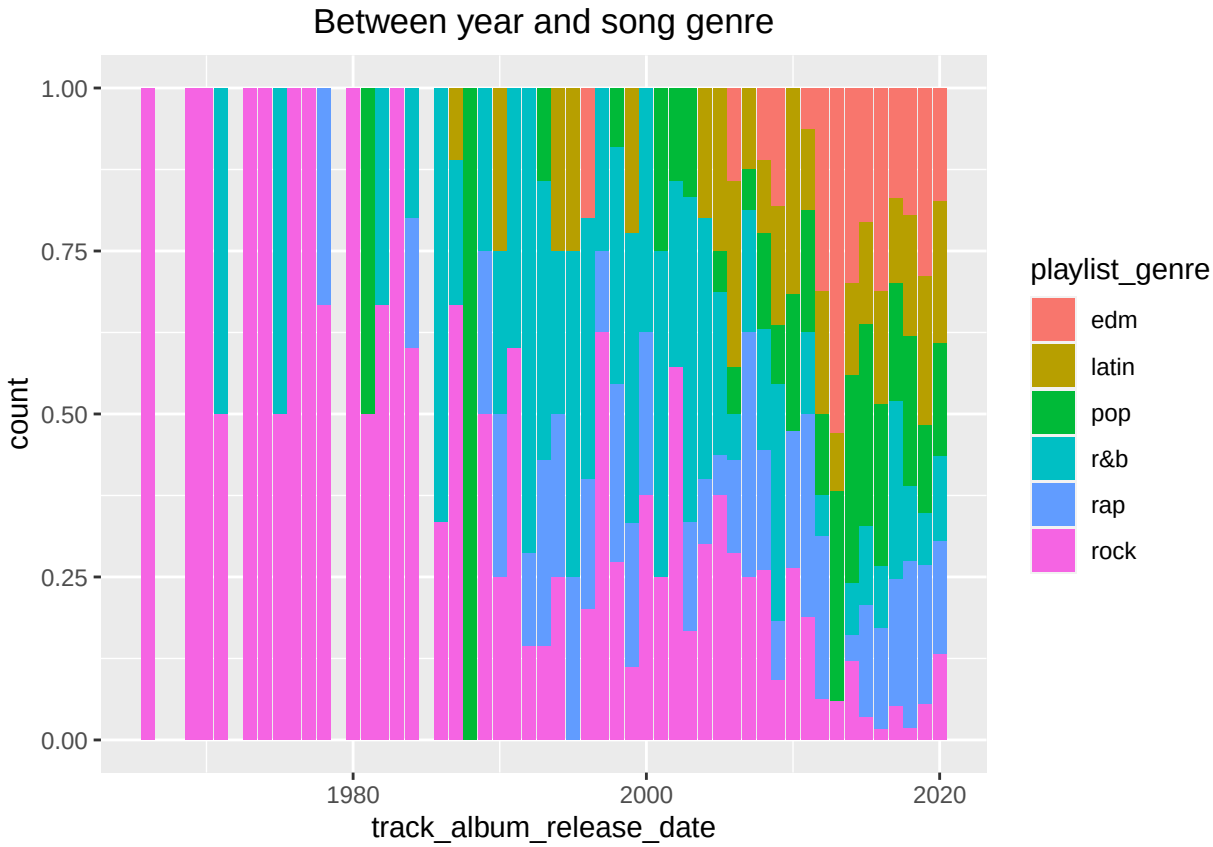


Songs which released in 2019 has the highest popularity

3rd plot

Predict which genre is the domain between the year from 1960 to 2020

```
ggplot(spotify3, aes(x=track_album_release_date, fill=playlist_genre)) +
  geom_bar(position = "fill") +
  ggtitle("Between year and song genre") +
  theme(plot.title = element_text(hjust = 0.5))
```



From 1960 to 1990+, rock has been the domain genre. ### Between 2000 and 2022, r&b has been the domain genre.

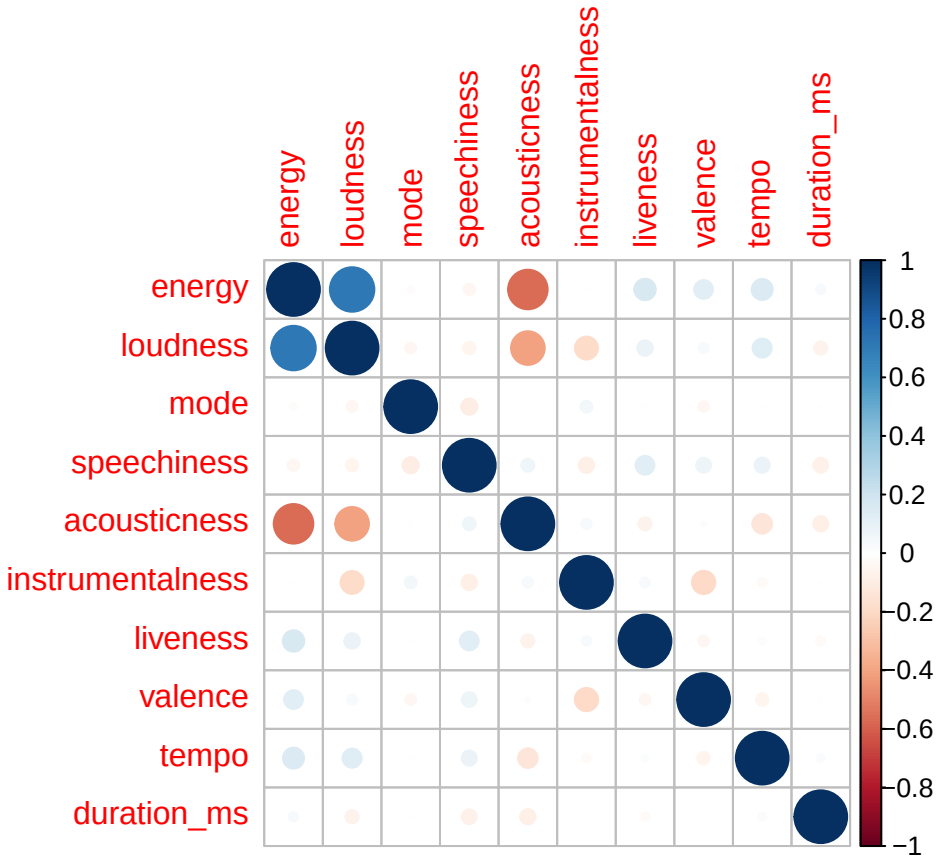
4th plot

To determine between those variables' correlation

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
M=cor(spotify3%>%
      select(energy, loudness, mode, speechiness, acousticness,
             instrumentalness, liveness,
             valence, tempo, duration_ms))
corrplot(M)
```



The loudness and energy has the highest correlation

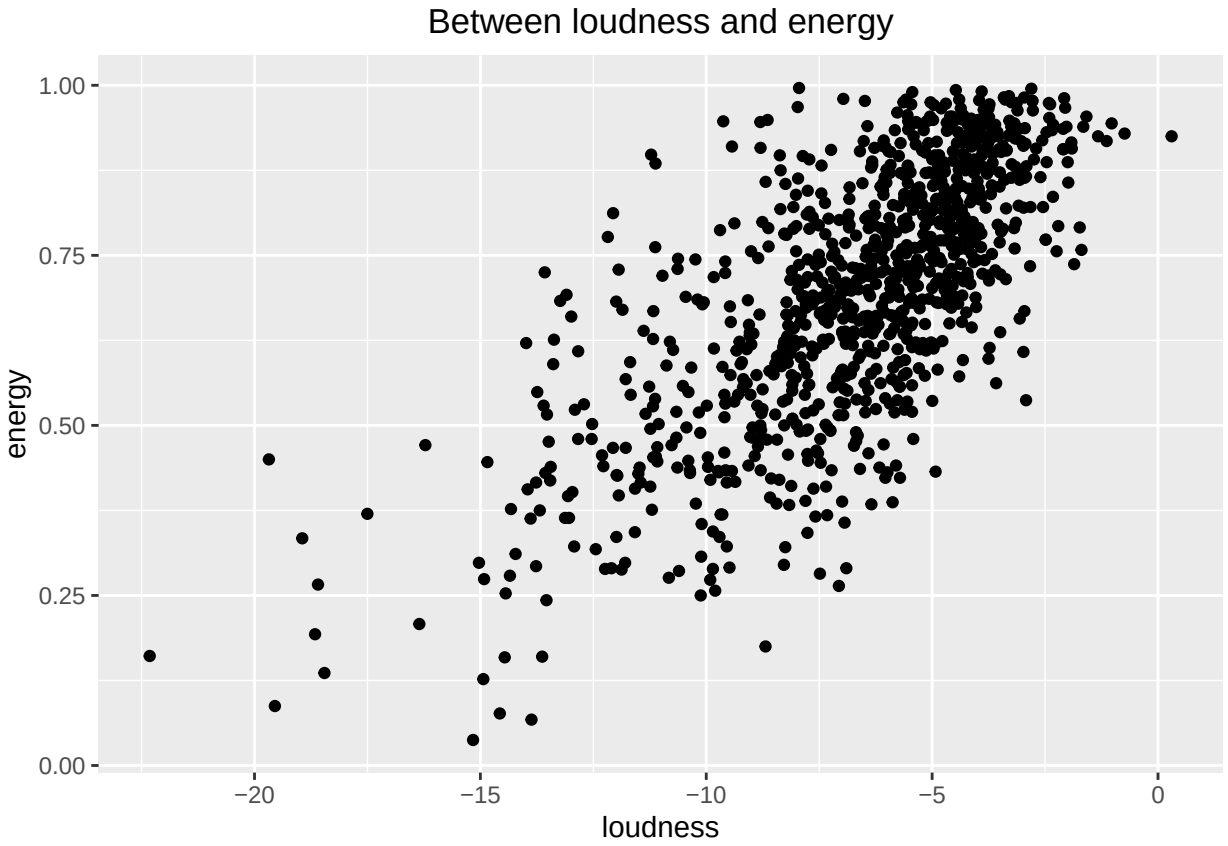
The information will repeat in the model because the higher correlation will induct the higher similarities.

Therefore, having two similar predictors in the model is undesirable.

5th plot

visualize the correlation into different graph

```
ggplot(spotify3, aes(x=loudness, y =energy)) + geom_point() +
  ggtitle("Between loudness and energy") +
  theme(plot.title = element_text(hjust = 0.5))
```



For the confidence interval:

The confidence interval illustrates the degree of uncertainty in the estimate.

Gives us a sense of how closely the results resemble the "actual value" of population exploration.

Typically, confidence intervals are expressed as "95%," which denotes a 95% likelihood that the test results would match the "real value."

3. Use predictive methods to determine trends and correlations for my data sets.

Pearson correlation within all the numerical variables

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.1     v rsample      1.1.0
## v dials        1.1.0     v tune         1.0.1
## v infer        1.0.3     v workflows    1.1.0
## v modeldata    1.0.1     v workflowsets 1.0.0
## v parsnip      1.0.3     v yardstick    1.1.0
## v recipes      1.0.3
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
```

```
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
round(cor(spotify3%>%
        select(energy, loudness, mode, speechiness,
               acousticness, instrumentalness, liveness,
               valence, tempo, duration_ms)), digits = 3)
```

```
##                  energy loudness   mode speechiness acousticness
## energy            1.000    0.711 -0.017      -0.047       -0.562
## loudness          0.711    1.000 -0.042      -0.055       -0.407
## mode             -0.017   -0.042  1.000      -0.095       -0.003
## speechiness      -0.047   -0.055 -0.095       1.000        0.060
## acousticness     -0.562   -0.407 -0.003       0.060        1.000
## instrumentalness -0.002   -0.190  0.050      -0.085        0.039
## liveness          0.165    0.087  0.005       0.127       -0.063
## valence           0.127    0.035 -0.040       0.079        0.010
## tempo             0.158    0.130 -0.008       0.083       -0.139
## duration_ms       0.031   -0.064  0.002      -0.079       -0.084
##                  instrumentalness liveness valence  tempo duration_ms
## energy                     -0.002    0.165   0.127  0.158       0.031
## loudness                   -0.190    0.087   0.035  0.130      -0.064
## mode                        0.050    0.005  -0.040 -0.008       0.002
## speechiness                -0.085    0.127   0.079  0.083      -0.079
## acousticness                0.039   -0.063   0.010 -0.139      -0.084
## instrumentalness            1.000    0.032  -0.194 -0.029      -0.001
## liveness                    0.032    1.000  -0.042  0.017      -0.029
## valence                    -0.194   -0.042   1.000 -0.055      -0.005
## tempo                      -0.029    0.017  -0.055  1.000       0.021
## duration_ms                -0.001   -0.029  -0.005  0.021       1.000
```

The varaible loudness and energy have the strongest correlation, which both of them are 0.690 pearson correlation between each other.

Data Splitting and Preprocessing

```
set.seed( 1223 )
set.seed( 1210 )
heart_split <- initial_split( spotify3, strata = playlist_genre )
heart_train <- training( heart_split )
heart_test <- testing( heart_split )
```

```
# create a recipe
spotify_recipe <- recipe(playlist_genre~., data = heart_train ) %>%
  step_zv(all_predictors()) %>%
  step_normalize( all_numeric() ) %>%
  step_corr(all_numeric()) %>%
  prep()

spotify_recipe
```

```
## Recipe
##
## Inputs:
##
```

```
##       role #variables
##    outcome          1
##  predictor         14
##
## Training data contained 747 data points and no missing data.
##
## Operations:
##
## Zero variance filter removed <none> [trained]
## Centering and scaling for track_popularity, track_album_release_date, dan... [trained]
## Correlation filter on <none> [trained]
```

```
spotify_train_preproc <- juice( spotify_recipe )
spotify_test_preproc <- bake( spotify_recipe, heart_test )

spotify_train_preproc %>%
  skim_without_charts()
```

Table 4: Data summary

Name	Piped data
Number of rows	747
Number of columns	15
Column type frequency:	
factor	1
numeric	14
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
playlist_genre	0	1	FALSE	6	edm: 152, rap: 135, pop: 126, lat: 125

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
track_popularity	0	1	0	1	-1.69	-0.81	0.15	0.83	2.02
track_album_release_date	0	1	0	1	-4.61	-0.15	0.45	0.65	0.75
danceability	0	1	0	1	-3.20	-0.56	0.14	0.73	2.13
energy	0	1	0	1	-3.55	-0.65	0.12	0.78	1.61
key	0	1	0	1	-1.45	-0.89	0.21	0.76	1.58
loudness	0	1	0	1	-4.33	-0.47	0.19	0.72	1.91
mode	0	1	0	1	-1.22	-1.22	0.82	0.82	0.82
speechiness	0	1	0	1	-0.80	-0.66	-0.44	0.30	6.60
acousticness	0	1	0	1	-0.80	-0.74	-0.43	0.38	3.26
instrumentalness	0	1	0	1	-0.40	-0.40	-0.40	-0.37	3.45
liveness	0	1	0	1	-1.08	-0.63	-0.40	0.36	5.08
valence	0	1	0	1	-1.97	-0.79	-0.06	0.79	2.05
tempo	0	1	0	1	-2.16	-0.75	0.02	0.36	3.68

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
duration_ms	0	1	0	1	-2.67	-0.62	-0.14	0.49	4.71

Tune and fit a model

Model Fitting for two different models: classification trees and random forest with Classification and cross-validation

Model Fitting 1: Classification trees

```
class_tree_spec <- decision_tree( mode = "classification") %>%
  set_engine( "rpart" )
```

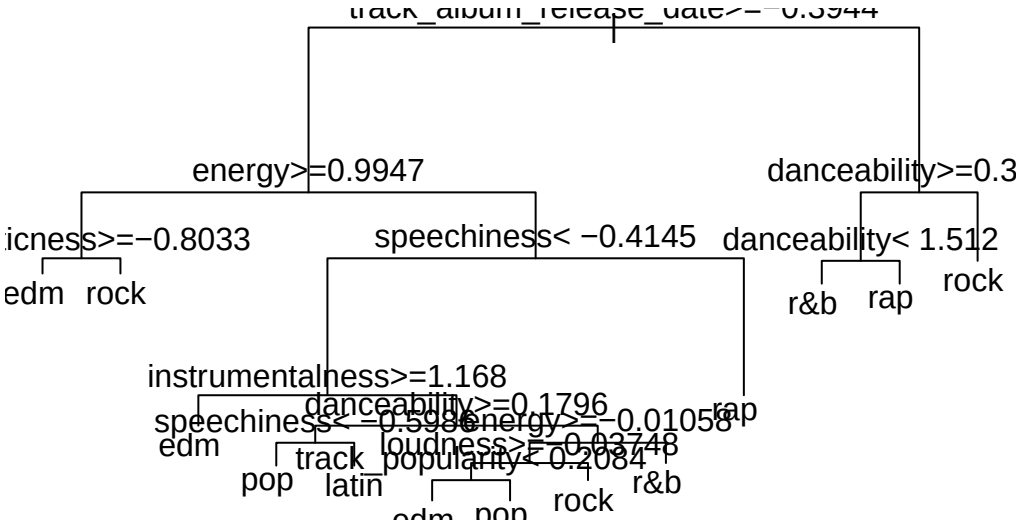
```
set.seed( 1012 )
heart_cv <- vfold_cv( spotify_train_preproc, v = 5, strata = playlist_genre )

heart_tree <- class_tree_spec %>%
  fit( playlist_genre ~ . , data = spotify_train_preproc )

heart_tree
```

```
## parsnip model object
##
## n= 747
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 747 595 edm (0.2 0.17 0.17 0.15 0.18 0.13)
##     2) track_album_release_date>=-0.3943584 585 436 edm (0.25 0.19 0.2 0.11 0.19 0.055)
##       4) energy>=0.9946958 105  37 edm (0.65 0.067 0.11 0.029 0.048 0.095)
##         8) acousticness>=-0.8032734 97  29 edm (0.7 0.072 0.1 0.031 0.052 0.041) *
##         9) acousticness< -0.8032734 8   2 rock (0 0 0.25 0 0 0.75) *
##       5) energy< 0.9946958 480 373 rap (0.17 0.21 0.22 0.13 0.22 0.046)
##        10) speechiness< -0.4144567 241 170 pop (0.2 0.24 0.29 0.11 0.071 0.087)
##          20) instrumentalness>=1.168334 30  14 edm (0.53 0.067 0.13 0 0.17 0.1) *
##          21) instrumentalness< 1.168334 211 144 pop (0.15 0.26 0.32 0.13 0.057 0.085)
##            42) danceability>=0.1795618 86  52 latin (0.081 0.4 0.37 0.058 0.081 0.012)
##              84) speechiness< -0.5985546 57  31 pop (0.12 0.3 0.46 0.07 0.035 0.018) *
##              85) speechiness>=-0.5985546 29  12 latin (0 0.59 0.21 0.034 0.17 0) *
##            43) danceability< 0.1795618 125  90 pop (0.2 0.17 0.28 0.18 0.04 0.14)
##              86) energy>=-0.01058421 62  40 pop (0.29 0.16 0.35 0.016 0.032 0.15)
##               172) loudness>=-0.03748387 55  33 pop (0.33 0.15 0.4 0.018 0.036 0.073)
##                 344) track_popularity< 0.2084196 26  12 edm (0.54 0.12 0.23 0 0.038 0.077) *
##                 345) track_popularity>=0.2084196 29  13 pop (0.14 0.17 0.55 0.034 0.034 0.069) *
##               173) loudness< -0.03748387 7   2 rock (0 0.29 0 0 0 0.71) *
##              87) energy< -0.01058421 63  42 r&b (0.11 0.17 0.21 0.33 0.048 0.13) *
##        11) speechiness>=-0.4144567 239 149 rap (0.14 0.19 0.14 0.15 0.38 0.0042) *
##     3) track_album_release_date< -0.3943584 162  94 rock (0.019 0.099 0.062 0.26 0.14 0.42)
##       6) danceability>=0.3287676 63  34 r&b (0.016 0.13 0.063 0.46 0.3 0.032)
##        12) danceability< 1.512241 53  24 r&b (0.019 0.13 0.075 0.55 0.19 0.038) *
##        13) danceability>=1.512241 10   1 rap (0 0.1 0 0 0.9 0) *
##       7) danceability< 0.3287676 99  33 rock (0.02 0.081 0.061 0.13 0.04 0.67) *
```

```
plot( heart_tree$fit )
text( heart_tree$fit, pretty = 0 )
```



Explanation of the classification tree

From the top, it is the overall probability of the track_album_release_date.

It implies the proportion of the track_album_release_date that the year of the song are -2.40%.

The second level of the classificantion trees are acousticness, which the proportion are -0.70%, or otherwise the song of the genre will be rock.

The thid level of the classificantion trees are danceability and speechiness, account for the -0.76% and 0.94% respectively.

From the third level, if the classificantion trees are speechiness less than 0.94%, it will be identify as the rap genre.

From the third level of the trees of the danceability, if it is large or equal to -0.7596%, as well as the proportion of tempo is large or equal to 0.082%, it will be identified as the edm (35%) and pop (31%) genre.

From the third level of the trees of the speechiness, if the speechiness is less than 0.94%, it will split two braches to identify the song genre; one of the braches will be the rap genre; one of the proportions of track_album_release_date is -0.2403163% will be continually to predict the song genre.

#As above, Assuming that the song of the tempo are less or equal to -0.2074%, it wil be 48% be the r&b and 40% will be the rock genre.

If the song of the valence is large or equal to -0.132%, it will be identify as the latin genre.

Or else, if the song also meets the variable of duration_ms is large or equal to 1.022%, it will be identical to the rap genre.

Or else, if the song also meet the variable of duration_ms are large or equal to 1.022% and the energy are large or equal to -0.0606%, it will be identify as the EDM genre or the r&b genre.

```
heart_resamples <- fit_resamples( object = class_tree_spec ,
                                  preprocessor = recipe(playlist_genre ~ ., data = spotify3),
                                  resamples = heart_cv )
heart_resamples %>%
  collect_metrics()
```

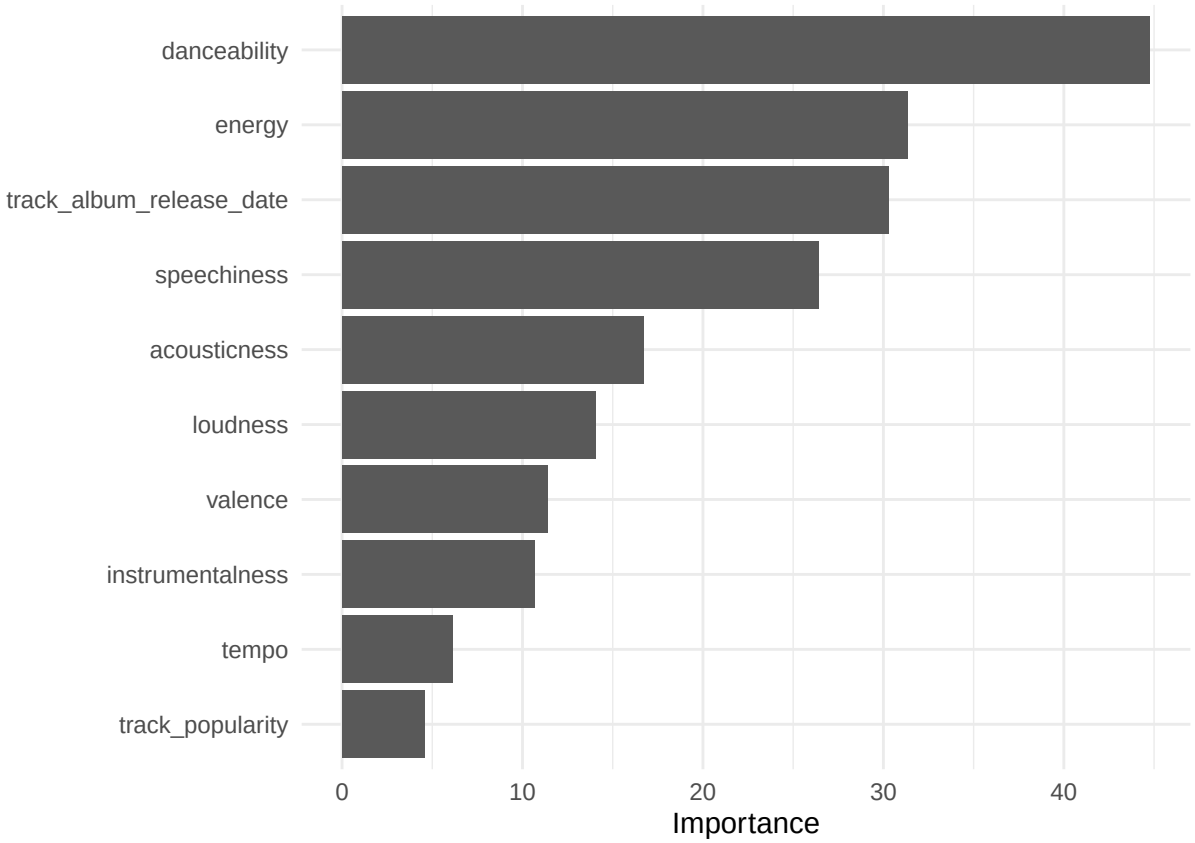
```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy multiclass 0.419     5 0.00739 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.735     5 0.00938 Preprocessor1_Model1
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
heart_tree %>%
  vip( num_features = 10 ) + theme_minimal()
```




```
heart_preds <- heart_tree %>%
  predict( new_data = heart_test, type = "class" ) %>%
  bind_cols( heart_test %>%
               select( playlist_genre ) )
```

```
# The track_ablum_release_date are the most important variable in the classification tree's prediction.
```

```
# the confusion matrix of the prediction
heart_preds %>%
  conf_mat( truth = playlist_genre , estimate = .pred_class)
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm     0     0   0   0   0    0
##      latin   0     0   0   0   0    0
##      pop     0     0   0   0   0    0
##      r&b     0     0   0   0   0    0
##      rap    51    42  43  37  46   34
##      rock    0     0   0   0   0    0
```

For the confusion matrix, it looks not reasonable and well in the confusion matrix because the figures are domained in the rap genre.

```
# calculate the sensitivity, specificity and accuracy
sensitivity( heart_preds, truth = playlist_genre, estimate = .pred_class ) %>%
  bind_rows( specificity( heart_preds, truth = playlist_genre, estimate = .pred_class ),
             accuracy( heart_preds, truth = playlist_genre, estimate = .pred_class ) )
```

```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity macro          0.167
## 2 specificity macro          0.833
## 3 accuracy    multiclass     0.182
```

The sensitivity, specificity and accuracy of the prediction are 16.67%, 83.33% and 17.39% respectively.

To a small conclusion, this model could be better. For the classification trees since the confusion matrix looks unreasonable, and the sensitivity, specificity and accuracy results could be higher. One of the reason is may the track_ablum_release_date are the most critical variable in the classification tree's prediction. The model cannot analyse another factor of the song, such as danceability, energy or loudness.

Model Fitting 2: Random forests

```
set.seed( 1223 )
# create a random forest
rf_spec <- rand_forest( mode = "classification", mtry = tune(), trees = 100,
  min_n = tune() ) %>%
  set_engine( "ranger", importance = "permutation"  )
rf_spec
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 100
##   min_n = tune()
##
```

```
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

```
rand_spec_grid <- grid_regular(
  finalize( mtry(),
            spotify_train_preproc %>%
               dplyr::select( -playlist_genre ) ), min_n(),levels = 5 )
 rand_spec_grid
```

```
## # A tibble: 25 x 2
##     mtry min_n
##    <int> <int>
##  1     1     2
##  2     4     2
##  3     7     2
##  4    10     2
##  5    14     2
##  6     1    11
##  7     4    11
##  8     7    11
##  9    10    11
## 10    14    11
## # ... with 15 more rows
```

```
 set.seed(1234)
office_boost <- bootstraps(spotify_train_preproc, times = 5)
office_boost
```

```
## # Bootstrap sampling
## # A tibble: 5 x 2
##   splits            id
##   <list>            <chr>
## 1 <split [747/282]> Bootstrap1
## 2 <split [747/285]> Bootstrap2
## 3 <split [747/269]> Bootstrap3
## 4 <split [747/272]> Bootstrap4
## 5 <split [747/274]> Bootstrap5
```

```
doParallel::registerDoParallel()
set.seed( 1959 )
rf_spec
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 100
##   min_n = tune()
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

```
rf_grid <- tune_grid( object = rf_spec,
                      preprocessor = recipe(playlist_genre ~ .
                                            , data = spotify_train_preproc),
                      resamples = office_boost,
                      grid = rand_spec_grid )
```

```
best_rf_rmse <- select_best( rf_grid, "accuracy" )
best_rf_rmse
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     4    11 Preprocessor1_Model07
```

The best parameter is 21.

```
final_rf <- finalize_model( rf_spec, best_rf_rmse )
final_rf
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 4
##   trees = 100
##   min_n = 11
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

```
final_model <- final_rf %>%
  fit(playlist_genre ~ . , data = spotify_train_preproc)

# The final model of the random forest
final_model
```

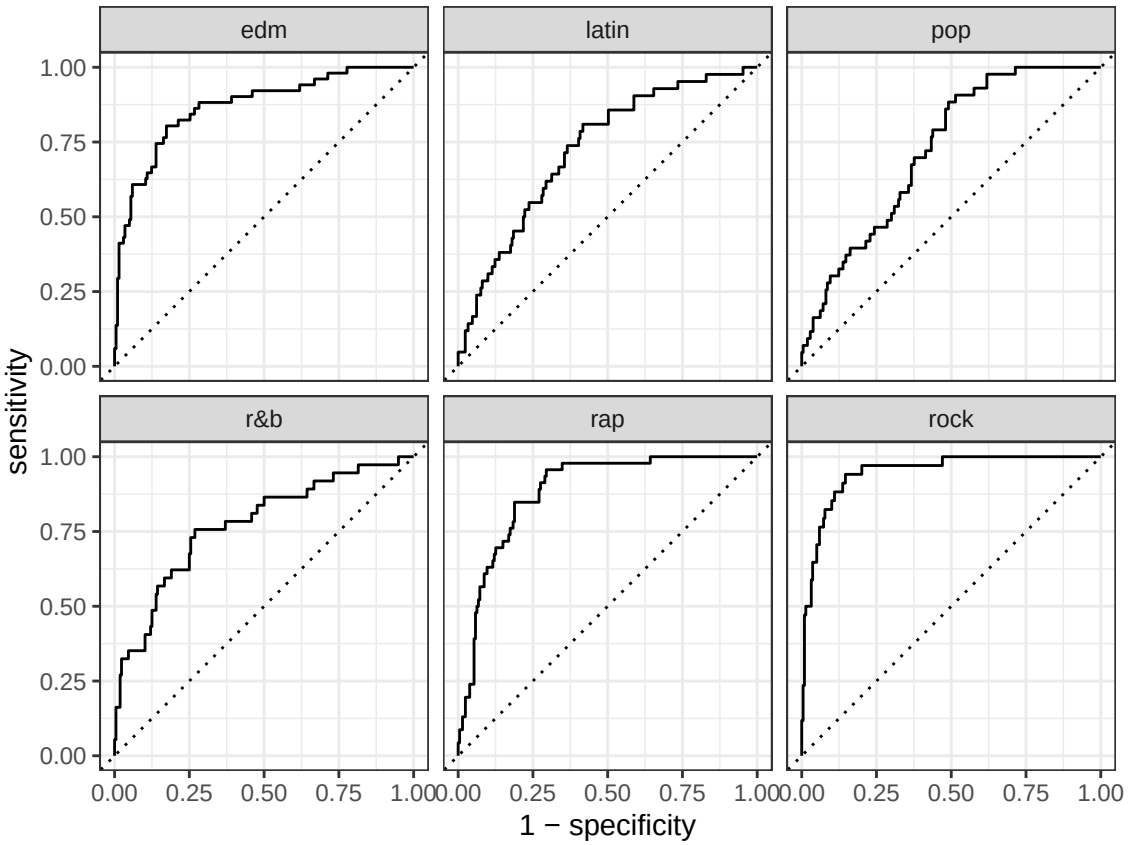
```
## parsnip model object
##
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~4L,      x), num.trees = ~100, min.
##
## Type:                             Probability estimation
## Number of trees:                  100
## Sample size:                      747
## Number of independent variables:  14
## Mtry:                             4
## Target node size:                 11
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.4716178
```

```
# ROC Curves
spotify3_preds <- predict( object = final_model ,
                           new_data = spotify_test_preproc,
                           type = "prob")

spotify3_preds
```

```
## # A tibble: 253 x 6
##    .pred_edm .pred_latin .pred_pop `.pred_r&b` .pred_rap .pred_rock
##        <dbl>       <dbl>     <dbl>       <dbl>     <dbl>      <dbl>
##  1    0.217       0.292     0.232       0.0732    0.114     0.0720
##  2    0.802       0.0346    0.0846      0.0385    0.0298    0.0103
##  3    0.136       0.104     0.126       0.201     0.207     0.226
##  4    0.122       0.179     0.320       0.0819    0.0925    0.205
##  5    0.0189      0.198     0.0270      0.180     0.567     0.00916
##  6    0.408       0.0752    0.105       0.0856    0.273     0.0540
##  7    0.0360      0.224     0.122       0.362     0.108     0.148
##  8    0.0559      0.122     0.150       0.236     0.197     0.239
##  9    0.850       0.0341    0.0509      0.0374    0.0266    0.00143
## 10    0.0949      0.0916    0.200       0.101     0.478     0.0344
## # ... with 243 more rows
```

```
spotify3_preds %>%
  roc_curve( truth = spotify_test_preproc$playlist_genre,
             estimate = .pred_edm, .pred_latin,
            .pred_pop, '.pred_r&b', .pred_rap, .pred_rock)%>%
  autoplot()
```



From the six graphs, the genre of rock is the best genre and the genre of pop is the worst.

```
spotify3_preds_class <- predict( object = final_model ,
                          new_data = spotify_test_preproc)%>%bind_cols(true=spotify_test_preproc$playli

# the confusion matrix
spotify3_preds_class%>%conf_mat(truth = true, estimate = .pred_class )
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm    32     5  11   2   4    0
##      latin   5    13  12   4   1    1
##      pop     7    11  12   2   2    6
##      r&b     3     5   1  13   7    2
##      rap     3     7   1   8  30    0
##      rock    1     1   6   8   2   25
```

From the confusion matrix, it saying the prediction in the random forest performances better since the matrix looks reasonable than the classification trees.

```
categorical_metrics <- metric_set(sensitivity, specificity,precision)
outcome<-spotify3_preds_class %>%
  categorical_metrics (truth = true, estimate =.pred_class)

outcome
```

```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity macro          0.492
## 2 specificity macro          0.899
## 3 precision   macro          0.478
```

The sensitivity, specificity and accuracy of the prediction are 45.384%, 89.12% and 47.23% respectively.

Compared to the results of the classification tree, the sensitivity, specificity and accuracy are also better than the classification

```
spotify3_preds %>%
  roc_auc( truth = spotify_test_preproc$playlist_genre, estimate = .pred_edm, .pred_latin,
           .pred_pop, '.pred_r&b', .pred_rap, .pred_rock)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.821
```

The AUC is 80%, which implies that it has 80% correct in the model. Therefore, we determine to use the random forest as our prediction model because it is better than the classification trees.