Spotify - Predicting music genre

Satyaveer Pattanaik

2023-02-20

Executive Summary

Spotify is an audio streaming and media services provider with over 365 million monthly active users, including 165 million paying subscribers. As Spotify is the world's largest music streaming service provider, the founders are interested in trying to predict which genre a song belongs to in order to better enhance their customer's experience and ultimately remain the best music streaming service available. If we can predict what genre a song belongs to, we can better recommend / advertise songs to customers and more effectively update compilation playlists.

We have been provided an extensive dataset collated by Spotify's web services division.

The founders at Spotify are interested in the following questions:

- Does the popularity of songs differ between genres?
- Is there a difference in *speechiness* for each genre?
- How does track popularity change over time?

For this report, we have been tasked with building a model to predict genre based on appropriate variables in the provided dataset. After consultation with an expert statistician, the founders have decided they would like us to compare the following three models:

- A linear discriminant analysis
- A K-nearest neighbours model
- A random forest

After tuning and evaluating models, it was determined that the random forest model was the most accurate while predicting song genre. The resulting model had an accuracy of **55.1%** on unseen data.

Methods

We have been provided an extensive dataset collated by Spotify's web services division. This is a massive dataset containing a whole bunch of information about different songs from different playlists on Spotify. It was collected using the spotifyr package in R, which lets you specify and scrape data from Spotify so that you can analyse it. The dataset is made available at https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-01-21/spotify_songs.csv

Due to limits on computing power, we reduced the dataset size by sampling 1000 songs per genre.

For my analysis we used the following variables as predictors `danceability`, `energy` , `key`, `loudness` , `mode` , `speechiness`, `acousticness` , `instrumentalness` , `liveness` , `valence` , `tempo` , `duration_ms`, `track_popularity` and `release_year`.

Our task was to to predict `playlist_genre`.

We extracted the year (YYYY format) from `track_album_release_date` in the original dataset and changed the variable to be called `release_year`.

We dropped the following variables:

1. `track_id`
2. `track_name`
3. `track_artist`
4. `track_album_id`
5. `track_album_name`
6. `playlist_name`
7. `playlist_ID`
8. `playlist_subgenre`

Variables 1-7 were dropped since they were identifying variables, hence they would not add anything significant to the model. The variable `playlist_subgenre` was dropped as it contains lot of information on the `playlist_genre` (which we are predicting) and hence would dominate over the other variables.

The following analysis which was completed in R version 4.2.0 using the tidyverse, tidymodels, lubridate, discrim and doParallel packages.

Results

After cleaning the dataset, we conducted a thorough exploratory data analysis. The observations are as follows:

- The `pop` genre had the highest median popularity (Popularity is a number from 0 to 100 indicating popularity. The higher the value, the more popular the song). The `edm` genre had the lowest median popularity. As shown in Figure 1 in the appendix, the ascending order of popularity of genres is as follows:

    1. `edm`
    2. `r&b`
    3. `rock`
    4. `rap`
    5. `latin`
    6. `pop`

- As expected, `rap` songs had the highest median *speechiness*, and `rock` songs had the lowest median *speechiness* (Speechiness is a value from 0 to 1 describing how "speechy" the track is. Higher values indicate spoken-word tracks). As shown in Figure 2 in the appendix, the ascending order of speechiness in the genres is as follows:

    1. `rock`
    2. `pop`

3. `edm`
4. `latin`
5. `r&b`
6. `rap`

- The popularity of songs rose from the 1960 until the year 1980. After 1980, the popularity dropped gradually hitting an all-time low in the year 2009. From 2010 onwards popularity picked till 2020 (the most recent year in the dataset) .

The Tidymodels package allows us to create a "recipe" for pre-processing our data before modeling. This eliminates redundancy when making multiple models. We used the recipe to specify the prediction formula, remove highly correlated variables, and then center and scale the numeric attributes so that they are on level playing fields when assessing their predictive power.

We tuned our models as per specifications given to us. We then tested our tuned models on cross-validated data (obtained from the preprocessed training data) to compare them.

For a multi-classification problem like this one that has fairly balanced classes, accuracy is the better metric than AUC (accuracy is a pure measure of true positives and true negatives; AUC is better for weighting false positives and negatives).

The random forest model performed at about **55.3%** accuracy on the preprocessed training data compared to about **49.5%** for the knn model and **46%** from linear discriminant analysis. Hence, the random forest model was selected as the best model for predicting genres and we fitted the model with the preprocessed training data for further evaluation.

If we guessed randomly which genre to assign to each song in this dataset, the accuracy would be **16.6%** (or 1 in 6). The random forest model (55%) improved it more than threefold.

On further evaluation, the random forest model had an accuracy of **55.1%** on unseen data (our preprocessed test data). This means we are correctly predicting that the song belongs to its genre approximately 54% of the time.

Discussion

As shown in Figure 6, the most important variable in our model was determined to be `release_year` (the year the song was released). The next three variables in decreasing order of importance are as follows:

1. `speechiness` : how *speechy* the song is
2. `danceability` : how danceable the song is
3. `tempo` : the tempo of the song

On the testing data, the overall sensitivity of the model is **55.1%** (notice it is the same as the accuracy metric). This means we are predicting that the song belongs to its correct genre approximately 54% of the time.

The overall specificity of the model is **91%**. This means that we are correctly predicting that a song does not belong to a particular genre approximately 91% of the time.

On comparing the sensitivity metrics with respect to each genre, we observe that `rock` songs are the easiest to classify followed by, in decreasing order of sensitivity:

1. `rap`

2. `edm`
3. `r&b`
4. `latin`
5. `pop`

Conclusion

As Spotify is the world's largest music streaming service provider, the company is interested in trying to predict which genre a song belongs to in order to better enhance their customer's experience and ultimately remain the best music streaming service available. If we can predict what genre a song belongs to, Spotify can better recommend / advertise songs to customers and more effectively update its compilation playlists.

On conducting a thorough exploratory data analysis, these were the key observations:

- `pop` songs had the highest median popularity while `edm` songs had the lowest median popularity.
- `rap` songs had the highest median *speechiness*, while `rock` songs had the lowest median *speechiness*
- The popularity of songs rose from the 1960 until the year 1980. After 1980, the popularity dropped gradually hitting an all-time low in the year 2009. From 2010 onwards popularity picked till 2020 (the most recent year in the dataset) .

For this report, we were tasked with building a model to predict song genre based on appropriate variables in the provided dataset. We compared the following three models (tuned as per requirement):

- A linear discriminant analysis
- A K-nearest neighbours model
- A random forest

After tuning and evaluating the three models, it was determined that a **random forest model** with

- **100** trees,
- `mtry` (no. of predictors to consider at each split) = **4**, and
- `min_n` (min. no. of nodes) = **30**

was the most accurate to predict song genre. The resulting model had an accuracy of **55.1%** on unseen data. Classifying fewer genres would likely improve this metric, and trying to classify more than 6 would likely drive it down further. Incorporating more variables such as artist name and playlist name may improve model performance, but we have left these variables out due to computational limits.

The most important variable in our model was determined to be `release_year` followed by `speechiness`, `danceability` and `tempo`.

The genres `pop`, `latin`, and `r&b` were the most difficult to classify while `edm`, `rap` and `rock` were easier to classify.

Appendix

```r
#loading data and cleaning
pacman::p_load(tidyverse, tidymodels, lubridate, discrim, doParallel)
spotify_songs <- readr::read_csv('spotify_songs.csv')
```

```
## 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.
```

```r
extract_regex <- "(\\d+)"
spotify_songs <- spotify_songs %>%
  mutate(release_year = str_extract(track_album_release_date, extract_regex))

spotify_songs <- spotify_songs %>%
  select(-track_album_release_date)%>%
  mutate(release_year = year(as.Date(release_year, format = "%Y")))

head(spotify_songs)
```

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

```r
spotify_songs %>%
  count(playlist_genre)
```

```
## # A tibble: 6 x 2
##   playlist_genre     n
##   <chr>          <int>
## 1 edm             6043
## 2 latin           5155
## 3 pop             5507
## 4 r&b             5431
## 5 rap             5746
## 6 rock            4951
```

```r
#sampling 1000 songs for each genre

set.seed(1829981)
spotify_songs <- spotify_songs %>%
  group_by(playlist_genre) %>%
  slice_sample(n = 1000) %>%
  ungroup()

#shuffling the rows in the dataset
set.seed(1829981)
rows <- sample(nrow(spotify_songs))
spotify_songs <- spotify_songs[rows,]

#checking if the sampling worked
spotify_songs %>%
  count(playlist_genre)
```

```
## # A tibble: 6 x 2
##   playlist_genre     n
##   <chr>          <int>
## 1 edm             1000
## 2 latin           1000
## 3 pop             1000
## 4 r&b             1000
## 5 rap             1000
## 6 rock            1000
```

```r
skimr::skim(spotify_songs)
```

Table 1: Data summary

Name	spotify_songs
Number of rows	6000
Number of columns	23
Column type frequency:	
character	9
numeric	14
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	5804	0
track_name	0	1	1	116	0	5457	0
track_artist	0	1	2	37	0	3522	0
track_album_id	0	1	22	22	0	5358	0
track_album_name	0	1	1	151	0	5096	0
playlist_name	0	1	6	120	0	447	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
playlist_id	0	1	22	22	0	468	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	42.44	25.06	0.00	23.00	46.00	62.00	99.00	
danceability	0	1	0.66	0.15	0.12	0.56	0.67	0.76	0.98	
energy	0	1	0.70	0.18	0.01	0.58	0.72	0.84	1.00	
key	0	1	5.33	3.63	0.00	2.00	6.00	9.00	11.00	
loudness	0	1	-6.78	2.97	-28.31	-8.24	-6.20	-4.72	-0.48	
mode	0	1	0.57	0.49	0.00	0.00	1.00	1.00	1.00	
speechiness	0	1	0.11	0.10	0.02	0.04	0.06	0.13	0.86	
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	0.01	0.97	
liveness	0	1	0.19	0.15	0.01	0.09	0.13	0.25	0.99	
valence	0	1	0.52	0.23	0.01	0.33	0.52	0.71	0.98	
tempo	0	1	121.01	26.97	37.11	100.00	121.03	134.01	212.06	
duration_ms	0	1	224589.68	59155.56	42105.00	187796.75	215296.00	252681.00	516760.00	
release_year	0	1	2010.94	11.75	1960.00	2008.00	2016.00	2019.00	2020.00	

```r
#dropping variables
spotify_songs <- spotify_songs %>%
  select(c(-'track_id', -'track_album_id', -'playlist_id', -'track_name',
           -'track_album_name', -'track_artist', -'playlist_name',
           -'playlist_subgenre')) %>%
  mutate(playlist_genre = as.factor(playlist_genre))
```

```r
#comparing popularity amongst genres
spotify_songs %>%
  ggplot(aes(reorder(playlist_genre, track_popularity),
             track_popularity, fill = playlist_genre))+
  geom_boxplot(show.legend = F) +
  labs(x = "Genre",
       y = "Popularity",
       title = "Popularity of songs across Genres") +
  theme_bw() +
  labs(caption = "Figure 1: Side-by-side boxplots comparing popularity of songs across Genres") +
  theme(plot.caption = element_text(hjust = 0.5))
```

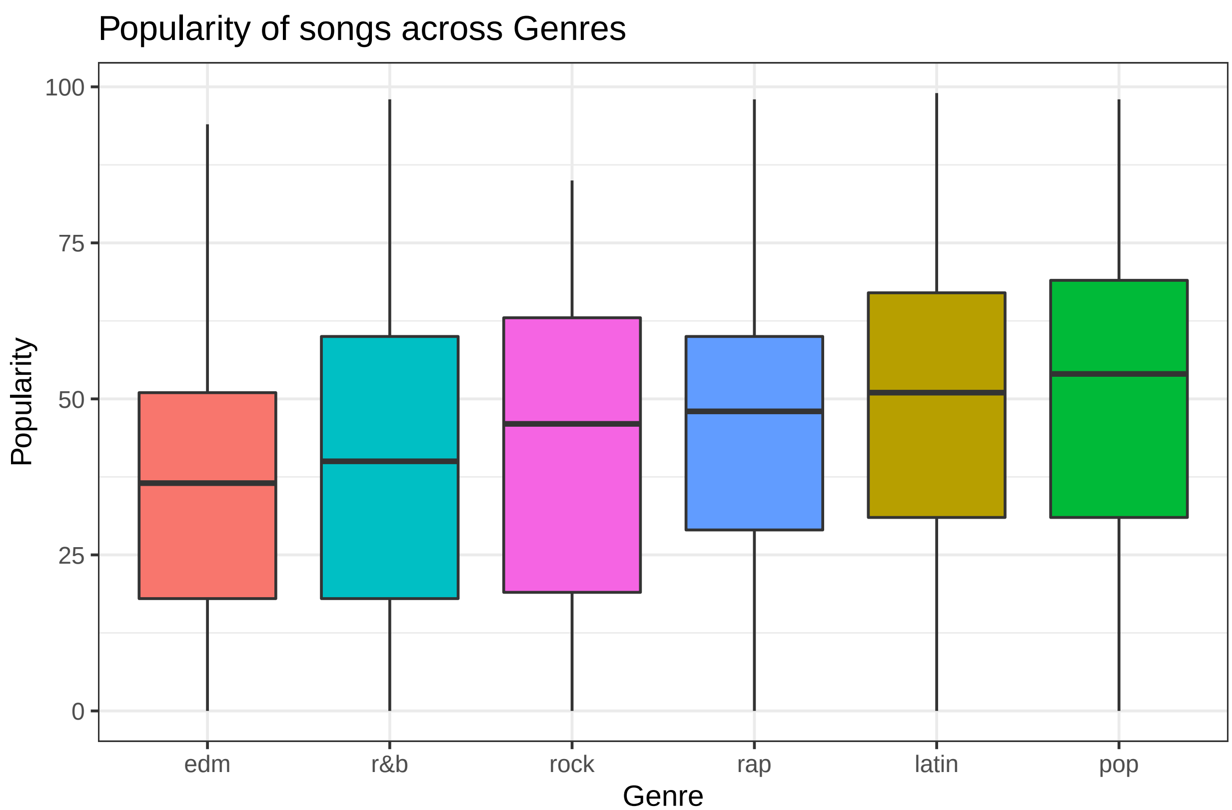

Figure 1: Side−by−side boxplots comparing popularity of songs across Genres

```
#comparing speechiness amongst genres
spotify_songs %>%
  ggplot(aes(reorder(playlist_genre, speechiness),
             y = speechiness, fill = playlist_genre)) +
  geom_boxplot(show.legend = F) +
  labs(x = "Genre",
       y = "Speechiness",
       title = "'Speechiness' of songs across Genres") +
  theme_bw() +
  labs(caption = "Figure 2: Side-by-side boxplots comparing 'speechiness' of songs across Genres") +
  theme(plot.caption = element_text(hjust = 0.5))
```

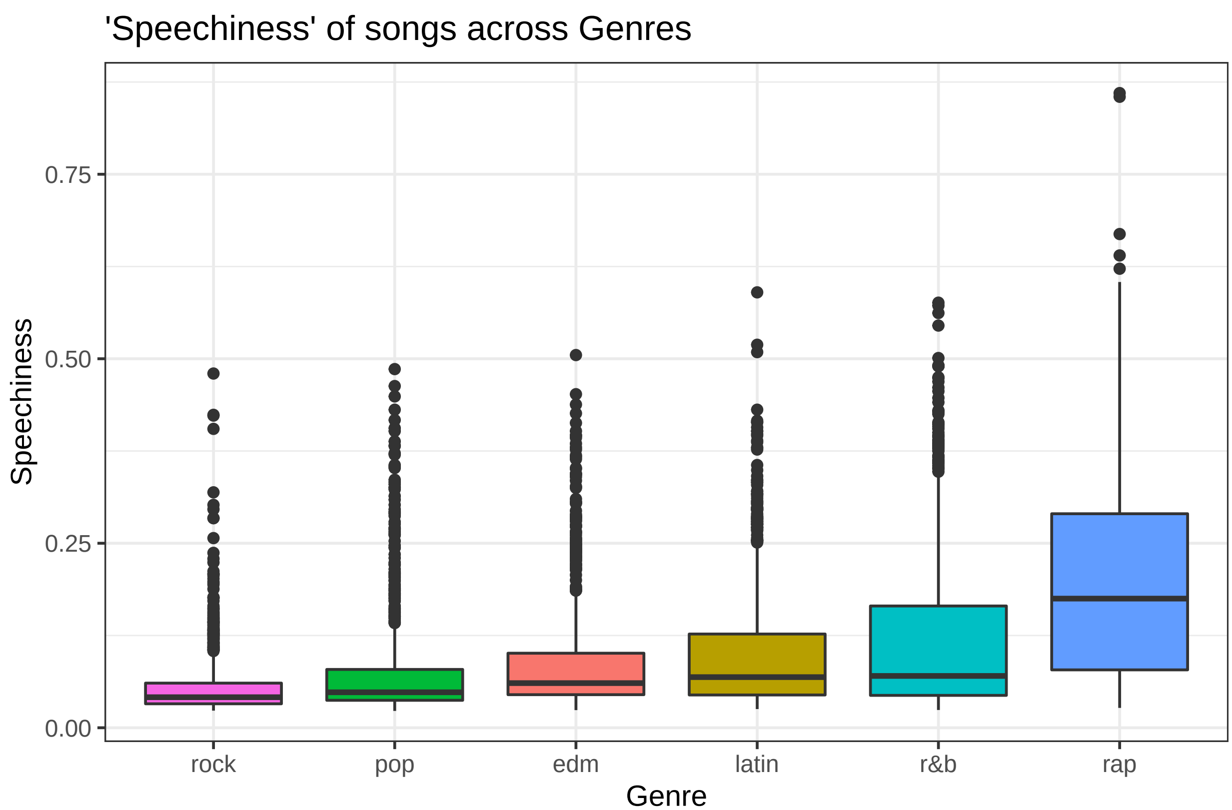

Figure 2: Side−by−side boxplots comparing 'speechiness' of songs across Genres

```
#tracking song popularity over time
spotify_songs %>%
  group_by(release_year) %>%
  summarise(mean_popularity = mean(track_popularity)) %>%
  ggplot(aes(x = release_year, y = mean_popularity)) +
  geom_point() +
  geom_line() +
  geom_smooth(se = F) +
  scale_x_continuous(breaks = seq(1960, 2020, 5)) +
  labs(x = "Year",
       y = "Mean Popularity",
       title = "Popularity of songs over time") +
  theme_bw() +
  labs(caption = "Figure 3: A line graph charting the mean popularity of songs (for each year) over tim
  theme(plot.caption = element_text(hjust = 0.5))
```

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

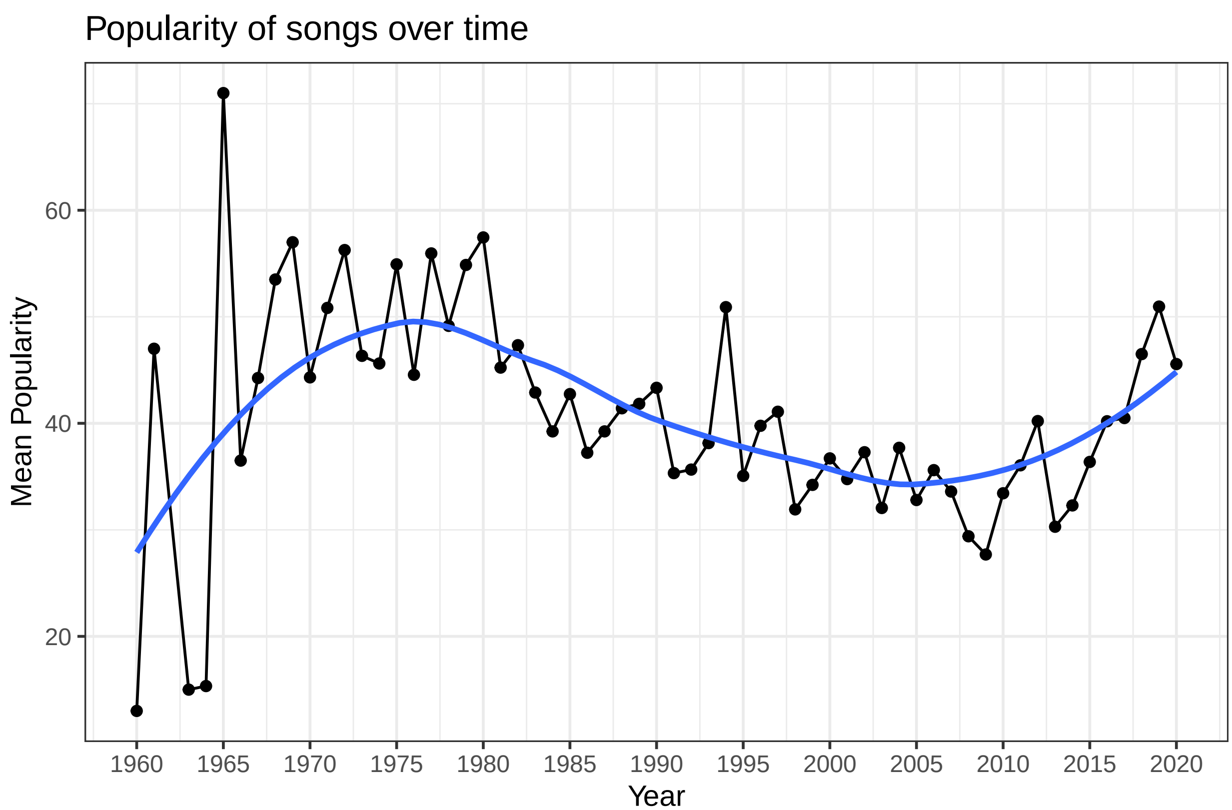

Figure 3: A line graph charting the mean popularity of songs (for each year) over time

```
#data splitting
set.seed(1829981)
spotify_split <- initial_split(spotify_songs, strata = playlist_genre)
spotify_train <- training(spotify_split)
spotify_test <- testing(spotify_split)

#data preprocessing
spotify_recipe <- recipe(playlist_genre ~ ., data = spotify_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 4500 data points and no missing data.
##
```

```
## Operations:
##
## Zero variance filter removed <none> [trained]
## Centering and scaling for track_popularity, danceability, energy, key, lo... [trained]
## Correlation filter on <none> [trained]
```

```
spotify_train_preproc <- juice(spotify_recipe)
```

```
#model specifications

#Linear Discriminant Analysis
lda_spec <- discrim_linear(mode = "classification") %>%
  set_engine("MASS")

#K-nearest neighbours model
knn_spec <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("classification")

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

```
#model tuning
#creating 5 bootstrap samples
set.seed(1829981)
spotify_boots <- bootstraps(spotify_train_preproc, times = 10,
                            strata = playlist_genre)

#random forest
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
```

```
cores <- parallel::detectCores(logical = F)
cl <- makePSOCKcluster(cores)
registerDoParallel(cl)
rf_grid <- tune_grid( object = rf_spec,
                      preprocessor = recipe(playlist_genre ~ . , data = spotify_train_preproc),
                      resamples = spotify_boots,
                      grid = rand_spec_grid )

#plotting the metrics
rf_grid %>%
  collect_metrics() %>%
  mutate( min_n = as.factor( min_n ) ) %>%
  ggplot( aes( x = mtry, y = mean, colour = min_n ) ) +
  geom_point( size = 2 ) +
  geom_line( alpha = 0.75 ) +
  facet_wrap( ~ .metric, scales = "free", nrow = 3 ) +
  theme_bw()  +
  labs(caption = "Figure 4: A line graph charting the mean accuracy and roc_auc for each level of tuning
  theme(plot.caption = element_text(hjust = 0.5))
```

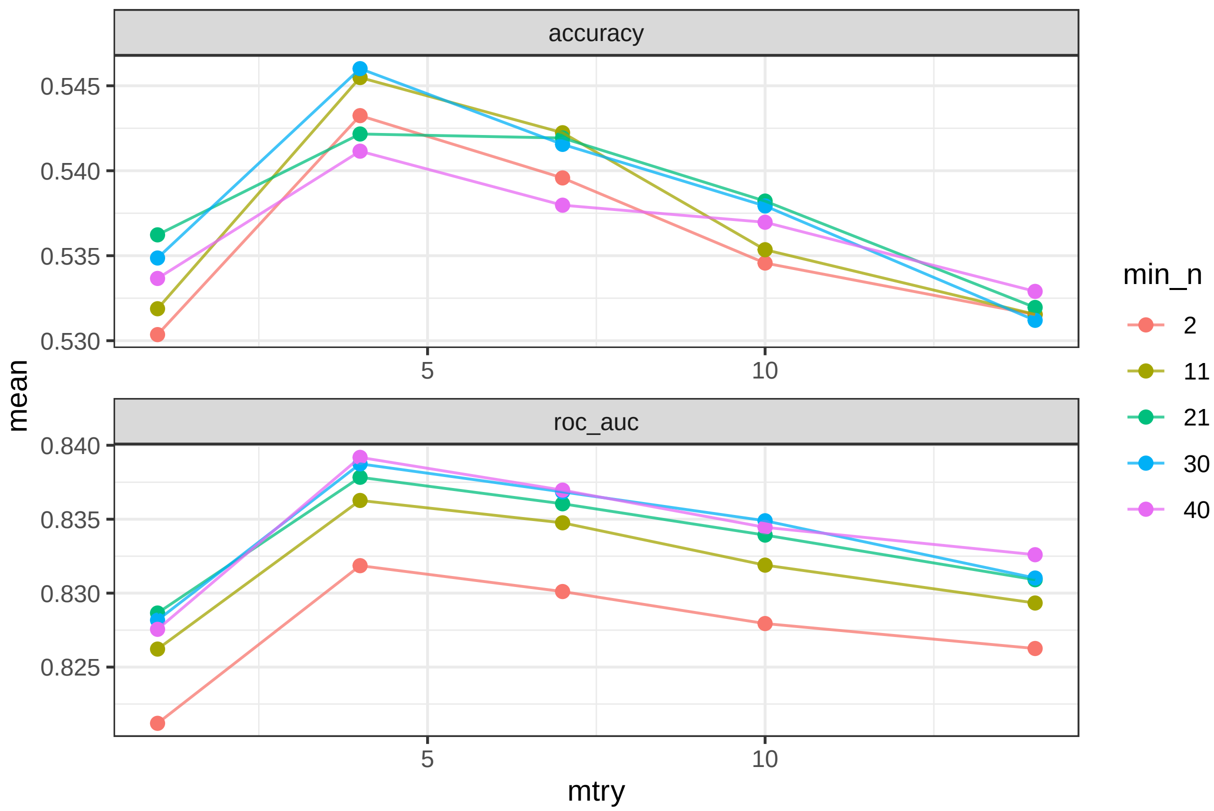

Figure 4: A line graph charting the mean accuracy and roc_auc for each level of tuning

```r
best_rf_acc <- select_best( rf_grid, "accuracy" )
best_rf_acc
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     4    30 Preprocessor1_Model17
```

```r
final_rf <- finalize_model( rf_spec, best_rf_acc )
final_rf
```

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

```r
#tuning the knn model
params_grid <- grid_regular(neighbors(range(1, 100)), levels = 20)


cores <- parallel::detectCores(logical = F)
cl <- makePSOCKcluster(cores)
registerDoParallel(cl)
knn_tuned <- tune_grid(object = knn_spec,
                       preprocessor = recipe(playlist_genre ~ . , data = spotify_train_preproc),
                       resamples = spotify_boots,
                       grid = params_grid)

#plotting the metrics
knn_tuned %>%
  collect_metrics() %>%
  ggplot( aes( x = neighbors, y = mean)) +
  geom_point( size = 2 ) +
  geom_line( alpha = 0.75 ) +
  facet_wrap( ~ .metric, scales = "free", nrow = 3 ) +
  theme_bw() +
  labs(caption = "Figure 5: A line graph charting the mean accuracy and roc_auc for each value of neighb
  theme(plot.caption = element_text(hjust = 0.5))
```

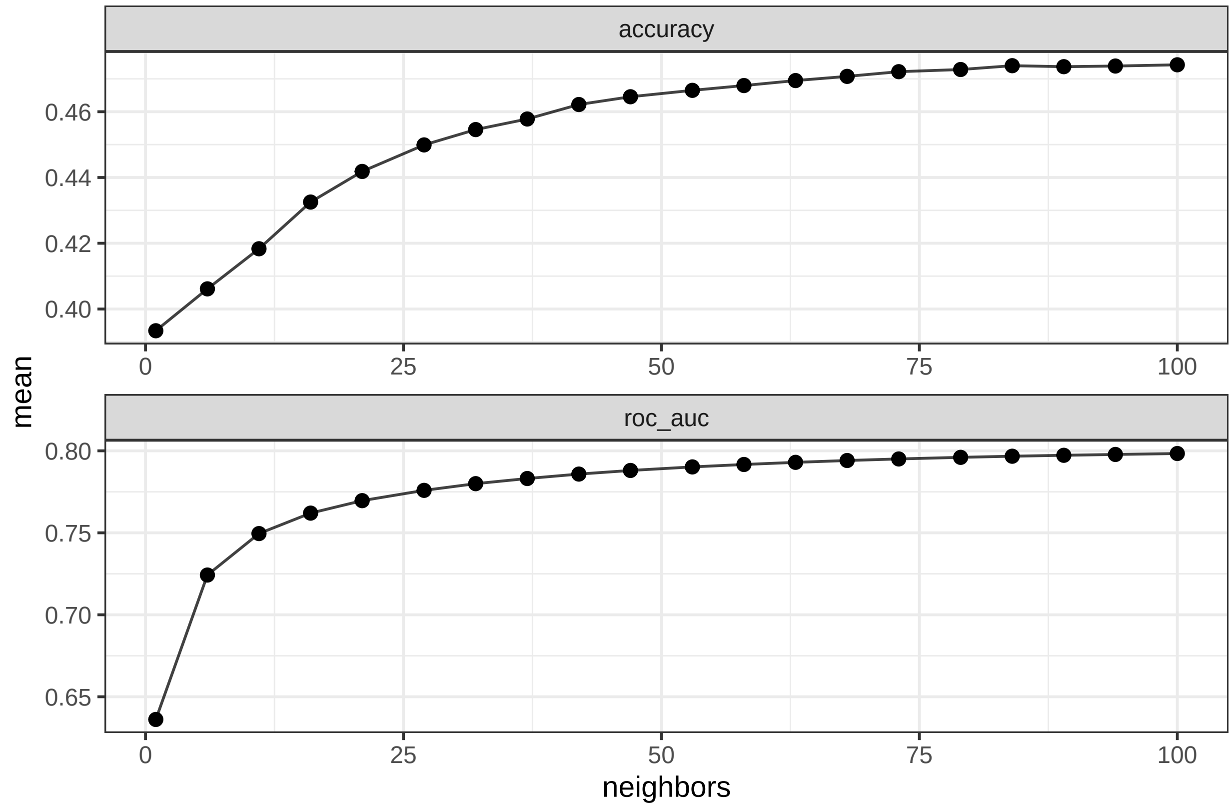

Figure 5: A line graph charting the mean accuracy and roc_auc for each value of neighbours

```
best_acc <- select_best(knn_tuned, metric = "accuracy")
best_acc %>%
  select(neighbors)
```

```
## # A tibble: 1 x 1
##   neighbors
##       <int>
## 1       100
```

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

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

```
#model selection

#10 fold cross-validation to determine best model
set.seed(1829981)
spotify_cv <- vfold_cv(spotify_train_preproc, v = 10, strata = playlist_genre)
```

```r
#lda
lda_cv <- fit_resamples( object = lda_spec,
                         preprocessor = recipe(playlist_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 0.46     10 0.00776 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.795    10 0.00511 Preprocessor1_Model1
```

```r
#knn
knn_cv <- fit_resamples( object = final_knn,
                        preprocessor = recipe(playlist_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 0.495    10 0.00595 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.811    10 0.00434 Preprocessor1_Model1
```

```r
#random forest
rf_cv <- fit_resamples( object = final_rf,
                        preprocessor = recipe(playlist_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 0.550    10 0.00579 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.846    10 0.00350 Preprocessor1_Model1
```

Random forest has the highest accuracy + roc_auc.

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

spotify_rf %>%
  vip::vip() +
  labs(caption = "Figure 6: Variable importance plot for our random forest model") +
  theme_bw() +
  theme(plot.caption = element_text(hjust = 0.5))
```

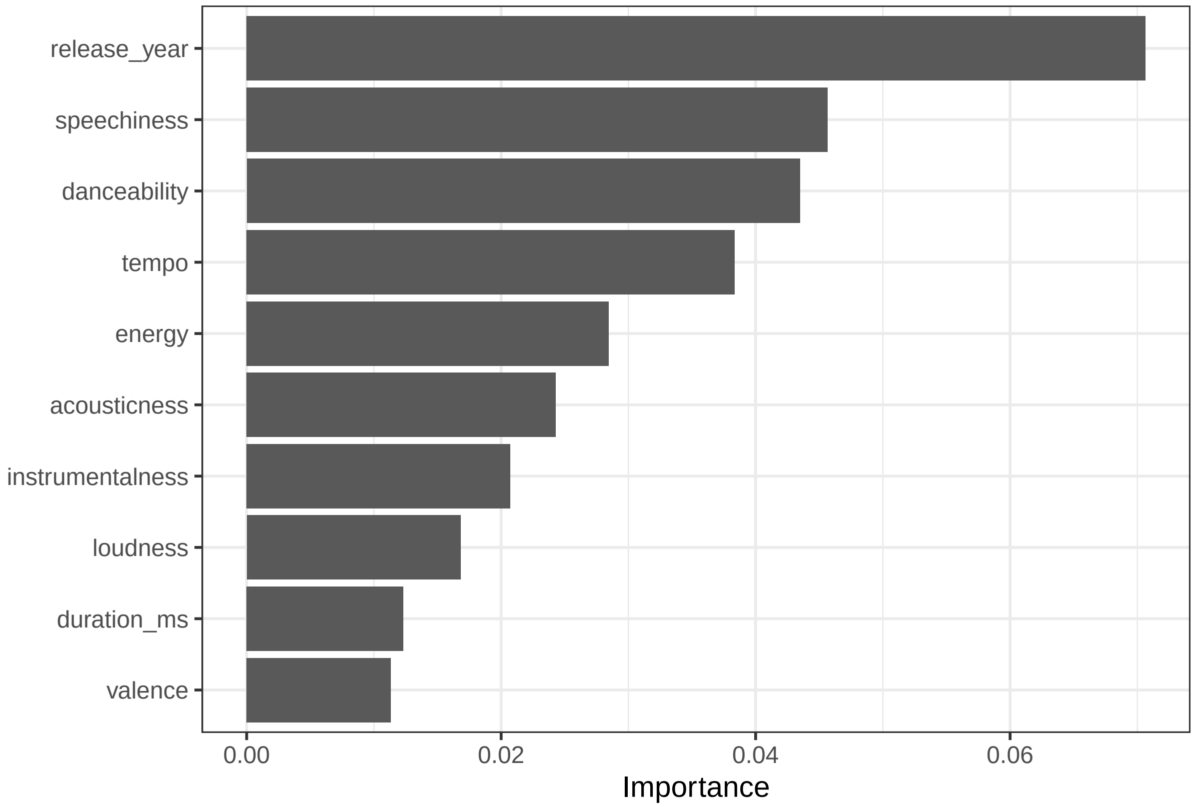

Figure 6: Variable importance plot for our random forest model

```
#test preprocessing
spotify_test_preproc <- bake(spotify_recipe, spotify_test)

rf_preds <- predict( spotify_rf, # Get class prediction
                     new_data = spotify_test_preproc,
                     type = "class" ) %>%
  bind_cols( spotify_test_preproc %>% #add on the true value
               dplyr::select( playlist_genre ) )

rf_preds %>%
  metrics( truth = playlist_genre, estimate = .pred_class )
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.541
## 2 kap      multiclass     0.450
```

```
rf_preds %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm   157    19  58  14  16    9
##      latin  12   110  36  19  23    4
```

```
##      pop    42    48  85  41  13   19
##      r&b    10    25  22  92  29   15
##      rap    22    42  21  54 166    1
##      rock    7     6  28  30   3  202
```

```
sensitivity(rf_preds, truth = playlist_genre, estimate = .pred_class) %>%
  bind_rows(specificity( rf_preds, truth = playlist_genre, estimate = .pred_class))
```

```
## # A tibble: 2 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity macro          0.541
## 2 specificity macro          0.908
```

```
#confusion matrix for edm vs others
edm_preds <- rf_preds
edm_preds$.pred_class <- fct_recode(edm_preds$.pred_class,
                                    other = "latin",
                                    other = "pop",
                                    other = "r&b",
                                    other = "rap",
                                    other = "rock")

edm_preds$playlist_genre <- fct_recode(edm_preds$playlist_genre,
                                    other = "latin",
                                    other = "pop",
                                    other = "r&b",
                                    other = "rap",
                                    other = "rock")
edm_preds %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction  edm other
##      edm    157   116
##      other   93  1134
```

```
sens(edm_preds, truth = playlist_genre, estimate = .pred_class) %>%
  bind_rows(spec( edm_preds, truth = playlist_genre, estimate = .pred_class))
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.628
## 2 spec    binary         0.907
```

```
#confusion matrix for latin vs others
latin_preds <- rf_preds
latin_preds$.pred_class <- fct_recode(latin_preds$.pred_class,
                                    other = "edm",
                                    other = "pop",
```

```
                                      other = "r&b",
                                      other = "rap",
                                      other = "rock")

latin_preds$playlist_genre <- fct_recode(latin_preds$playlist_genre,
                                      other = "edm",
                                      other = "pop",
                                      other = "r&b",
                                      other = "rap",
                                      other = "rock")
latin_preds %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction other latin
##      other  1156   140
##      latin    94   110
```

```
sens(latin_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second") %>%
  bind_rows(spec( latin_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second"))
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.44
## 2 spec    binary         0.925
```

```
#confusion matrix for pop vs others
pop_preds <- rf_preds
pop_preds$.pred_class <- fct_recode(pop_preds$.pred_class,
                                      other = "edm",
                                      other = "latin",
                                      other = "r&b",
                                      other = "rap",
                                      other = "rock")

pop_preds$playlist_genre <- fct_recode(pop_preds$playlist_genre,
                                      other = "edm",
                                      other = "latin",
                                      other = "r&b",
                                      other = "rap",
                                      other = "rock")
pop_preds %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction other  pop
##      other  1087  165
##      pop     163   85
```

```r
sens(pop_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second") %>%
  bind_rows(spec( pop_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second"))
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.34
## 2 spec    binary         0.870
```

```r
#confusion matrix for r&b vs others
`r&b_preds` <- rf_preds
`r&b_preds`$.pred_class <- fct_recode(`r&b_preds`$.pred_class,
                                      other = "edm",
                                      other = "latin",
                                      other = "pop",
                                      other = "rap",
                                      other = "rock")

`r&b_preds`$playlist_genre <- fct_recode(`r&b_preds`$playlist_genre,
                                      other = "edm",
                                      other = "latin",
                                      other = "pop",
                                      other = "rap",
                                      other = "rock")
`r&b_preds` %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction other  r&b
##      other  1149  158
##      r&b     101   92
```

```r
sens(`r&b_preds`, truth = playlist_genre, estimate = .pred_class, event_level = "second") %>%
  bind_rows(spec( `r&b_preds`, truth = playlist_genre, estimate = .pred_class, event_level = "second"))
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.368
## 2 spec    binary         0.919
```

```r
#confusion matrix for rap vs others
rap_preds <- rf_preds
rap_preds$.pred_class <- fct_recode(rap_preds$.pred_class,
                                      other = "edm",
                                      other = "latin",
                                      other = "r&b",
                                      other = "pop",
                                      other = "rock")

rap_preds$playlist_genre <- fct_recode(rap_preds$playlist_genre,
```

```r
                                   other = "edm",
                                   other = "latin",
                                   other = "r&b",
                                   other = "pop",
                                   other = "rock")
rap_preds %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction other  rap
##      other  1110   84
##      rap     140  166
```

```r
sens(rap_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second") %>%
  bind_rows(spec( rap_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second"))
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.664
## 2 spec    binary         0.888
```

```r
#confusion matrix for rock vs others
rock_preds <- rf_preds
rock_preds$.pred_class <- fct_recode(rock_preds$.pred_class,
                                   other = "edm",
                                   other = "latin",
                                   other = "r&b",
                                   other = "pop",
                                   other = "rap")

rock_preds$playlist_genre <- fct_recode(rock_preds$playlist_genre,
                                   other = "edm",
                                   other = "latin",
                                   other = "r&b",
                                   other = "pop",
                                   other = "rap")
rock_preds %>%
  conf_mat(playlist_genre, .pred_class)
```

```
##           Truth
## Prediction other rock
##      other  1176   48
##      rock     74  202
```

```r
sens(rock_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second") %>%
  bind_rows(spec( rock_preds, truth = playlist_genre, estimate = .pred_class, event_level = "second"))
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.808
## 2 spec    binary         0.941
```