Final Report

Beatrice Febrina

2023-04-20

Executive Summary

Spotify is an audio streaming and media services provider with over 365 million monthly active users, including 165 million paying subscribers. As the world's largest music streaming provider, they are interested in how year, tempo, danceability and speechiness factors can predict a song's genre to enhance customer's experience.

After carrying a full analysis and building a model using The random forest with 100 trees and 5 levels and the data provided, we can conclude that although the model is pretty accurate in predicting the true negatives (specificity), the model is not very accurate in predicting the true positive (sensitivity). The model's senstivity can be improved slightly by using more variables as predictors in predicting the genre. However, such improvement may still consider the model to be inaccurate.

Methods

After reducing the spotify data to 6000 observations (1000 observations per genre). The spotify data with variable year, speechiness, danceability, tempo, popularity, energy, key, loudness, acousticness, instrumentalness, liveness, valence, duration_ms and mode is used to model the song's genre prediction. In this report we will be looking at how well the following factors can predict song genre:

- Year
- Tempo
- Danceability
- Speechiness

Additionally, another prediction using all relevant variables including mode, popularity, energy, key, loudness, acousticness, instrumentalness, liveness, valence and duration in ms is also used to show how well the model can predict songs with just the 4 above factors compared to using more factors.

The following analysis was completed in R version 4.01.1 using the tidyverse, tidymodels and dplyr packages.

The steps carried out to build the model is:

- Split and Preprocessing
- Model Fitting and Tuning
- Model Evaluation
- Model Selection

In the splitting process, intial_split is used from the rsample package. The data is split into 4500 observations in the trainiang set and 1500 observations in the testing set as seen in the appendix (split section). Data splitting will help with avoiding over fitting and the evaluation of models.

Preprocessing is the generalised term for performing mathematical operations on your data before modelling it to improve the predictive power of the model. Genre is the outcome variable and 4 preprocessing steps are carried out to improve the model:

- step_zv: remove predictors that have 0 variance

- step_dummy: to convert categorical variables to dummy variables
- step_normalise: improve model performance by normalising all predictors to have a mean of 0 and standard deviation of 1
- step_corr: remove highly correlated predictor variable.

The preprocessed training data can be seen in appendix (preprocessing section) after juicing the recipe.

As consulted by the statistician, 3 models would be compared for predicting on this dataset:

- The Linear discriminant analysis (LDA)
- The K-nearest neighbours model with a range of 1 to 100 and 20 levels (KNN)
- The random forest with 100 trees and 5 levels (RF)

The KNN and RF models need to be tuned to ensure that it performs at its best. This process is done by tuning the models with the 10-fold cross validation set from the preprocessed tuning data and a grid of ranges. Using the collect_metrics function, best roc_auc is chosen to determine the best parameters for the models.

Results

Explanatory Analysis

Does the popularity of songs differ between genres?

There seems to be a difference between popularity across different genres. Based on figure 1, EDM has the lowest median popularity at around 37 while pop has the highest median popularity at around 52. The other genres (latin, rock, rap r&b) has similar median popularity at between 40-50. Besides that, rock has the largest IQR compared to other genres, which means that "rock" songs has more variability in terms of popularity. While rap has the smallest IQR which can indicate less variability.

Is there a difference in speechiness for each genre?

There seems to be similar median of speechiness across all genres except rap as seen in figure 2. Latin, rock, r&b, EDM and rock has a median speechiness approximately at 0.06 with rock having the slightly lower speechiness compared to the other genres at around 0.04-0.05. However, rap has the highest speechiness by a considerable amount at around 0.17. This is predictable considering rap songs has more lyrics compared to the other genres.

How does track popularity change over time?

There seems to be more or less constant popularity between 1960 - 2000 with slight increase to around 50 from 40 in 1980 as seen in figure 3. In 2010s, However, there is a significant drop in popularity to around 30. Looking at figure 4, r&b and edm dropped the most compared to other genres in 2010 to around 25 and pop has the lowest drop at only 50 to 43 from 1980 to 2010. With this sudden drop in 2010, there must be economic downturns on the music industry that might affect this dropp across all genres. However, the popularity for all genres quickly pick up from 2010 to 2020 except for rock. Popularity for all genres rose to around 55 in 2020. Though genres such as latin, pop, r&b, and rap follow this trend (seen in figure 4), edm's popularity fall slightly short at only 45, while rock does not follow the trend at all. Rock's popularity peaked at around 1985 and it has been dropping ever since with only around 37 popularity in 2020.

Model Evaluation

Table 1 shows the AUC and accuracy of all the 3 models. The RF model with mtry of 4, 100 trees and min_n of 40 has the highest AUC at 0.846. At this parameter, the accuracy is also better compared to other models at 0.549. An AUC of 0.846 means that the RF classifier works well and are able to distinguish between the correct genre and the incorrect genre.

The model fitting is done twice. The first one with genre as the outcome variable and year, tempo, danceability and speechiness as the predictor. The second one with genre as the outcome variable and all variables as the predictor. Looking at figure 5, all predictors have an importance of over 0.05. Year having the highest

importance at around 0.094 followed by tempo at 0.085 and danceability at 0.059 and finally speechiness at around 0.05.

Figure 6 shows that the 4 variable above are the most important predictor. However the importance of tempo dropped significantly to around 0.06 and Year becoming the most important predictor quite significantly compared to other variables.

Thus the variable year, tempo, danceability and speechiness are strong predictors in predicting the genre of the songs compared to other variables.

Discussion

Based on the model's outcomes with just the 4 predictors, the comparison between the predicted genre and the actual genre shows only a 0.461 accuracy. Table 2 also shows that the sensitivity is very low at 0.463 which means that if the actual genre is "rock" (it could be edm or pop or etc.), we are correctly predicting them as "rock" approximately 47.1% of the time. However, the model has far better specificity at 0.894. Which means that if the actual genre is not "rock" we are correctly predicting them as not "rock" approximately 89.4% of the time.

Table 3 shows how the sensitivity and the specificity would be if we fit all variables as predictors to the model. It shows that the specificity stays the same however the sensitivity improved to 0.551. This means that given the actual genre is "rock" and the model predicting it correctly as "rock" 54.8% of the time, which is an 7.7% increase.

This means that using the factors year, tempo, danceability and speechiness is not very accurate in correctly predicting the correct genre. Though using more variables as predictor increases the accuracy, it still would not make the model very accurate. However the model is pretty accurate at correctly predicting the wrong genre.

Additionally, after looking at the confusion matrix, we can also conclude that identifying "rock" songs has the highest accuracy for both prediction model (with just 4 predictors and all predictors) and "r&b" and pop" songs has the lowest accuracy ("r&b" lowest with just 4 predictors and "pop" lowest with all predictors).

Conclusion

After performing analysis of the data, data variables required as a predictor in the model built to predict a song's genre are year, speechiness, danceability, tempo, popularity, energy, key, loudness, acousticness, instrumentalness, liveness, valence, duration_ms. Additionally, after assessing which model to choose between LDA, KNN and RF, after tuning and fitting the model, RF with 100 trees and 5 levels has the best AUC. Thus, that model is used to predict the song genre using year, tempo, speechiness and danceability variable as the predictor. Year is the most important predictor in predicting the genre followed by tempo, danceability and speechiness. Results of the prediction using the model used however do not provide the most accurate prediction as sensitivity is only 0.463. Sensitivity can be improved by using all the variables as predictors in predicting the genre. However, it only improve the sensitivity by 0.077. Nevertheless, some genres are more easily distinguised and would provide more accurate predictions such as "rock". "Pop" songs however is the least accurate in prediction. Thus, this model may not be very effective in predicting a song's genre especially some genres more than others.

Appendix

```
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.
```

```
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, ...
```

Data Cleaning

```
spotify_songs <- spotify_songs %>% drop_na()

spotify_songs <- spotify_songs %>%
  rename(genre = playlist_genre)  %>%
  rename(popularity = track_popularity) %>%
  rename(date = track_album_release_date)

spotify_songs <- spotify_songs %>% mutate(year = lubridate::year(ymd(date)))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `year = lubridate::year(ymd(date))`.
## Caused by warning:
## !  1886 failed to parse.
```

```
spotify_songs <- spotify_songs %>% drop_na()

spotify_songs <- spotify_songs %>%
  mutate(year = as.integer(year),
         key = as.integer(key),
         genre = as.factor(genre),
         mode = as.factor(mode),
         popularity = as.integer(popularity))
```

```r
spotify_new <- subset(spotify_songs, select = -c(track_id, track_name, track_artist, track_album_id, tra

spotify_new <- spotify_new %>% relocate(genre,year,speechiness,danceability,tempo,mode)

spotify_new %>% count(genre)
```

```
## # A tibble: 6 x 2
##   genre     n
##   <fct> <int>
## 1 edm    5969
## 2 latin  4961
## 3 pop    5303
## 4 r&b    5094
## 5 rap    5468
## 6 rock   4147
```

```r
set.seed(1879186)
spotify_final <- spotify_new %>%
  group_by(genre) %>%
  sample_n(1000)

spotify_final <- spotify_final %>% ungroup(genre)

skim_without_charts(spotify_final)
```

Table 1: Data summary

Name	spotify_final
Number of rows	6000
Number of columns	15
Column type frequency:	
factor	2
numeric	13
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
genre	0	1	FALSE	6	edm: 1000, lat: 1000, pop: 1000, r&b: 1000
mode	0	1	FALSE	2	1: 3404, 0: 2596

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
year	0	1	2011.40	11.31	1958.00	2009.00	2016.00	2019.00	2020.00
speechiness	0	1	0.11	0.10	0.02	0.04	0.06	0.13	0.88
danceability	0	1	0.66	0.15	0.10	0.56	0.67	0.76	0.98

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
tempo	0	1	120.64	26.92	37.11	99.56	120.68	134.01	206.68
popularity	0	1	43.21	24.82	0.00	26.00	46.00	63.00	99.00
energy	0	1	0.70	0.18	0.04	0.58	0.72	0.84	1.00
key	0	1	5.31	3.59	0.00	2.00	6.00	8.00	11.00
loudness	0	1	-6.71	2.99	-36.62	-8.14	-6.15	-4.64	1.27
acousticness	0	1	0.18	0.22	0.00	0.02	0.08	0.26	0.98
instrumentalness	0	1	0.08	0.22	0.00	0.00	0.00	0.01	0.99
liveness	0	1	0.19	0.15	0.01	0.09	0.13	0.24	0.98
valence	0	1	0.51	0.23	0.00	0.33	0.51	0.69	0.99
duration_ms	0	1	225572.07	59853.30	31875.00	187412.00	215426.50	253503.00	517125.00

Explanatory Analysis

```
# Does the popularity of songs differ between genres?

ggplot(spotify_final, aes(x = genre, y = popularity)) +
  geom_boxplot(aes(fill = genre)) +
  labs(caption = "Figure 1: Side-by-side boxplot of popularity against genre")+
  theme(plot.caption = element_text(hjust = 0.5))
```

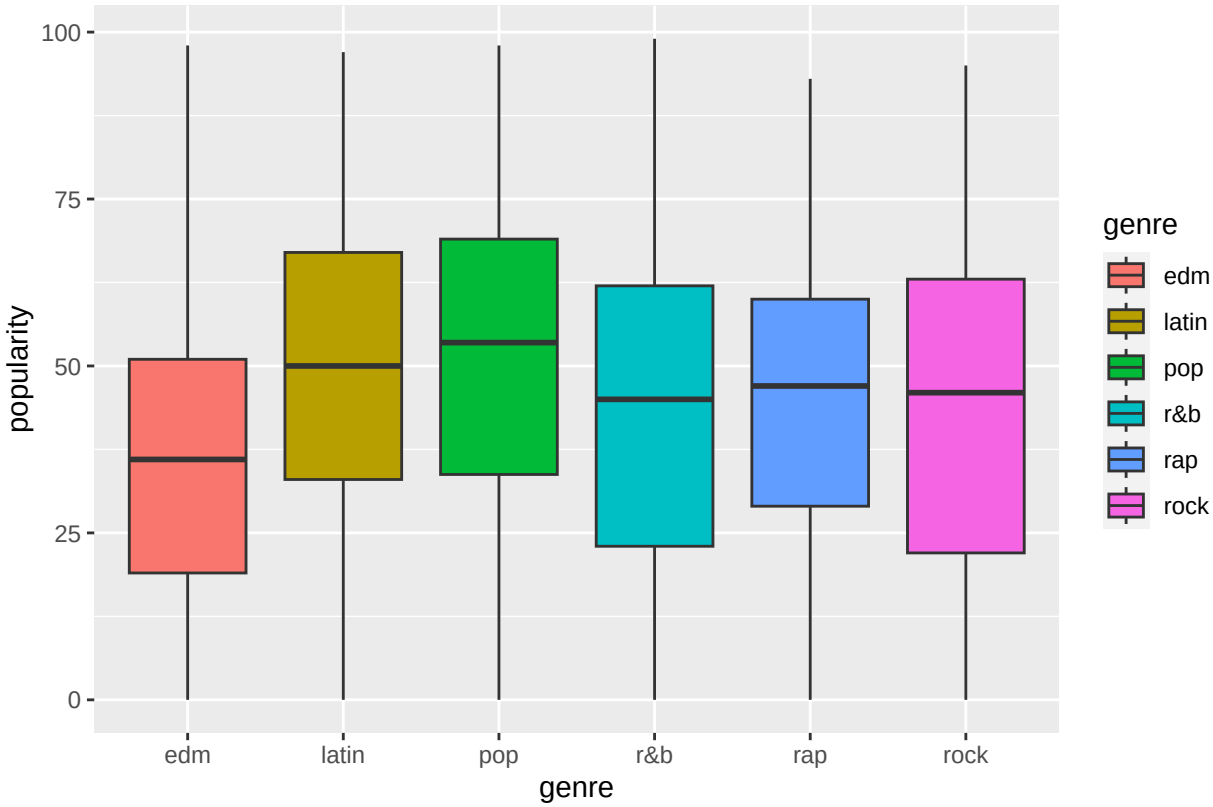

Figure 1: Side−by−side boxplot of popularity against genre

```
# Is there a difference in speechiness for each genre?

ggplot(spotify_final, aes(x = genre, y = speechiness)) +
  geom_boxplot(aes(fill = genre)) +
  labs(caption = "Figure 2: Side-by-side boxplot of speechiness against genre")+
```

```
  theme(plot.caption = element_text(hjust = 0.5))
```

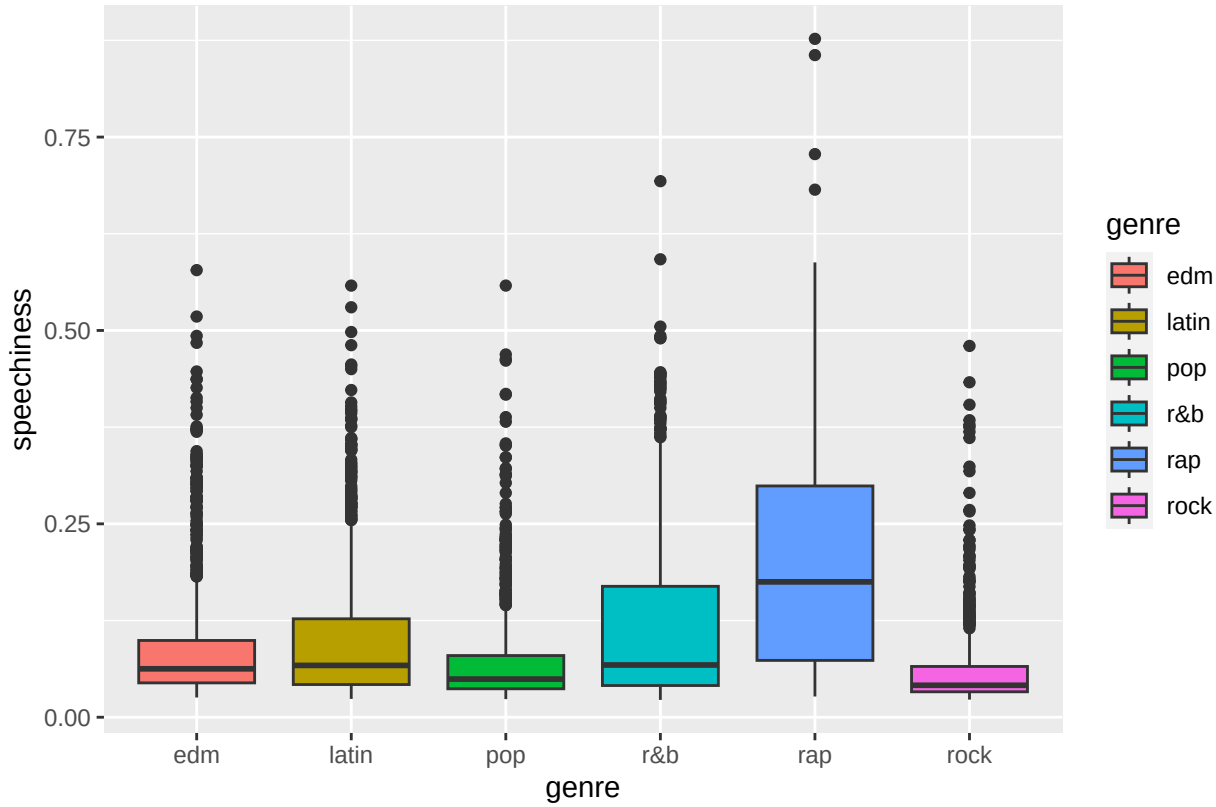

Figure 2: Side−by−side boxplot of speechiness against genre

```
# How does track popularity change over time?

ggplot(spotify_final, aes(x = year, y = popularity)) +
  geom_smooth() +
  labs(caption = "Figure 3: Line graph of popularity against year ")+
  theme(plot.caption = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

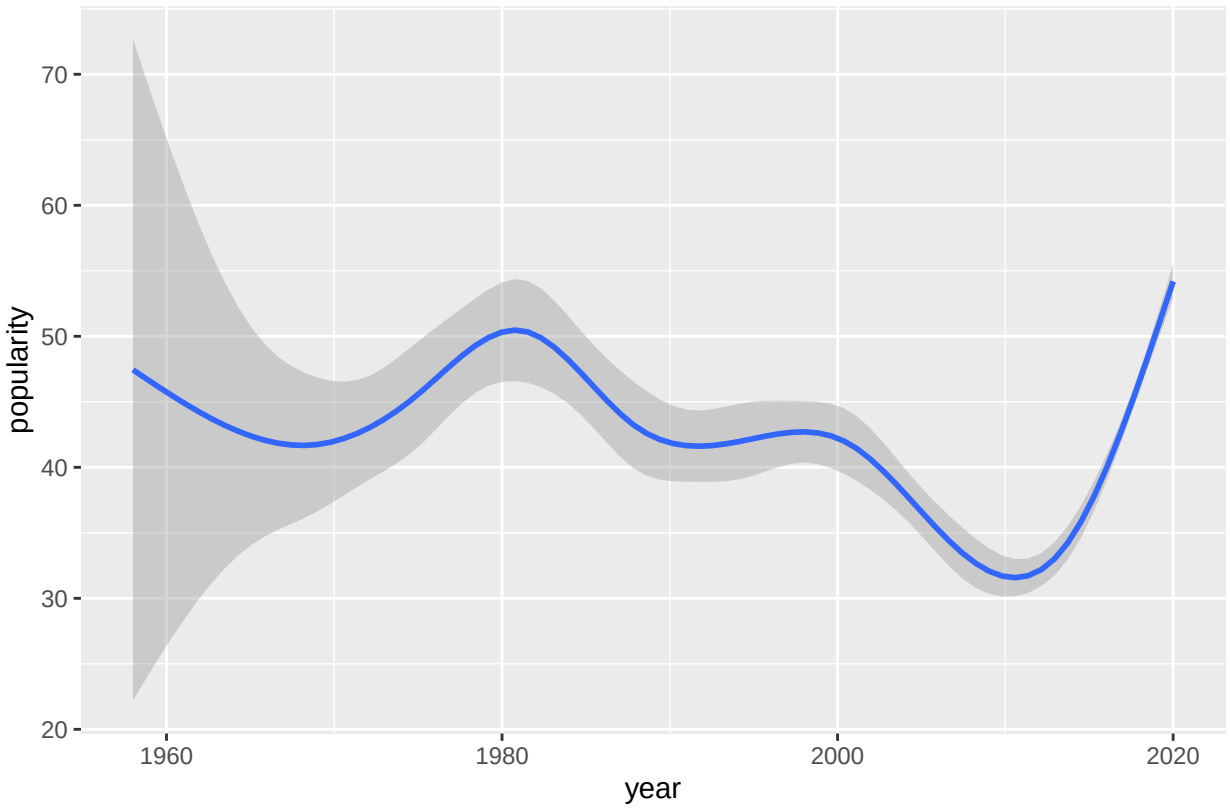

Figure 3: Line graph of popularity against year

```
ggplot(spotify_final, aes(x = year, y = popularity, color = genre)) +
  geom_smooth() +
  labs(caption = "Figure 4: Line graph of popularity against year segmented by each genre")+
  theme(plot.caption = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

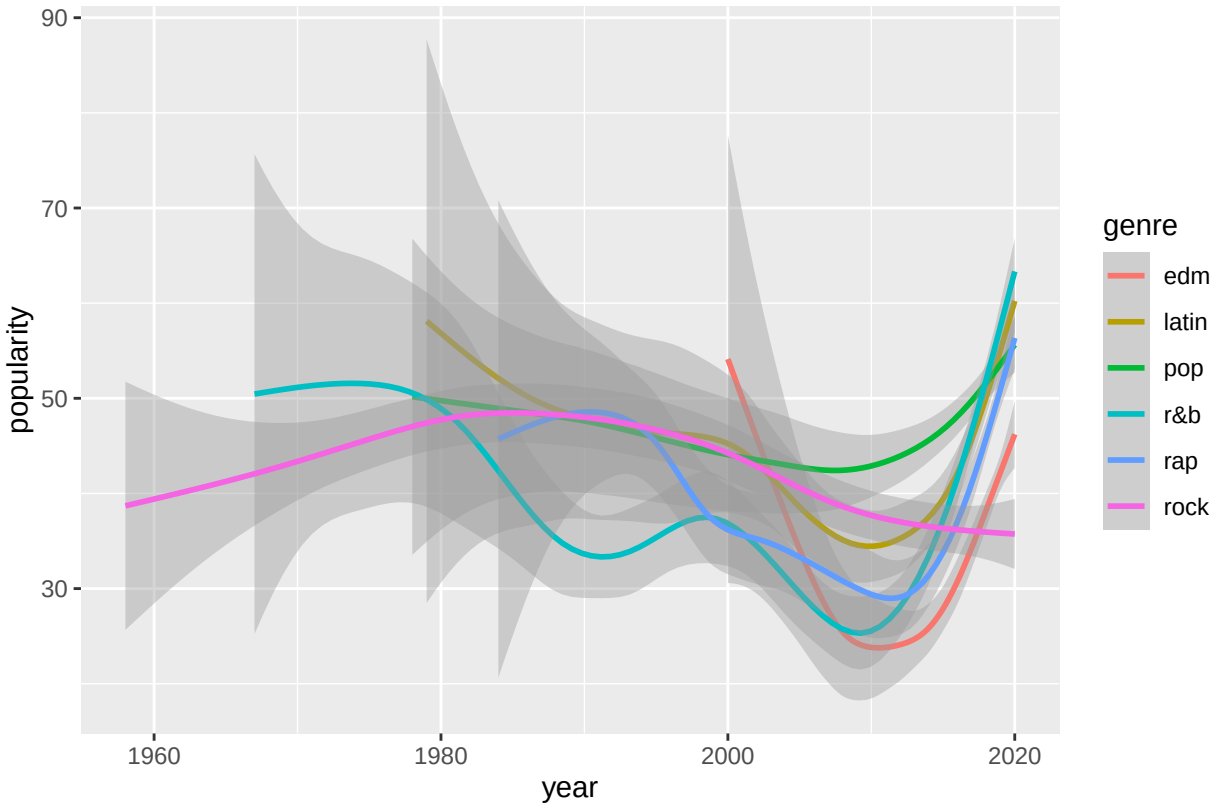

Figure 4: Line graph of popularity against year segmented by each genre

Split

```
set.seed( 1879186 )
spotify_split <- initial_split(spotify_final)
spotify_train <- training( spotify_split )
spotify_test <- testing(spotify_split)

spotify_split
```

```
## <Training/Testing/Total>
## <4500/1500/6000>
```

Preprocessing

```
spotify_recipe <- recipe( genre ~ ., data = spotify_train ) %>%
  step_zv( all_predictors() ) %>%
  step_dummy( all_nominal(), -all_outcomes() ) %>%
  step_normalize( all_predictors() ) %>%
  step_corr( all_predictors() ) %>%
  prep()

spotify_recipe
```

```
##
## -- Recipe ----------------------------------------------------------------------
```

```
##
## -- Inputs
## Number of variables by role
## outcome:    1
## predictor: 14
##
## -- Training information
## Training data contained 4500 data points and no incomplete rows.
##
## -- Operations
## * Zero variance filter removed: <none> | Trained
## * Dummy variables from: mode | Trained
## * Centering and scaling for: year, speechiness, danceability, ... | Trained
## * Correlation filter on: <none> | Trained
```

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

set.seed( 1879186 )
spotify_cv <- vfold_cv( spotify_train_preproc, v = 10 )

skim_without_charts(spotify_train_preproc)
```

Table 4: Data summary

Name	spotify_train_preproc
Number of rows	4500
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
genre	0	1	FALSE	6	lat: 769, rap: 760, pop: 753, edm: 747

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
year	0	1	0	1	-4.67	-0.29	0.41	0.68	0.77
speechiness	0	1	0	1	-0.83	-0.65	-0.44	0.23	7.36

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
danceability	0	1	0	1	-3.80	-0.63	0.11	0.74	2.23
tempo	0	1	0	1	-3.12	-0.79	0.01	0.49	3.20
popularity	0	1	0	1	-1.73	-0.72	0.12	0.80	2.24
energy	0	1	0	1	-3.61	-0.64	0.13	0.79	1.66
key	0	1	0	1	-1.48	-0.92	0.19	0.75	1.59
loudness	0	1	0	1	-9.99	-0.48	0.20	0.69	2.66
acousticness	0	1	0	1	-0.81	-0.73	-0.44	0.39	3.57
instrumentalness	0	1	0	1	-0.38	-0.38	-0.38	-0.35	4.13
liveness	0	1	0	1	-1.18	-0.62	-0.40	0.37	5.30
valence	0	1	0	1	-2.11	-0.78	0.01	0.78	2.00
duration_ms	0	1	0	1	-3.14	-0.64	-0.17	0.47	4.88
mode_X1	0	1	0	1	-1.16	-1.16	0.86	0.86	0.86

Models Spec

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


# knn
knn_spec <- nearest_neighbor(mode = "classification", neighbors = tune()) %>%
  set_engine("kknn")

# LDA
lda_spec <- discrim_linear( mode = "classification" ) %>%
set_engine( "MASS" )
```

Model Fitting

LDA

```
spotify_lda<- discrim_linear(mode = "classification") %>% set_engine("MASS") %>%
  fit(genre ~ year + speechiness + danceability + tempo, data = spotify_train_preproc)

spotify_lda
```

```
## parsnip model object
##
## Call:
## lda(genre ~ year + speechiness + danceability + tempo, data = data)
##
## Prior probabilities of groups:
##       edm     latin       pop       r&b       rap      rock
## 0.1660000 0.1708889 0.1673333 0.1615556 0.1688889 0.1653333
##
## Group means:
##             year speechiness danceability       tempo
```

```
## edm    0.4828518 -0.13414668  -0.02381416  0.20683913
## latin  0.3004931 -0.04391742   0.40338396 -0.11155503
## pop    0.2711581 -0.34851709  -0.07444695 -0.01968431
## r&b   -0.1239233  0.12465422   0.05999182 -0.25496263
## rap    0.2144028  0.87419763   0.48599591  0.00444815
## rock  -1.1677493 -0.48198976  -0.87274931  0.17214585
##
## Coefficients of linear discriminants:
##                       LD1        LD2        LD3        LD4
## year          0.89361431  0.6407804 -0.2805158 -0.3905028
## speechiness   0.35221230 -0.9491113 -0.3505713 -0.3377265
## danceability  0.62796519 -0.1109027  0.4870575  0.8029809
## tempo        -0.04558273  0.1087526 -0.7139135  0.7377566
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.7535 0.2086 0.0300 0.0079
```

KNN

```
knn_grid <- grid_regular(neighbors( range = c(1, 100) ),
                         levels = 20)

knn_grid
```

```
## # A tibble: 20 x 1
##    neighbors
##        <int>
##  1         1
##  2         6
##  3        11
##  4        16
##  5        21
##  6        27
##  7        32
##  8        37
##  9        42
## 10        47
## 11        53
## 12        58
## 13        63
## 14        68
## 15        73
## 16        79
## 17        84
## 18        89
## 19        94
## 20       100
```

```
doParallel::registerDoParallel()
knn_tune <- tune_grid(object = knn_spec,
                      preprocessor = recipe(genre ~ ., data = spotify_train_preproc),
                      resamples = spotify_cv,
                      grid = knn_grid)
```

```
knn_tune %>% collect_metrics()
```

```
## # A tibble: 40 x 7
##    neighbors .metric  .estimator  mean     n std_err .config
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1         1 accuracy multiclass 0.392    10 0.0101  Preprocessor1_Model01
##  2         1 roc_auc  hand_till  0.634    10 0.00584 Preprocessor1_Model01
##  3         6 accuracy multiclass 0.417    10 0.0110  Preprocessor1_Model02
##  4         6 roc_auc  hand_till  0.747    10 0.00478 Preprocessor1_Model02
##  5        11 accuracy multiclass 0.458    10 0.00984 Preprocessor1_Model03
##  6        11 roc_auc  hand_till  0.772    10 0.00494 Preprocessor1_Model03
##  7        16 accuracy multiclass 0.47     10 0.00921 Preprocessor1_Model04
##  8        16 roc_auc  hand_till  0.784    10 0.00483 Preprocessor1_Model04
##  9        21 accuracy multiclass 0.478    10 0.00903 Preprocessor1_Model05
## 10        21 roc_auc  hand_till  0.791    10 0.00487 Preprocessor1_Model05
## # ... with 30 more rows
```

```
best_auc_knn <- select_best(knn_tune, "roc_auc")

final_knn <- finalize_model(knn_spec, best_auc_knn)
final_knn
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 100
##
## Computational engine: kknn
```

```
spotify_knn <- final_knn %>% fit(genre ~ year + speechiness + danceability + tempo, data = spotify_trai

spotify_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = genre ~ year + speechiness + danceability +     tempo, data = data, ks = r
##
## Type of response variable: nominal
## Minimal misclassification: 0.5295556
## Best kernel: optimal
## Best k: 100
```

Random Forest

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

rf_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
```

```
doParallel::registerDoParallel() # This makes macs run a little faster
rf_tuned <- tune_grid( object = rf_spec,
                       preprocessor = recipe(genre ~ . , data = spotify_train_preproc),
                       resamples = spotify_cv,
                       grid = rf_spec_grid )

rf_tuned %>% collect_metrics()
```

```
## # A tibble: 50 x 8
##     mtry min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     2 accuracy multiclass 0.529    10 0.00971 Preprocessor1_Model01
##  2     1     2 roc_auc  hand_till  0.828    10 0.00443 Preprocessor1_Model01
##  3     4     2 accuracy multiclass 0.537    10 0.00804 Preprocessor1_Model02
##  4     4     2 roc_auc  hand_till  0.836    10 0.00349 Preprocessor1_Model02
##  5     7     2 accuracy multiclass 0.546    10 0.00653 Preprocessor1_Model03
##  6     7     2 roc_auc  hand_till  0.835    10 0.00359 Preprocessor1_Model03
##  7    10     2 accuracy multiclass 0.541    10 0.00724 Preprocessor1_Model04
##  8    10     2 roc_auc  hand_till  0.836    10 0.00385 Preprocessor1_Model04
##  9    14     2 accuracy multiclass 0.532    10 0.00705 Preprocessor1_Model05
## 10    14     2 roc_auc  hand_till  0.836    10 0.00377 Preprocessor1_Model05
## # ... with 40 more rows
```

```
best_auc_rf <- select_best( rf_tuned, "roc_auc" )

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

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 4
##   trees = 100
##   min_n = 30
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

```
spotify_rf <- final_rf %>%
  fit( genre ~  year + speechiness + danceability + tempo, data = spotify_train_preproc )

spotify_rf
```

```
## 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:                      4500
## Number of independent variables:  4
## Mtry:                             4
## Target node size:                 30
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.4754715
```

Model Selection

```
# Rf
set.seed(1879186)
rf_cv <- fit_resamples( object = final_rf,
                        preprocessor = recipe(genre ~ . , data = spotify_train_preproc),
                        resamples = spotify_cv )
rf_cv %>%
  collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy multiclass 0.549    10 0.00632 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.846    10 0.00336 Preprocessor1_Model1
```

```
# knn
set.seed(1879186)
knn_cv <- fit_resamples( object = final_knn,
                        preprocessor = recipe(genre ~ ., data = spotify_train_preproc),
                        resamples = spotify_cv )

knn_cv %>%
  collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy multiclass 0.490    10 0.00674 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.810    10 0.00417 Preprocessor1_Model1
```

```
# LDA
set.seed(1879186)
```

```
lda_cv <- fit_resamples(object = lda_spec,
preprocessor = recipe(genre ~ ., data = spotify_train_preproc),
resamples = spotify_cv)

lda_cv %>%
collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy multiclass 0.468    10 0.00500 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.796    10 0.00475 Preprocessor1_Model1
```

```
cv_compare <- lda_cv %>% collect_metrics %>% dplyr::select(.metric, .estimator, mean) %>% rename(LDA_mea
  knn_cv %>% collect_metrics %>% dplyr::select(KNN_mean = mean)
)%>% bind_cols(
  rf_cv %>% collect_metrics %>% dplyr::select(RF_mean = mean)
)
cv_compare %>% knitr::kable(digits = 3, format.args = list(big.mark = ","),
caption = "Table 1: Mean metrics of accuracy and AUC for LDA, KNN and RF models")
```

Table 7: Table 1: Mean metrics of accuracy and AUC for LDA, KNN and RF models

.metric	.estimator	LDA_mean	KNN_mean	RF_mean
accuracy	multiclass	0.468	0.49	0.549
roc_auc	hand_till	0.796	0.81	0.846

Model Evaluation

```
set.seed(1879186)
spotify_rf_4 <- final_rf %>%
  fit( genre ~ year + speechiness + danceability + tempo , data = spotify_train_preproc )

spotify_rf_4
```

```
## parsnip model object
##
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~4L,      x), num.trees = ~100, min.r
##
## Type:                             Probability estimation
## Number of trees:                  100
## Sample size:                      4500
## Number of independent variables:  4
## Mtry:                             4
## Target node size:                 30
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.4745151
```

```
spotify_imp_4 <- spotify_rf_4 %>% vip()
spotify_imp_4 + labs(caption = "Figure 5: Level of importance of year, tempo, danceability and speechine
```

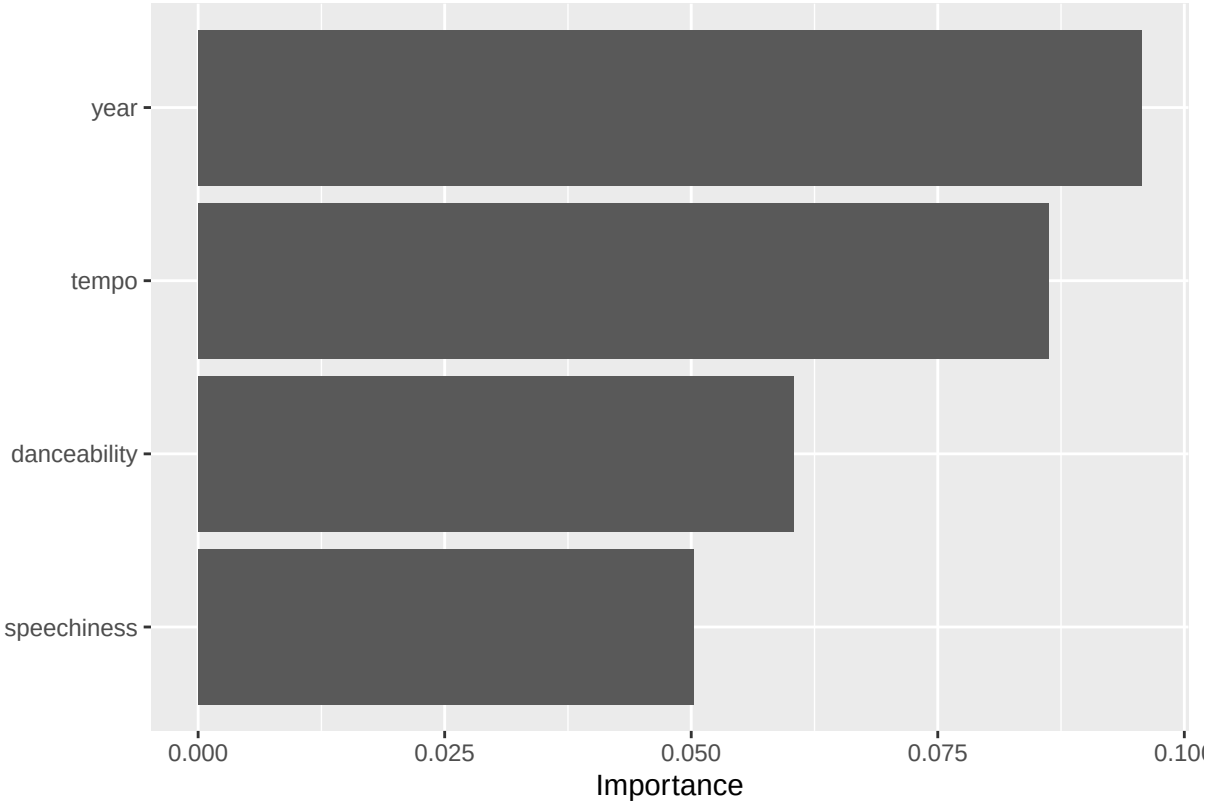

Figure 5: Level of importance of year, tempo, danceability and speechiness variable in predicting genre

```
set.seed(1879186)
spotify_rf_all <- final_rf %>%
  fit( genre ~ . , data = spotify_train_preproc )

spotify_rf_all
```

```
## 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:                      4500
## Number of independent variables:  14
## Mtry:                             4
## Target node size:                 30
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.4518611
```

```
spotify_rf_all %>% vip() + labs(caption = "Figure 6: Level of importance of all variable in predicting g
```

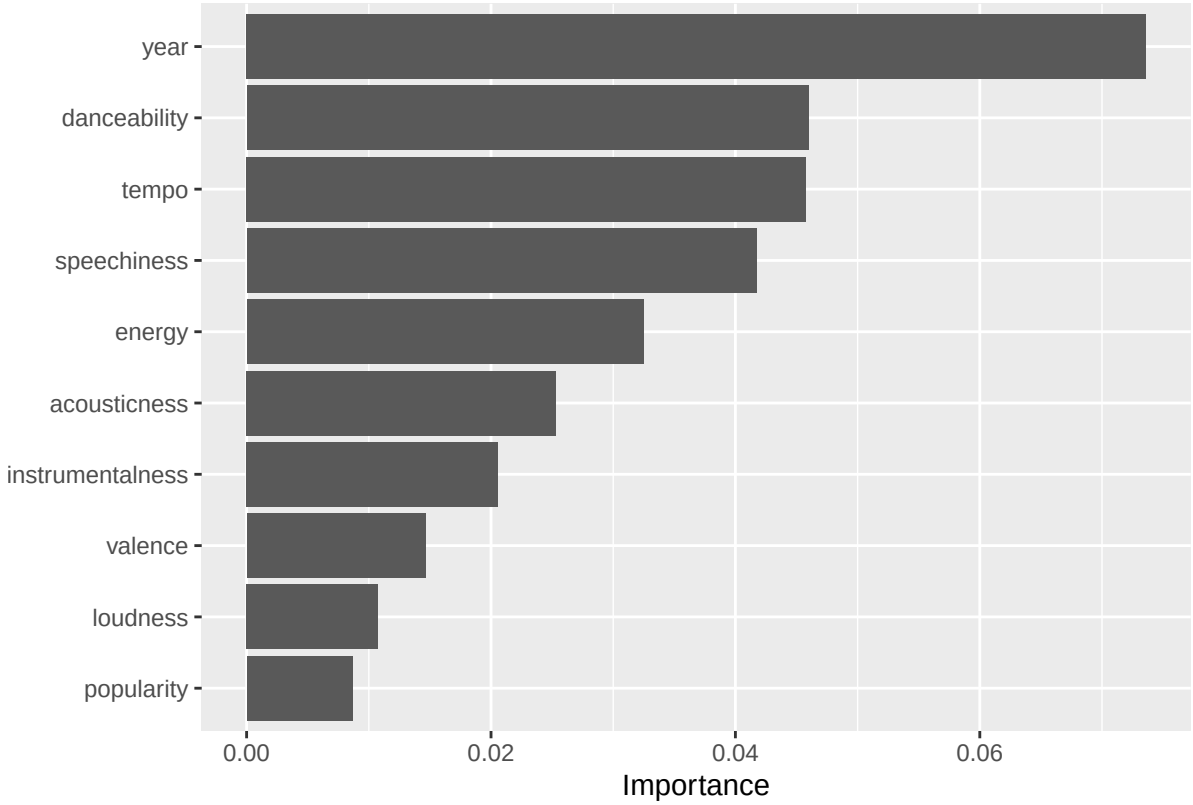

Figure 6: Level of importance of all variable in predicting genre

Model Prediction

only fitting year, danceability, tempo and speechiness

```
spotify_preds <- predict( spotify_rf_4, # The predictions
                          new_data = spotify_test_preproc ) %>%
  bind_cols( spotify_test_preproc %>%  # Add the truth
               dplyr::select( genre ) )%>% bind_cols(
  predict(spotify_rf_4, spotify_test_preproc, type = "prob")
)
spotify_preds
```

```
## # A tibble: 1,500 x 8
##    .pred_class genre .pred_edm .pred_latin .pred_pop `.pred_r&b` .pred~1 .pred~2
##    <fct>       <fct>     <dbl>       <dbl>     <dbl>       <dbl>   <dbl>   <dbl>
##  1 pop         edm      0.0585      0.264     0.427       0.127  0.102   0.0215
##  2 r&b         edm      0.0732      0.0582    0.172       0.281  0.222   0.193
##  3 edm         edm      0.517       0.188     0.0950      0.0562 0.143   0.00174
##  4 latin       edm      0.266       0.370     0.170       0.0485 0.0303  0.116
##  5 pop         edm      0.207       0.123     0.549       0.0446 0.0205  0.0549
##  6 edm         edm      0.405       0.0717    0.229       0.230  0.00805 0.0569
##  7 edm         edm      0.681       0.0609    0.0493      0.0597 0.0774  0.0717
##  8 edm         edm      0.550       0.105     0.130       0.0895 0.124   0.00151
##  9 latin       edm      0.0718      0.387     0.234       0.155  0.121   0.0313
## 10 edm         edm      0.358       0.173     0.168       0.0953 0.179   0.0271
```

```
## # ... with 1,490 more rows, and abbreviated variable names 1: .pred_rap,
## #   2: .pred_rock
```

```
spotify_preds %>%
  metrics( truth = genre, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.468
## 2 kap      multiclass     0.363
```

```
spotify_cm <- spotify_preds %>% conf_mat(truth = genre, estimate = .pred_class)

spotify_cm
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm   144    23  52  11  15   18
##      latin  23   104  40  52  36   10
##      pop    50    33  81  40  18   39
##      r&b     5    22  15  72  24   23
##      rap    24    34  29  63 139    4
##      rock    7    15  30  35   8  162
```

```
spotify_preds %>%
sensitivity( truth = genre , estimate = .pred_class ) %>% bind_rows(
  spotify_preds %>% specificity(truth = genre, estimate = .pred_class)
) %>% knitr::kable(digits = 3, format.args = list(big.mark = ","),
caption = "Table 2: sensitivity and specificity of the RF model using the variable year, tempo, danceabi
```

Table 8: Table 2: sensitivity and specificity of the RF model using the variable year, tempo, danceability and speechiness

.metric	.estimator	.estimate
sensitivity	macro	0.471
specificity	macro	0.894

fitting all variables

```
spotify_preds_t <- predict( spotify_rf_all, # The predictions
                        new_data = spotify_test_preproc ) %>%
  bind_cols( spotify_test_preproc %>%  # Add the truth
               dplyr::select( genre ) )

spotify_preds_t %>%
  metrics( truth = genre, estimate = .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.548
## 2 kap      multiclass     0.458
```

```
spotify_cm_t <- spotify_preds_t %>% conf_mat(truth = genre, estimate = .pred_class)


spotify_cm_t
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm   159    19  41   8  14   12
##      latin  17   103  37  28  27    8
##      pop    52    40  97  32  22   22
##      r&b     9    23  31 118  22   17
##      rap    12    37  17  66 150    2
##      rock    4     9  24  21   5  195
```

```
spotify_preds_t %>%
sensitivity( truth = genre , estimate = .pred_class ) %>% bind_rows(
  spotify_preds %>% specificity(truth = genre, estimate = .pred_class)
) %>% knitr::kable(digits = 3, format.args = list(big.mark = ","),
caption = "Table 3: sensitivity and specificity of the RF model using all variable")
```

Table 9: Table 3: sensitivity and specificity of the RF model using all variable

.metric	.estimator	.estimate
sensitivity	macro	0.548
specificity	macro	0.894