Final Report

Sifan Xu

2022-08-10

Executive summary

Spotify is the world's largest music streaming service provider. Its founders are interested in trying to predict what kind of songs belong to, so as to better improve the customer experience. They are interested in how the year, speechiness, danceability and tempo of the songs will affect the genre of songs.

The founders provided a large data set and asked us to reduce the number of each kinds of songs to 1000 due to the computing power. They are also interested in the popularity of songs in different genres, the difference in speechiness for each genre and the change of track popularity over time.

Next, three models are built to predict genre based on the provided dataset which are linear discriminant analysis(LDA), K-nearest neighbours(KNN) model and random forest. Metrics such as AUC, sensitivity, specificity of the models are analyzed to select the best model.

After analysis, there are four main findings. First, besides EDM, other kinds of songs have similar popularity. Second, rap has the highest speechiness and other kinds of songs have similar speechiness. Third, Rap is becoming more and more popular, and the average popularity of all genres fluctuates over time. Fourth, it could be found that random forest model is the best model and the accuracy of the random forest model is 51.2%

Methods

The dataset is available at https://raw.githubusercontent.com/rfordatascience/ tidytuesday/master/data/2020/2020-01-21/spotify_songs.csv which contains 32833 observations and 23 variables. In order to reduce the computation cost, the number of each kinds of songs are reduced to 1000.

- Does the popularity of songs differ between genres?

In order to analyze the question, I drew a boxplot to analyze the popularity of different songs which is showed in Figure 1. The x-axis is the genre of the songs and the y-axis is the popularity.

- Is there a difference in speechiness for each genre

For this question, I also drew a boxplot shown in Figure 2 to make analysis. The x-axis is the genre of the songs and the y-axis is the speechiness.

- How does track popularity change over time?

First, I extracted the year from track_album_release_date. Next, I drew two kinds of the charts to analyze the change of popularity over time. I drew a scatter plot shown in Figure 3 to investigate what kind of songs are becoming more popular. The x-axis is the year and the y-axis is the popularity. Each type of song is represented by dots of different colors. Figure 4 shows the average popularity of all kinds of the songs over year. The x-axis is the year and the y-axis is the popularity.

Model selection

In this report, I randomly selected 1000 songs for each song for the computing power at first. After that, I chose year, speechiness, danceability and tempo as predictors which were required by Spotify. Next, I preprocess the data and all the predictor variables have mean computationally 0 and standard deviation 1. After preprocessing the data, I spilt the data into train set and test set. The train set will be used to train the model and the trained model will be tested in the test set to evaluate the metrics of the model. When testing the model, I created confusion matrix and calculate the sensitivity, specificity, accuracy and AUC of the model. The proportion of true positive samples in actual positive samples is called sensitivity and the proportion of true negative samples in actual negative samples is called specificity. AUC is the Area Under ROC curve, while ROC curve is used to test the prediction of two classifications and has little significance in multi classification problems. Therefore, in this model, I use multiclass.roc function to calculate the overall value of the AUC of each genre curve as the AUC of the multiclass model.

- Linear discriminant analysis

Ji and Ye (2008) proposed that LDA is a supervised dimension reduction method. In the process of dimension reduction, the influence of category is considered. LDA is a dimension reduction method based on the best classification effect. Therefore, the sample sets of different classes have the largest classification interval after dimension reduction. In this model, I use discrim library to built the model.The year, speechiness, danceability and tempo are predictors and the response is genre.

- K-nearest neighbours model

According to Lesmeister (2015), the k-nearest neighbor method is an inert learning algorithm, which can be used for regression and classification. Its main idea is the voting mechanism. For a test case x, find the closest k data on the labeled training data set and vote with their labels. The classification problem is voted. The regression problem uses weighted average or direct average method. In this model, it could be found that the best K is 47 after tuning and the best K is used to fit the model.

- Random forest

Nwanganga and Chapple (2020) explain how to evaluate and select the right model and how to improve model performance using integration methods such as random forest. The forest is composed of many trees, so the results of random forest depend on the results of multiple decision trees. In this model, the number of trees is 100, the mtry is 1 and the n_min is 40. Mtry defines how many predictors we wish to consider at each split, and n_min gives a stopping criterion. After fitting the model, I used vip() function to obtain the importance of the variables in the model.

Results

- Does the popularity of songs differ between genres?

From Figure 1, it could be found that the edm genre had the lowest median popularity which is approximately 36. The median popularity of r&b genre is around 46 which is higher than edm while lower than other genres. The other genres have similar median popularity which is around 50 and the pop has the highest popularity.

- Is there a difference in speechiness for each genre

From Figure 2, the median speechiness of rap is highest and it could near to 0.19. The other kinds of songs have similar speechiness at about 0.7.

- How does track popularity change over time?

Figure 3 shows that rap is becoming more and more popular. Figure 4 shows that the average popularity of all genres fluctuates over time and it is expected to continue to rise in the next few years.

- Linear discriminant analysis

After evaluating the LDA model, the overall sensitivity is 0.389, the overall specificity is 0.878, the overall AUC is 0.742, the accuracy is 0.386. According to the confusion matrix, the sensitivity of rock is highest which means that it could be easier to predict rock, while the sensitivity of r&b is lowest which means the model would mistake r&b for other genres.

- K-nearest neighbours model

In KNN model, the overall sensitivity is 0.479, the overall specificity is 0.896, the overall AUC is 0.793, the accuracy is 0.479. According to the confusion matrix, r&b and pop have the lowest sensitivity which means it is difficult for the model to predict r&b and pop correctly. The sensitivity of edm is highest which means that it could be easier to predict edm.

- Random forest

In random forest model, the overall sensitivity is 0.512, the overall specificity is 0.903, the overall AUC is 0.811, the accuracy is 0.512. According to the confusion matrix, r&b and pop have the lowest sensitivity which means it is difficult for the model to predict r&b and pop correctly. The sensitivity of edm and rock is highest which means that it could be easier to predict edm and rock. Figure 5 shows the importance of the variables and the year is of the most importance.

Discussion

The popularity of songs differs between genres, the pop has the highest median popularity over the past 60 years while the edm has the lowest. There is a difference in speechiness between rap and other genres. The speechiness of rap is almost three times as much as other genres, and other genres have the similar speechiness. With the passage of time, the popularity of different songs has also changed, rap has become more and more popular. The average popularity of all genres fluctuates over time and it is expected to continue to rise in the next few years.

After comparing the models, LDA model has the lowest AUC, sensitivity, specificity and accuracy, it is not appropriate to choose LDA model as the final model. It could be found that random forest model has the highest AUC, sensitivity, specificity and accuracy. The ability of the random forest model to predict edm and rock is similar to that of the KNN model, but the ability to predict the other four songs is stronger than that of the KNN model. Therefore, it is recommend to select random forest model as the final model.

Conclusion

According to our analysis, the median popularity of pop is the highest over the past 60 years while the edm has the lowest median popularity. The speechiness of rap is almost three times as much as other genres, and other genres have the similar speechiness. Rap has become more and more popular and the average popularity of all genres is expected to rise in the next few years. Therefore, it is suggested that put more effort at rap.

The best model is random forest model with 100 trees, mtry is 1 and the n_min is 40 which means 100 trees to be considered, 1predictor to consider at each split and stop at 40. This model could predict edm and rock fairly well while is weak in predicting r&b and pop. The overall accuracy of predicting the genre of the songs correctly is 0.512. Among the four predictors asked by Spotify, the year is most important.

References

S. Ji and J. Ye, "Generalized Linear Discriminant Analysis: A Unified Framework and Efficient Model Selection," in IEEE Transactions on Neural Networks, vol. 19, no. 10, pp. 1768-1782, Oct. 2008, doi: 10.1109/TNN.2008.2002078.

Lesmeister, C 2015, Mastering machine learning with R: master machine learning techniques with R to deliver insights for complex projects, Packt Publishing, Birmingham.

Nwanganga, FC & Chapple, M 2020, Practical machine learning in R, John Wiley and Sons, Indianapolis.

Appendix

```
#load the library
library(tidyverse)
library(dplyr)
library(rsample)
library(recipes)
library(discrim)
library(yardstick)
library(pROC)
library(caret)
library(tune)
library(kknn)
library(parsnip)
library(tidymodels)
library(vip)
```

```
#load the dataset
spotify_songs <-readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/da
```

```
## 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.
```

```
skimr::skim_without_charts(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
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

```r
df <- spotify_songs %>%
  select(playlist_genre,track_popularity,track_album_release_date,speechiness,danceability,tempo) %>%
  drop_na()

df <- df %>%
  mutate(year = substr(track_album_release_date, 1, 4))


df$playlist_genre=factor(df$playlist_genre)

skimr::skim_without_charts(df)
```

Table 4: Data summary

Name	df
Number of rows	32833
Number of columns	7
Column type frequency:	
character	2
factor	1

Table 4: Data summary

numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
track_album_release_date	0	1	4	10	0	4530	0
year	0	1	4	4	0	63	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
playlist_genre	0	1	FALSE	6	edm: 6043, rap: 5746, pop: 5507, r&b: 5431

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
track_popularity	0	1	42.48	24.98	0	24.00	45.00	62.00	100.00
speechiness	0	1	0.11	0.10	0	0.04	0.06	0.13	0.92
danceability	0	1	0.65	0.15	0	0.56	0.67	0.76	0.98
tempo	0	1	120.88	26.90	0	99.96	121.98	133.92	239.44

```r
set.seed(2022)

df_edm = df %>%
  filter(playlist_genre=="edm")
df_edm=df_edm[sample(nrow(df_edm),1000,replace=F),]

df_latin = df %>%
  filter(playlist_genre=="latin")
df_latin=df_latin[sample(nrow(df_latin),1000,replace=F),]

df_pop = df %>%
  filter(playlist_genre=="pop")
df_pop=df_pop[sample(nrow(df_pop),1000,replace=F),]

df_rb = df %>%
  filter(playlist_genre=="r&b")
df_rb=df_rb[sample(nrow(df_rb),1000,replace=F),]

df_rap = df %>%
  filter(playlist_genre=="rap")
df_rap=df_rap[sample(nrow(df_rap),1000,replace=F),]
```

```r
df_rock = df %>%
  filter(playlist_genre=="rock")
df_rock=df_rock[sample(nrow(df_rock),1000,replace=F),]

df = rbind(df_edm, df_latin, df_pop, df_rb, df_rap, df_rock)
head(df)
```

```
## # A tibble: 6 x 7
##   playlist_genre track_popularity track_album_release_~ speechiness danceability
##   <fct>                     <dbl> <chr>                       <dbl>        <dbl>
## 1 edm                           0 2014-02-17                 0.0314        0.474
## 2 edm                          39 2018-03-14                 0.0627        0.838
## 3 edm                          45 2019-08-22                 0.0695        0.552
## 4 edm                          70 2016-11-18                 0.132         0.554
## 5 edm                          39 2019-02-01                 0.183         0.808
## 6 edm                          32 2009-06-08                 0.0813        0.659
## # ... with 2 more variables: tempo <dbl>, year <chr>
```

```r
ggplot(df, aes(x=playlist_genre,y=track_popularity, fill = playlist_genre)) +
  geom_boxplot() +
  labs(title = 'Figure1: Boxplot of relationship between popularity and genres',caption = "Figure1: Box
  theme(plot.caption = element_text(hjust = 0.5))
```

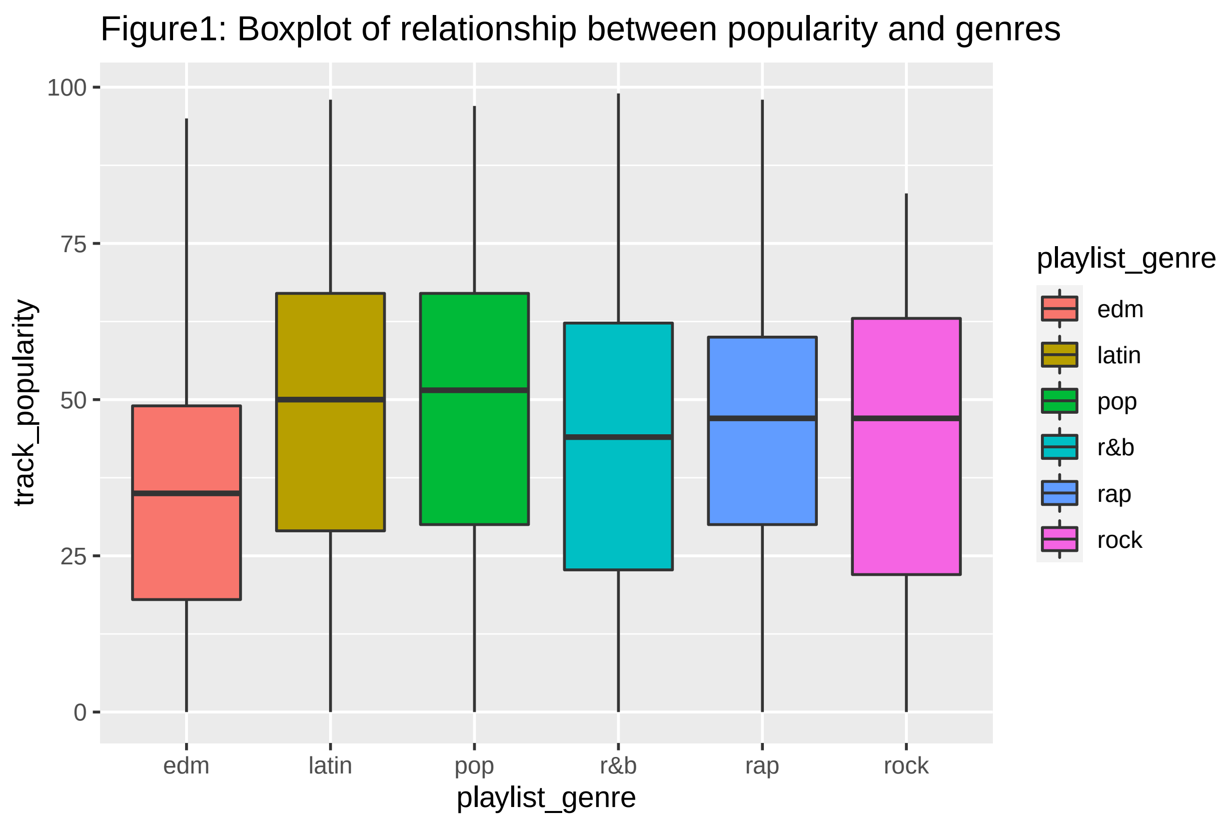

Figure1: Boxplot of relationship between popularity and genres

```r
ggplot(df, aes(x=playlist_genre,y=speechiness, fill = playlist_genre)) +
  geom_boxplot() +
  labs(title = 'Figure 2: Boxplot of relationship between speechiness and genres', caption = 'Figure 2:
  theme(plot.caption = element_text(hjust = 0.5))
```

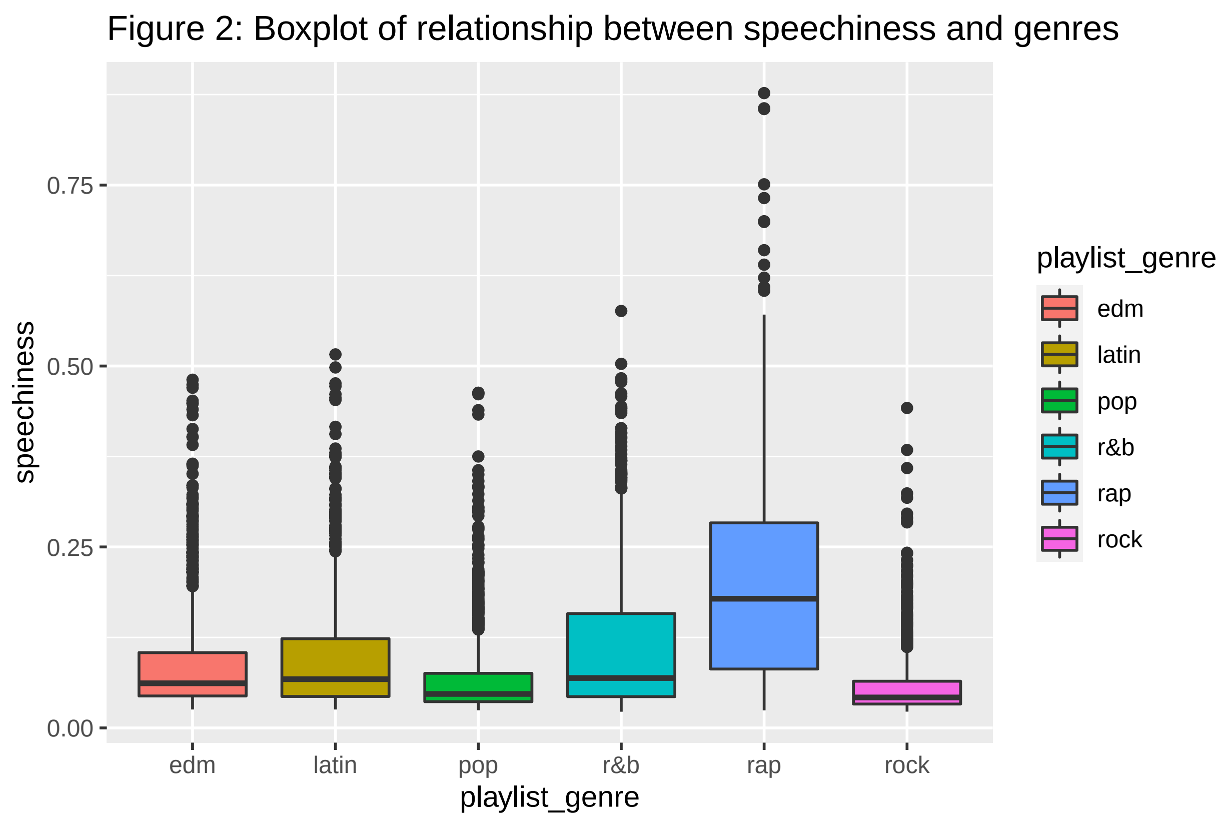

Figure 2: Boxplot of relationship between speechiness and genres

```r
ggplot(df, aes(x=year,y=track_popularity, col = playlist_genre)) +
  geom_point() +
  labs(title = 'Figure 3: Scatter plot relationship between popularity and time', caption = 'Figure 3: S
  theme(plot.caption = element_text(hjust = 0.5))
```

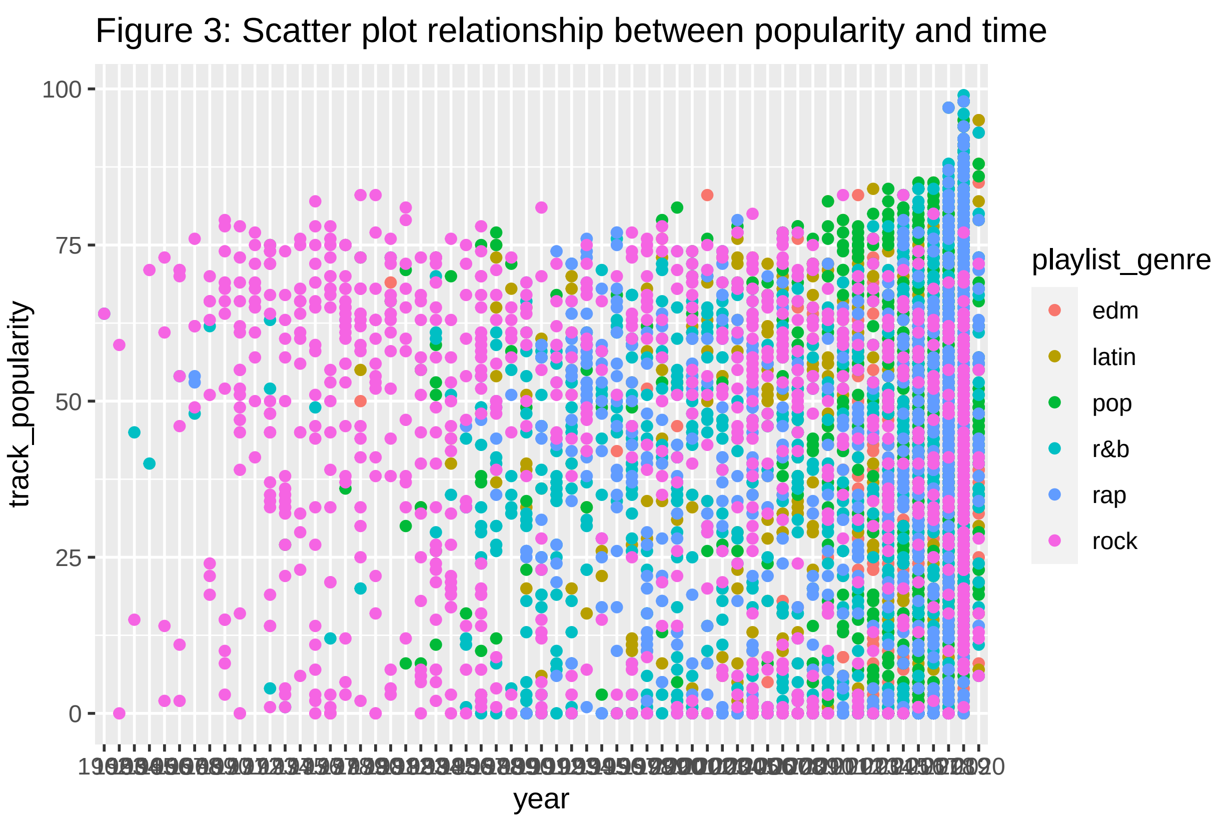

Figure 3: Scatter plot relationship between popularity and time

```r
# line of mean popularity over year
mean_popu<-df %>%
  group_by(year) %>%
  summarise(mean_popularity = mean(track_popularity))

mean_popu$year<-as.numeric(mean_popu$year)
ggplot(mean_popu, aes(x=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 = "Figure 4: Popularity of songs over time",
       caption = "Figure 4: Popularity of songs over time") +
  theme_bw() +
  theme(plot.caption = element_text(hjust = 0.5))
```

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

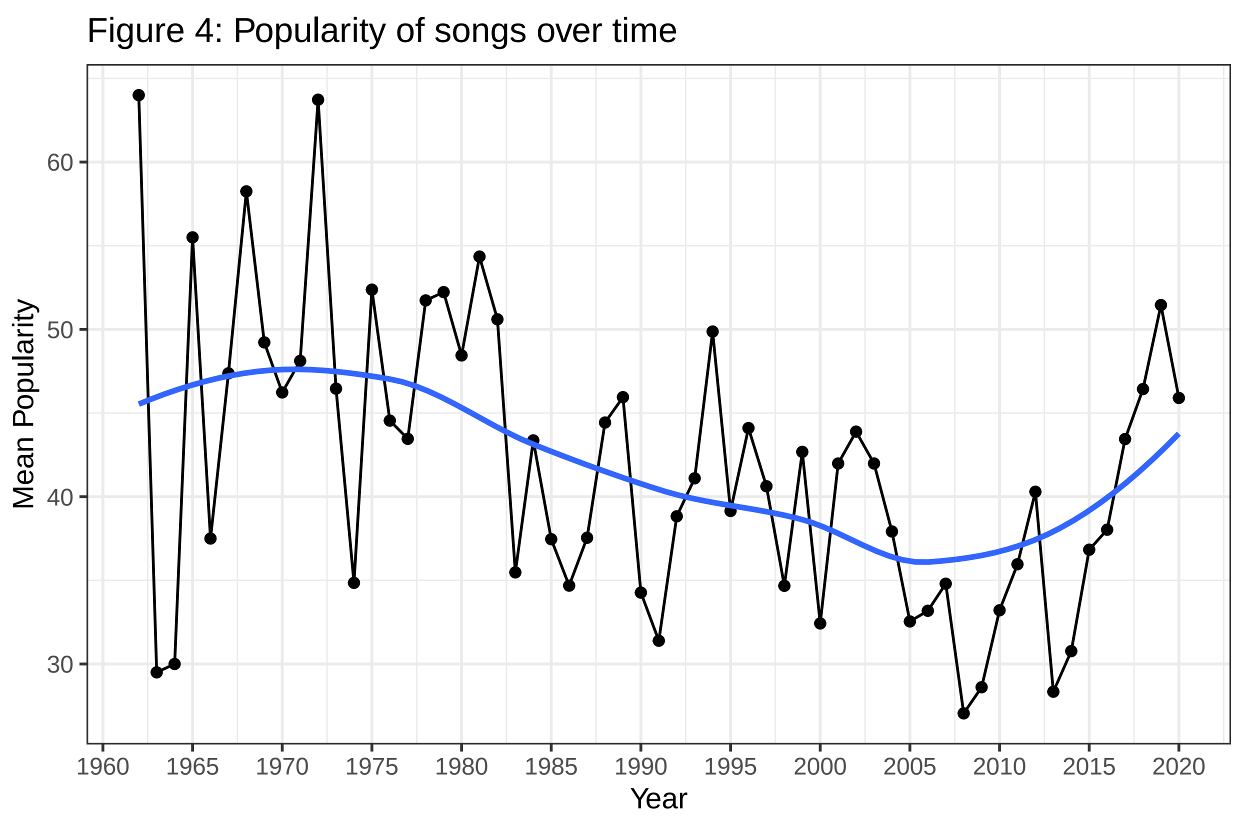

Figure 4: Popularity of songs over time

```
# create data frame for model
df_predict = rbind(df_edm, df_latin, df_pop, df_rb, df_rap, df_rock)

df_predict = df_predict %>%
  select(playlist_genre, year, speechiness, danceability,tempo)


df_predict$year <- as.numeric(df_predict$year)

df_predict <- data.frame(df_predict)

head(df_predict)
```

```
##   playlist_genre year speechiness danceability   tempo
## 1            edm 2014      0.0314        0.474 127.985
## 2            edm 2018      0.0627        0.838 128.061
## 3            edm 2019      0.0695        0.552 128.055
## 4            edm 2016      0.1320        0.554 104.957
## 5            edm 2019      0.1830        0.808 126.099
## 6            edm 2009      0.0813        0.659 117.604
```

```
#split the data
set.seed(2022)
df_split=initial_split(df_predict)
train=training(df_split)
```

```r
test=testing(df_split)
head(train)
```

```
##      playlist_genre year speechiness danceability   tempo
## 4324            rap 2017      0.3880        0.815 103.989
## 1459          latin 2019      0.1530        0.740 176.054
## 2871            pop 2010      0.0368        0.525 146.830
## 3915            r&b 2007      0.3690        0.745  83.985
## 708             edm 2018      0.1920        0.945 121.013
## 4870            rap 2019      0.1240        0.838 120.003
```

```r
#preprocess the data
train_recipe = recipe(playlist_genre~.,data=train) %>%
  themis::step_downsample( playlist_genre ) %>%
  step_dummy( all_nominal(), -all_outcomes() ) %>%
  step_normalize( all_predictors() ) %>%
prep()

train_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          4
##
## Training data contained 4500 data points and no missing data.
##
## Operations:
##
## Down-sampling based on playlist_genre [trained]
## Dummy variables from <none> [trained]
## Centering and scaling for year, speechiness, danceability, tempo [trained]
```

```r
train_preprocessed=juice(train_recipe)
test_preprocessed=bake(train_recipe,new_data=test)
skimr::skim_without_charts(train_preprocessed)
```

Table 8: Data summary

Name	train_preprocessed
Number of rows	4392
Number of columns	5
Column type frequency:	
factor	1
numeric	4

Table 8: Data summary

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
playlist_genre	0	1	FALSE	6	edm: 732, lat: 732, pop: 732, r&b: 732

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
year	0	1	0	1	-4.14	-0.24	0.44	0.70	0.78
speechiness	0	1	0	1	-0.83	-0.65	-0.42	0.23	7.69
danceability	0	1	0	1	-3.65	-0.61	0.11	0.71	2.22
tempo	0	1	0	1	-3.13	-0.79	0.03	0.48	3.46

```
skimr::skim_without_charts(test_preprocessed)
```

Table 11: Data summary

Name	test_preprocessed
Number of rows	1500
Number of columns	5
Column type frequency:	
factor	1
numeric	4
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
playlist_genre	0	1	FALSE	6	r&b: 268, edm: 256, rap: 252, roc: 249

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
year	0	1	-0.01	1.00	-4.06	-0.24	0.44	0.70	0.78
speechiness	0	1	-0.01	0.97	-0.83	-0.65	-0.46	0.27	5.92
danceability	0	1	-0.02	1.00	-3.44	-0.67	0.10	0.72	2.08
tempo	0	1	-0.01	1.00	-2.22	-0.85	0.03	0.51	3.24

```r
#train lda model
df_lda <- discrim_linear( mode = "classification" ) %>%
  set_engine( "MASS" ) %>%
  fit(playlist_genre ~ year + speechiness + danceability + tempo, data = train_preprocessed)
df_lda
```

```
## parsnip model object
##
## Call:
## lda(playlist_genre ~ year + speechiness + danceability + tempo,
##     data = data)
##
## Prior probabilities of groups:
##       edm     latin       pop       r&b       rap      rock
## 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
##
## Group means:
##             year speechiness danceability        tempo
## edm    0.4823116 -0.15714602   0.04377602  0.176307613
## latin  0.3118791 -0.04574441   0.41829116 -0.005769976
## pop    0.3027073 -0.34763095  -0.12703770 -0.013939869
## r&b   -0.1289467  0.08489865   0.16363423 -0.257788520
## rap    0.1376153  0.93138912   0.41829116 -0.027467736
## rock  -1.1055667 -0.46576639  -0.91695487  0.128658487
##
## Coefficients of linear discriminants:
##                     LD1         LD2        LD3        LD4
## year         0.83785927  0.63739900 -0.3894299 -0.3710029
## speechiness  0.36178588 -0.96671782 -0.4062666 -0.2092846
## danceability 0.70436836 -0.06283684  0.6553237  0.6280108
## tempo        0.02805214  0.12069537 -0.4958207  0.9016102
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.7211 0.2418 0.0273 0.0097
```

```r
#predict lda outcomes
lda_preds <- predict( df_lda, test_preprocessed ) %>%
  bind_cols( test_preprocessed %>% select( playlist_genre ) )
head(lda_preds)
```

```
## # A tibble: 6 x 2
##   .pred_class playlist_genre
##   <fct>       <fct>
## 1 pop         edm
## 2 rap         edm
## 3 edm         edm
## 4 pop         edm
## 5 latin       edm
## 6 pop         edm
```

```
#create confusion matrix of lda
lda_preds %>% conf_mat( truth = playlist_genre, estimate = .pred_class )
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm    62    27  38  22  34   13
##      latin  51    86  58  44  40    3
##      pop   104    45  93  47  22   57
##      r&b     1    16  10  48  17   15
##      rap    35    44  17  56 132    3
##      rock    3    12  29  51   7  158
```

```
#calculate overall sensitivity and specificity of lda
lda_preds %>%
  sens( truth = playlist_genre, estimate = .pred_class ) %>%
  bind_rows( lda_preds %>%
               spec( truth = playlist_genre, estimate = .pred_class ) )
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.389
## 2 spec    macro          0.878
```

```
#calculate accuracy
mean(test_preprocessed$playlist_genre == lda_preds$.pred_class
)
```

```
## [1] 0.386
```

```
lda_roc <- predict(df_lda, test_preprocessed, type = "prob")
colnames(lda_roc) <- c("edm","latin","pop","r&b","rap","rock")
lda_roc <- as.matrix(lda_roc)
head(lda_roc)
```

```
##            edm     latin        pop       r&b        rap        rock
## [1,] 0.2833307 0.1691566 0.30804048 0.1419859 0.05704242 0.040443947
## [2,] 0.1776982 0.2649749 0.09827503 0.1400920 0.31699326 0.001966576
## [3,] 0.2661771 0.2006900 0.26342696 0.1601884 0.07867491 0.030842638
## [4,] 0.2361437 0.1580513 0.29811229 0.1648559 0.04454316 0.098293643
## [5,] 0.2380776 0.2780318 0.22987496 0.1730587 0.07149926 0.009457595
## [6,] 0.2641497 0.1140392 0.35777468 0.1271624 0.03501745 0.101856621
```

```
#calculate overall auc of lda
lda_auc <- multiclass.roc(test_preprocessed$playlist_genre,lda_roc)
lda_auc
```

```
##
## Call:
## multiclass.roc.default(response = test_preprocessed$playlist_genre,     predictor = lda_roc)
##
## Data: multivariate predictor lda_roc with 6 levels of test_preprocessed$playlist_genre: edm, latin, 
## Multi-class area under the curve: 0.742
```

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

set.seed( 1223 )
genre_cv <- vfold_cv( train_preprocessed, v = 5 )

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

knn_tune <- tune_grid(object = knn_spec, preprocessor = recipe(playlist_genre ~ ., data = train_preproc

best_acc <- select_best( knn_tune, "accuracy")

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

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 47
##
## Computational engine: kknn
```

```
df_knn <- knn_spec_final %>%
  fit( playlist_genre ~ . , data = train_preprocessed )

df_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = playlist_genre ~ ., data = data, ks = min_rows(47L,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.5243625
## Best kernel: optimal
## Best k: 47
```

```
#show knn model outcomes
knn_preds <- predict( df_knn, test_preprocessed, type = "class") %>% bind_cols( select( test_preprocess
head(knn_preds)
```

```
## # A tibble: 6 x 2
##   .pred_class playlist_genre
##   <fct>       <fct>
## 1 edm         edm
## 2 edm         edm
## 3 edm         edm
## 4 edm         edm
## 5 pop         edm
## 6 edm         edm
```

```r
#create confusion matrix of knn model
knn_preds %>% conf_mat( truth = playlist_genre, estimate = .pred_class )
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm   182    24  47  29  27   15
##      latin  17   100  54  47  42    3
##      pop    26    38  69  33  12   38
##      r&b     2    20  14  66  30   17
##      rap    19    34  23  52 131    6
##      rock   10    14  38  41  10  170
```

```r
#calculate overall sensitivity and specificity of knn
knn_preds %>%
  sens( truth = playlist_genre, estimate = .pred_class ) %>%
  bind_rows( knn_preds %>%
               spec( truth = playlist_genre, estimate = .pred_class ) )
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.479
## 2 spec    macro          0.896
```

```r
#show accuracy of knn model
mean(test_preprocessed$playlist_genre == knn_preds$.pred_class
)
```

```
## [1] 0.4786667
```

```r
knn_roc <- predict(df_knn, test_preprocessed, type = "prob")
colnames(knn_roc) <- c("edm","latin","pop","r&b","rap","rock")
knn_roc <- as.matrix(knn_roc)
head(knn_roc)
```

```
##            edm      latin        pop         r&b        rap        rock
## [1,] 0.6440008 0.10824537 0.08600163 0.047528774 0.00000000 0.114223390
## [2,] 0.4263408 0.12046707 0.04777777 0.221586458 0.18382792 0.000000000
## [3,] 0.6582008 0.02962267 0.12428121 0.060018481 0.05735061 0.070526233
## [4,] 0.7552865 0.01885687 0.21590869 0.003132899 0.00000000 0.006815082
## [5,] 0.2230367 0.27976305 0.29007199 0.156765494 0.02764605 0.022716720
## [6,] 0.5146909 0.04280635 0.29995719 0.095855411 0.01500562 0.031684523
```

```r
#calcualte overall auc of knn
knn_auc <- multiclass.roc(test_preprocessed$playlist_genre,knn_roc)
knn_auc
```

```
##
## Call:
## multiclass.roc.default(response = test_preprocessed$playlist_genre,     predictor = knn_roc)
##
## Data: multivariate predictor knn_roc with 6 levels of test_preprocessed$playlist_genre: edm, latin, p
## Multi-class area under the curve: 0.7927
```

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

set.seed(2022)
df_boots <- bootstraps(train_preprocessed, times = 10, strata = playlist_genre)

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

set.seed(2022)
rf_grid <- tune_grid( object = rf_spec,
                      preprocessor = recipe(playlist_genre ~ . , data = train_preprocessed),
                      resamples = df_boots,
                      grid = rand_spec_grid )

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 ) +
  labs(caption = "Figure 5: acc and auc")+
  theme(plot.caption = element_text(hjust = 0.5))
```

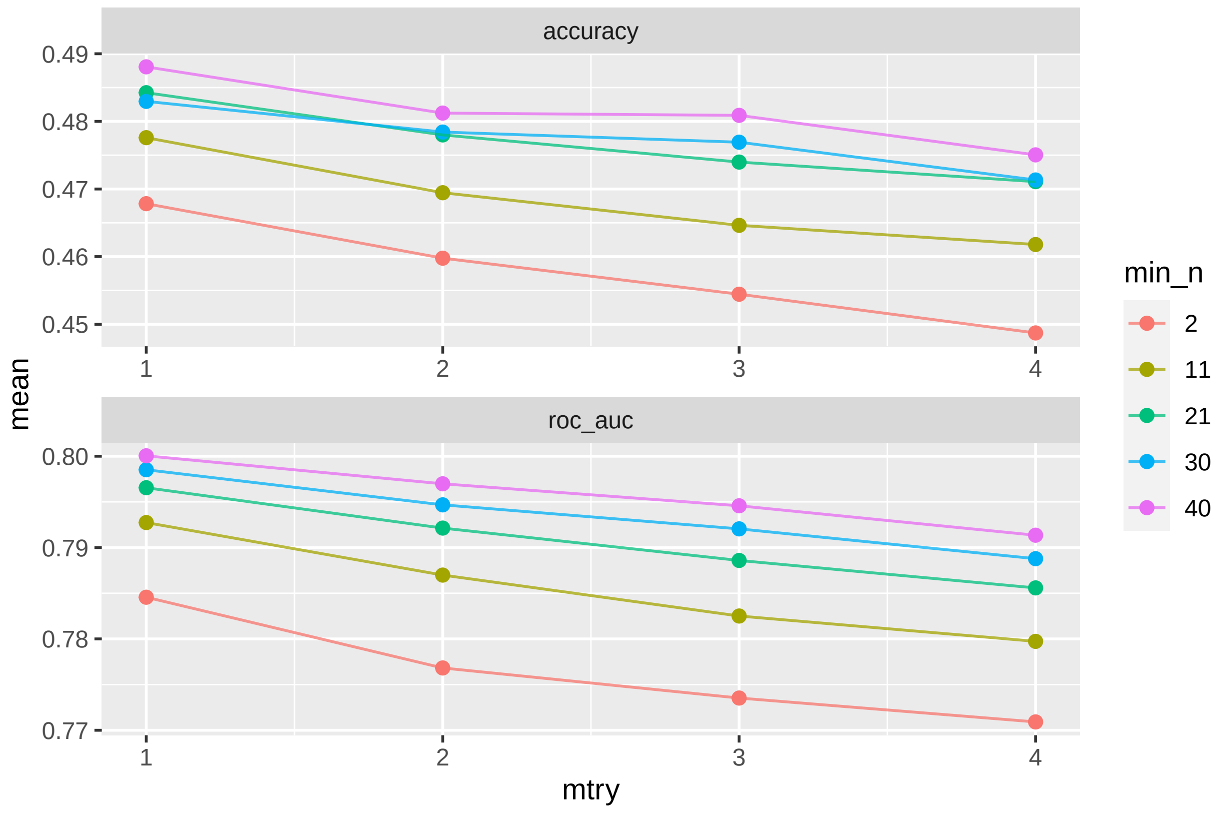

Figure 5: acc and auc

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

rf_final <- finalize_model( rf_spec, best_rf_acc )

df_rf <- rf_final %>%
  fit( playlist_genre~., data = train_preprocessed )
df_rf
```

```
## parsnip model object
##
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~1L,      x), num.trees = ~100, min.
##
## Type:                             Probability estimation
## Number of trees:                  100
## Sample size:                      4392
## Number of independent variables:  4
## Mtry:                             1
## Target node size:                 40
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.4823718
```

```
rf_preds <- predict( df_rf, test_preprocessed ) %>%
  bind_cols( test_preprocessed %>% select( playlist_genre ) )
head(rf_preds)
```

```
## # A tibble: 6 x 2
##   .pred_class playlist_genre
##   <fct>       <fct>
## 1 edm         edm
## 2 edm         edm
## 3 edm         edm
## 4 edm         edm
## 5 latin       edm
## 6 edm         edm
```

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

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm   182    17  30  17  23    6
##      latin  14   101  50  32  33    6
##      pop    28    38  82  36  14   30
##      r&b     3    18  25  76  27   20
##      rap    18    39  18  64 146    6
##      rock   11    17  40  43   9  181
```

```
#overall sensitivity and specificity of rf
rf_preds %>%
  sens( truth = playlist_genre, estimate = .pred_class ) %>%
  bind_rows( rf_preds %>%
               spec( truth = playlist_genre, estimate = .pred_class ) )
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.512
## 2 spec    macro          0.903
```

```
#accuracy of rf
mean(test_preprocessed$playlist_genre == rf_preds$.pred_class
)
```

```
## [1] 0.512
```

```
rf_roc <- predict(df_rf, test_preprocessed, type = "prob")
colnames(rf_roc) <- c("edm","latin","pop","r&b","rap","rock")
rf_roc <- as.matrix(rf_roc)
head(rf_roc)
```

```
##            edm      latin        pop        r&b         rap        rock
## [1,] 0.6686528 0.06605847 0.10190202 0.04233672 0.041866539 0.079183480
## [2,] 0.4733084 0.11111448 0.07716536 0.18827907 0.145710596 0.004422125
## [3,] 0.7094675 0.07811359 0.09061289 0.04161180 0.037337383 0.042856833
## [4,] 0.7188120 0.04671307 0.18665378 0.01791553 0.002417341 0.027488265
## [5,] 0.1108191 0.39630230 0.23449009 0.16245546 0.060439817 0.035493189
## [6,] 0.6271378 0.04991957 0.14361753 0.05626854 0.029974537 0.093082054
```

```
#overall auc of rf
rf_auc <- multiclass.roc(test_preprocessed$playlist_genre,rf_roc)
rf_auc
```

```
##
## Call:
## multiclass.roc.default(response = test_preprocessed$playlist_genre,     predictor = rf_roc)
##
## Data: multivariate predictor rf_roc with 6 levels of test_preprocessed$playlist_genre: edm, latin, p
## Multi-class area under the curve: 0.8108
```

```
#importance of variables in rf
df_rf %>%
  vip( ) +
  labs(caption = "Figure 6: Importance variable for random forest model")+
  theme(plot.caption = element_text(hjust = 0.5))
```

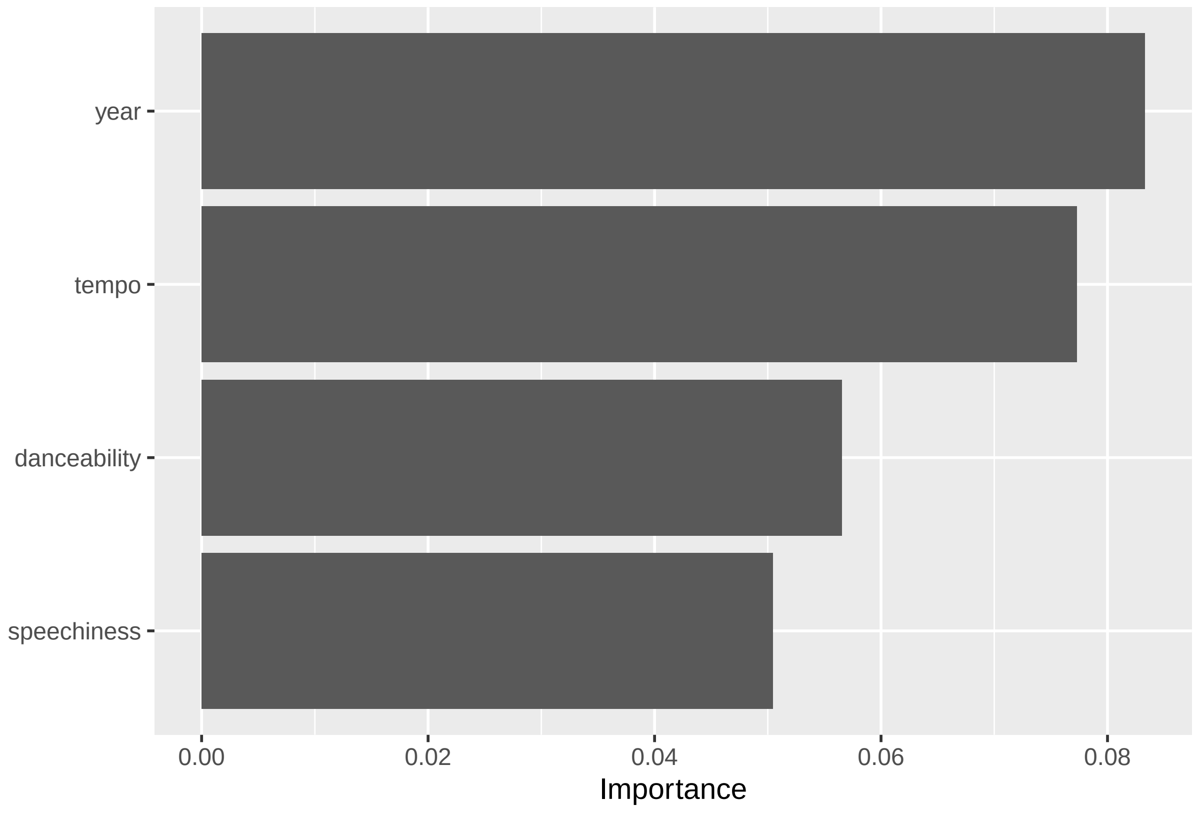

Figure 6: Importance variable for random forest model