Final Report

Prithviraj Banerjee

2022-08-21

Executive Summary

In order to better enhance their customer's experience and ultimately remain the best music streaming service available, founders of Spotify are interested in how they can better predict the genre of a song based on the year of release, speechiness, danceability of the song & also the song tempo. After a full analysis, it is noted that track popularity actually differ across genres e.g. pop genre is more popular than others whereas edm is not that popular. Similar goes for speechiness as rap songs are seen to be more speechy than others (which is also quite expected). However different genres became more popular in different time points for example in the early 1970's rock was more popular, as time passed by, in the mid 90's Latin, rock & blues songs took over and then in the late 2000 edm music came into the picture and now they are more popular than others which is natural keeping in mind the progress of electronic music and powerful softwares. In order to predict the genre of a particular song based on the covariates, we used several classification models and noticed that random forest model is giving the best performance.

Methods

(Bivariate summaries, model tuning, Model selection)

As advised, only the following variables were considered :-

- year of the song released.
- speechiness of the song.
- danceability.
- tempo.
- track popularity.
- track genre.

where, track genre was our categorical response variable that we want to predict. No other variables are considered in the following analysis which was completed in R version 4.2.0 using the tidyverse and dplyr packages. The first step for reviewing the data was to consider the dependence of the track genre based on all the covariates.

- Here we saw that the average track popularity was maximum for pop genre, whereas it is minimum for edm genre across the years.
- Next we did the same for the variables speechiness and saw that average speechiness value is higher for rap genre. It is also very natural since rap songs mainly involve lyrics.
- Average Tempo across different genres is more or less same. Though it is interesting to note that in rap genre all the songs have more or less the same tempo range. But in edm genre, the tempo can vary in a huge range of values.
- It is difficult to find any trend in track popularity across different times, from the scatterplot only. For we fitted linear trend models to note that rock genre was the most popular in the early 1970s and in the meantime Latin pop rap these genres became more popular in the early 1990s and specifically edm genre which was introduced late 1980s, rapidly gained popularity over time. Now most of the songs belong to this genre.

Results

After a thorough analysis, we found that all the predictors are important in terms of their predicting power. In order to answer the specific questions, we fitted linear models.

- We fitted a model, where we predict a track popularity using playlist genre and compare it with only intercept model to find that the first model is highly significant. This gives evidence in support of the fact that popularity truly varies across genres.
- We did the same as taking speechiness as the response and got that it is also very significant.
- We fitted a linear interaction model for predicting the popularity using variables track year and playlist genre along with their interaction terms and found it is more significant than the model only considering track year as a predictor.
- Next we fitted a LDA model for predicting playlist genre for all the covariates. We found that the model was correctly classifying 620 observations in the test data set out of 1500 cases. Since, we cannot tune any parameter in the LDA model, this is the best performance we can achieve here.
- Here, we fitted the KNN model and as instructed we took 20 levels in the range 1 to 100 in different choices of k. After tuning we found that for k=15, the model performance was the best in terms of auc. This knn model correctly classified 684 observations in the test data set out of 1500 cases which is a significant improvement in terms of the previous model.
- Lastly, we fit a random forest model with 100 trees and for tuning we consider 5 different levels for each of the parameters mtry and min_n. After tuning we found that this the best model for mtry = 1 and min_n = 5. Here also we saw a significant improvement as it classified 748 observations out of 1500 cases.

Discussion

Below we represent the different performance metrices for all the three models: We can clearly see that the Random Forest is the best model as it has highest specificity, lowest False Positive and False Negative rates across all the genres.

	Specificity	FPR	FNR
LDA	0.784	0.216	0.582
KNN	0.810	0.190	0.540
RF	0.828	0.172	0.504

Conclusions

We have already discussed how popularity and speechiness vary across different genres. We also saw the dependence of track popularity over time. Lastly, since our goal was to predict the track genre based on the selected predictors, out of the three proposed classifiers i.e. LDA, KNN, Random Forest, we recommend using the random forest model as its performance was best among all three in terms of all the comparison metrices. In fact, the model may be improved furthur if we consider other covariates also and we increase the number of trees and then tune the hyper parameters. As these models are very complex in their structure, we cannot exactly interpret the relationship between predictors and the response.

Appendix

```
options(digits = 3)
library(ggplot2)
```

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

data <- data %>%
  select(playlist_genre,track_album_release_date,track_popularity,danceability,speechiness,tempo)

data <- data %>%
  mutate(track_year = str_match(track_album_release_date,"[0-9]{1,4}"))

data <- data %>%
  mutate(playlist_genre = as_factor(playlist_genre))

# Counting the number of each genre
data %>%
  count(playlist_genre)
```

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

```
data <- data %>% group_by(playlist_genre)

# Slicing the data to get 1000 count for each genre
data_slice <- data %>% slice_sample(n = 1000)

# Now it can be checked
data_slice %>% count(playlist_genre)
```

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

```
names(data_slice)[which(names(data_slice) == "track_year[,1]")] <- "track_year"

data_slice <- data_slice %>% mutate(track_year = as.numeric(track_year))
```

```
# popularity differ between genres
ggplot(data_slice,aes(x = playlist_genre,y = track_popularity,fill = playlist_genre)) +
  geom_boxplot()
```

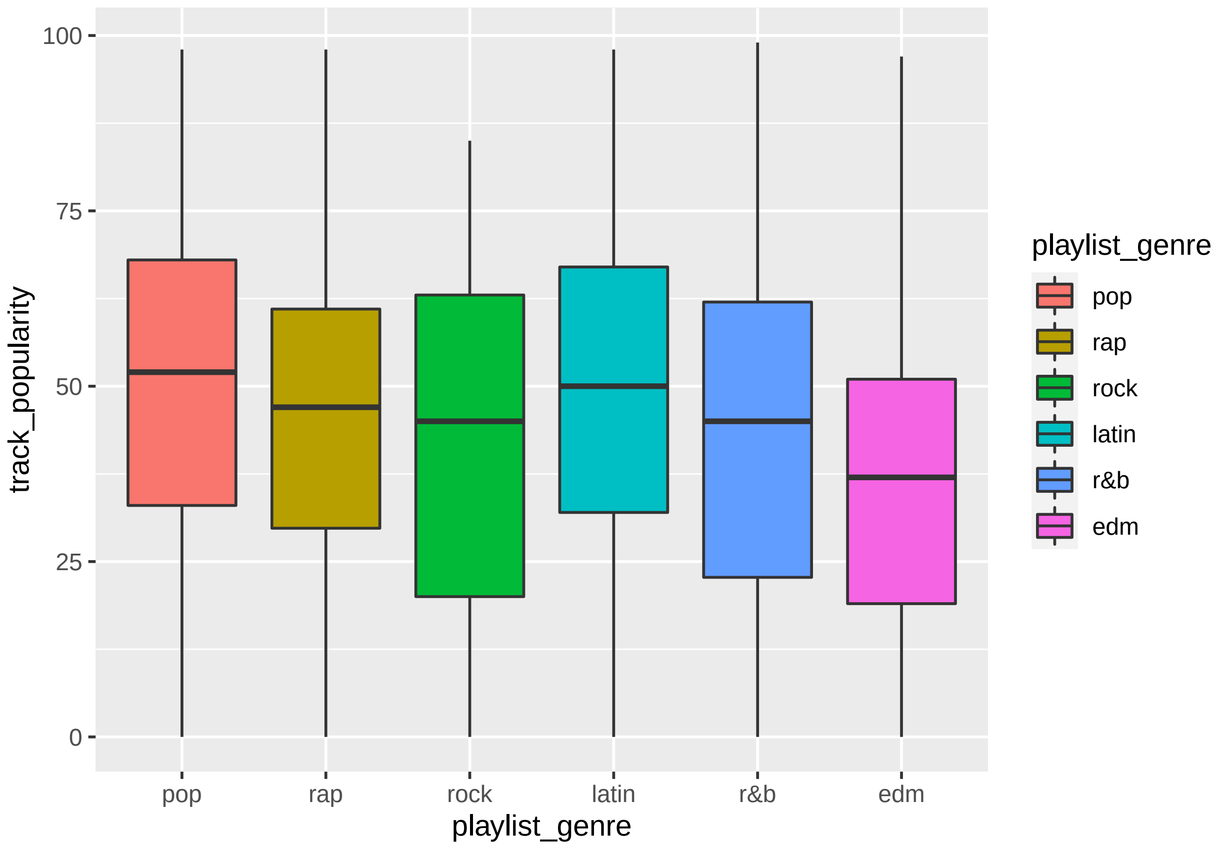


Figure 1: Track_Popularity across different genres

```
# difference in speechiness
ggplot(data_slice,aes(x = playlist_genre,y = speechiness,fill = playlist_genre)) +
  geom_boxplot()
```

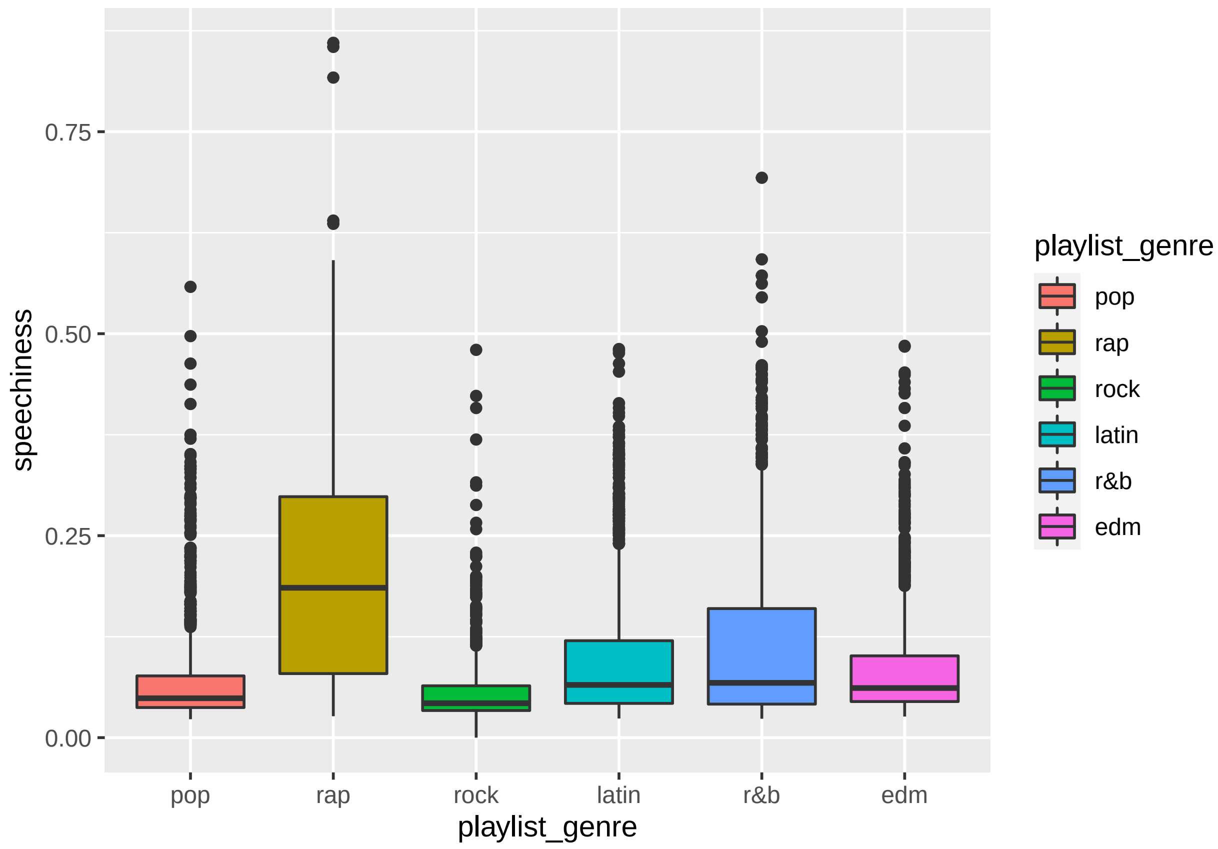


Figure 2: Speechiness across different genres

```
# difference in tempo
ggplot(data_slice,aes(x = playlist_genre,y = tempo,fill = playlist_genre)) +
  geom_boxplot()
```

```
# difference in popularity accross years
ggplot(data_slice,aes(x = track_year,y = track_popularity)) +
  geom_point() + facet_grid(rows = vars(playlist_genre))
```

```
ggplot(data_slice,aes(x = as.numeric(track_year),y = as.numeric(track_popularity),colour = factor(playli
  geom_smooth(method = "lm", se = FALSE)
```

```
# popularity
model1 <- lm(track_popularity ~ playlist_genre,data = data_slice)
model0 <- lm(track_popularity ~ 1,data = data_slice)
anova(model0,model1)
```

```
## Analysis of Variance Table
##
## Model 1: track_popularity ~ 1
## Model 2: track_popularity ~ playlist_genre
##   Res.Df     RSS Df Sum of Sq    F Pr(>F)
```

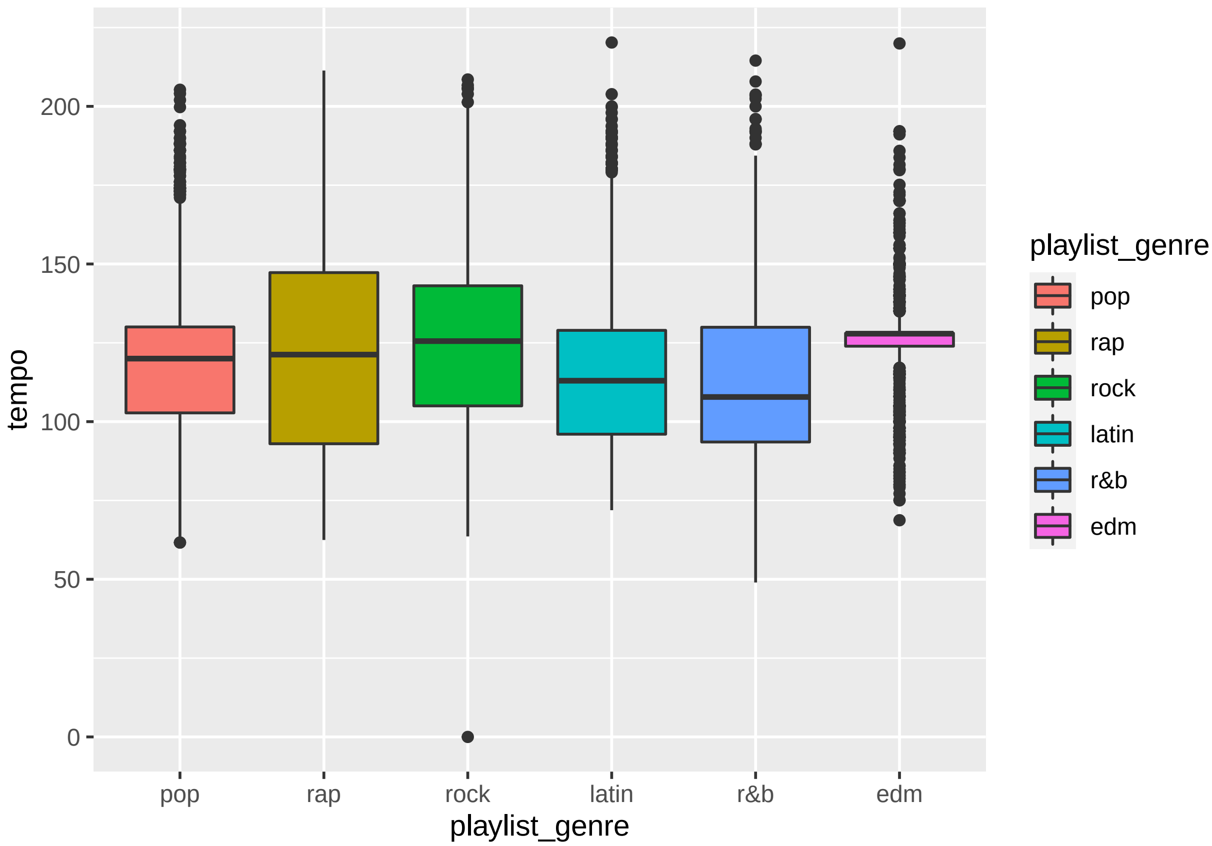


Figure 3: Tempo across different genres

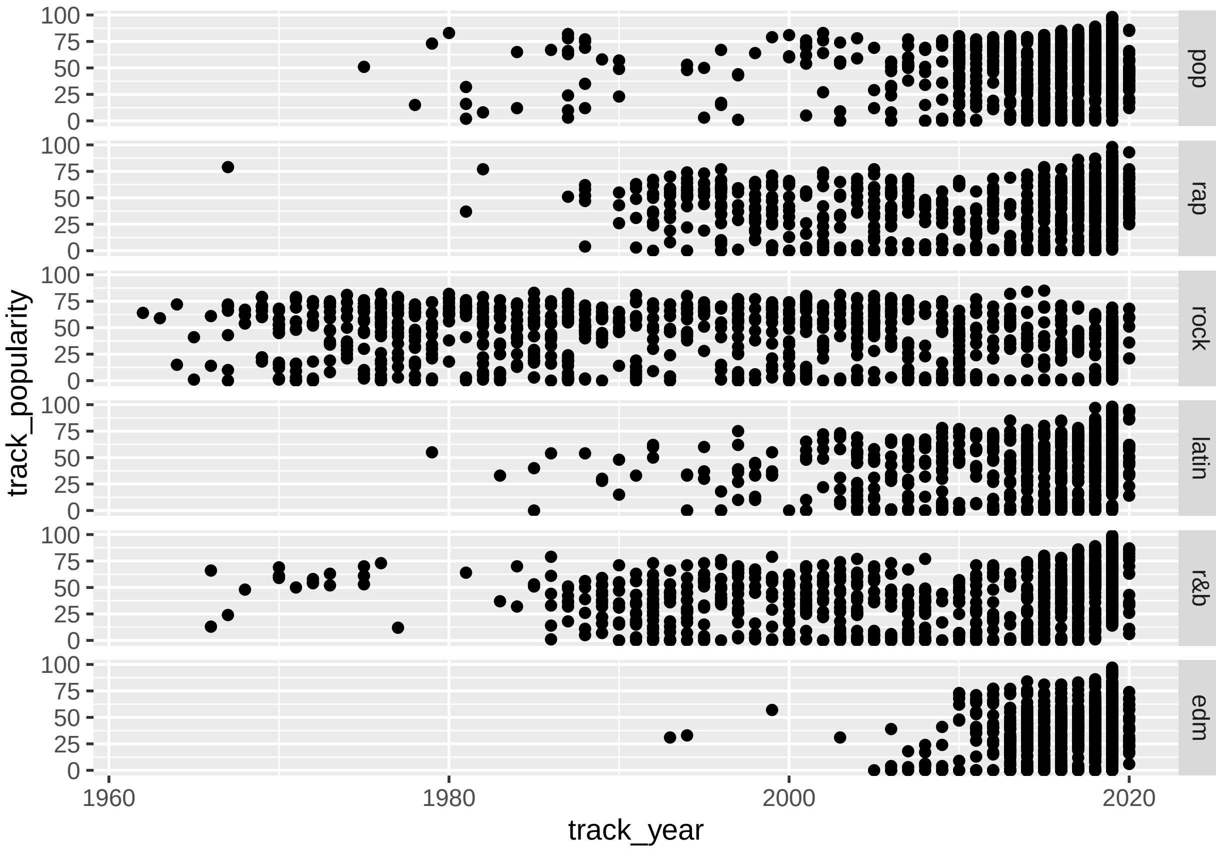


Figure 4: Popularity change with time

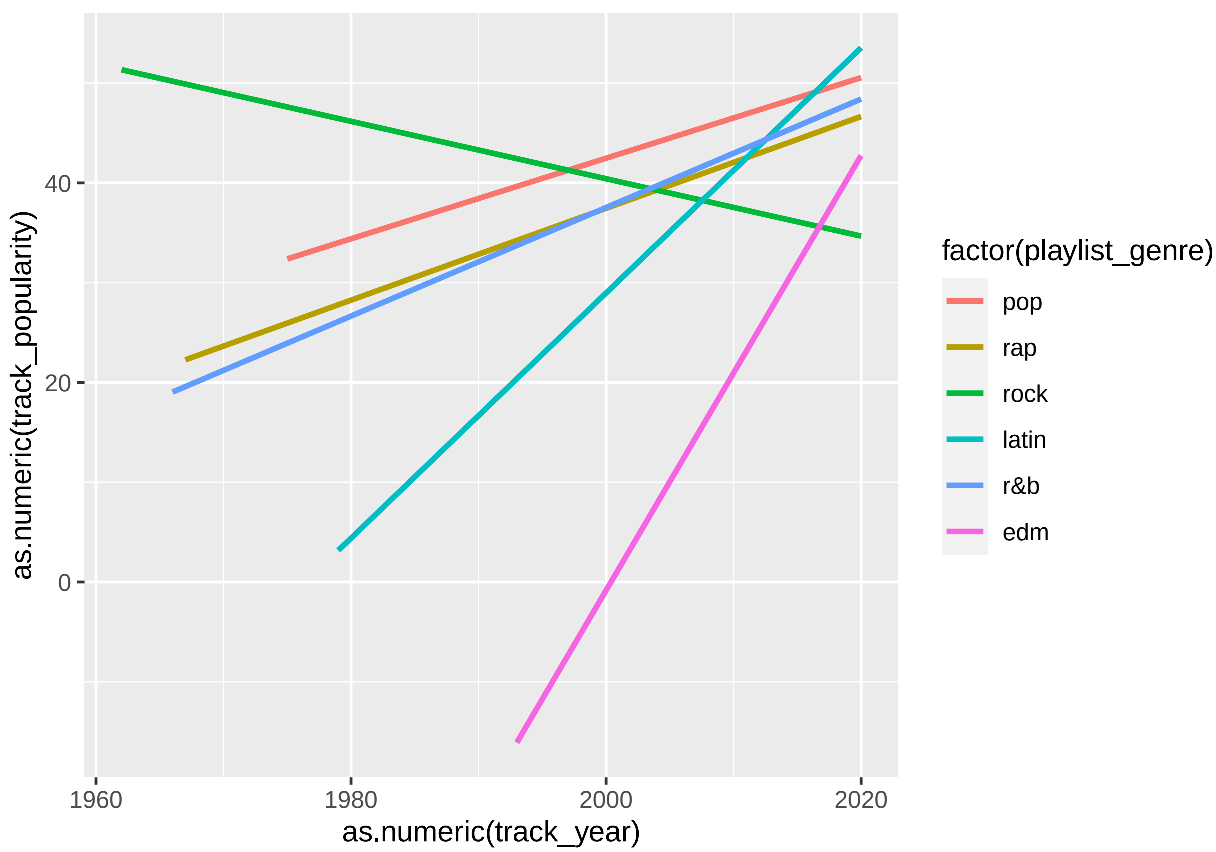


Figure 5: Popularity change with time

```
## 1   5999 3696290
## 2   5994 3592063  5    104227 34.8 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model1)
```

```
##
## Call:
## lm(formula = track_popularity ~ playlist_genre, data = data_slice)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -48.37 -17.18   2.82  18.88  61.48
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           48.368      0.774   62.48  < 2e-16 ***
## playlist_genrerap     -5.101      1.095   -4.66  3.2e-06 ***
## playlist_genrerock    -7.192      1.095   -6.57  5.5e-11 ***
## playlist_genrelatin   -1.399      1.095   -1.28      0.2
## playlist_genrer&b     -5.875      1.095   -5.37  8.3e-08 ***
## playlist_genreedm    -12.852      1.095  -11.74  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.5 on 5994 degrees of freedom
## Multiple R-squared:  0.0282, Adjusted R-squared:  0.0274
## F-statistic: 34.8 on 5 and 5994 DF,  p-value: <2e-16
```

```
# speechinees
model1 <- lm(speechiness ~ playlist_genre,data = data_slice)
model0 <- lm(speechiness ~ 1,data = data_slice)
anova(model0,model1)
```

```
## Analysis of Variance Table
##
## Model 1: speechiness ~ 1
## Model 2: speechiness ~ playlist_genre
##   Res.Df  RSS Df Sum of Sq   F Pr(>F)
## 1   5999 61.3
## 2   5994 48.0  5      13.3 333 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# year
model1 <- lm(track_popularity ~ track_year*playlist_genre,data = data_slice)
model0 <- lm(track_popularity ~ track_year,data = data_slice)
anova(model0,model1)
```

```
## Analysis of Variance Table
##
## Model 1: track_popularity ~ track_year
## Model 2: track_popularity ~ track_year * playlist_genre
##   Res.Df     RSS Df Sum of Sq    F Pr(>F)
## 1   5998 3675148
```

```
## 2   5988 3405167 10    269981 47.5 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model1)
```

```
##
## Call:
## lm(formula = track_popularity ~ track_year * playlist_genre,
##     data = data_slice)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -52.31 -16.76   2.84  18.70  60.02
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -7.65e+02   2.29e+02   -3.34  0.00084 ***
## track_year                     4.04e-01   1.14e-01    3.55  0.00039 ***
## playlist_genrerap             -1.18e+02   2.89e+02   -0.41  0.68342
## playlist_genrerock             1.38e+03   2.47e+02    5.59  2.4e-08 ***
## playlist_genrelatin           -1.66e+03   3.30e+02   -5.04  4.9e-07 ***
## playlist_genrer&b             -2.85e+02   2.66e+02   -1.07  0.28315
## playlist_genreedm             -3.59e+03   5.55e+02   -6.48  9.9e-11 ***
## track_year:playlist_genrerap    5.65e-02   1.44e-01    0.39  0.69421
## track_year:playlist_genrerock  -6.91e-01   1.23e-01   -5.63  1.8e-08 ***
## track_year:playlist_genrelatin  8.25e-01   1.64e-01    5.03  5.0e-07 ***
## track_year:playlist_genrer&b    1.40e-01   1.32e-01    1.06  0.28840
## track_year:playlist_genreedm    1.78e+00   2.75e-01    6.45  1.2e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.8 on 5988 degrees of freedom
## Multiple R-squared:  0.0788, Adjusted R-squared:  0.0771
## F-statistic: 46.5 on 11 and 5988 DF,  p-value: <2e-16
```

```
library(tidymodels)
set.seed(2022)
data_split <- initial_split(data_slice)

data_train <- training(data_split)
data_test <- testing(data_split)

head(data_train)
```

```
## # A tibble: 6 x 7
## # Groups:   playlist_genre [5]
##   playlist_genre track_album_release_~ track_popularity danceability speechiness
##   <fct>          <chr>                            <dbl>        <dbl>       <dbl>
## 1 r&b            1995                                58        0.702      0.035
## 2 rap            2017-08-19                          59        0.896      0.0544
## 3 rock           2004                                69        0.606      0.0598
## 4 latin          2018-10-05                          59        0.689      0.111
## 5 pop            2017-04-07                          65        0.615      0.0291
## 6 r&b            2015-04-22                          44        0.588      0.194
```

```
## # ... with 2 more variables: tempo <dbl>, track_year <dbl>
```

```
# Making the recipe
#rm(data_rec)
data_rec <- recipe(playlist_genre ~ track_popularity+danceability+speechiness+tempo+track_year,data = da

data_rec <- data_rec %>%
  step_dummy(all_nominal_predictors()) %>% step_normalize(all_numeric_predictors()) %>%
  prep()


data_juiced <- juice(data_rec)

data_bake <- bake(data_rec, new_data = data_test)
```

```
# LDA MOdel Fitting
library(discrim)
library(MASS)

data_lda_tidy <- discrim_linear( mode = "classification" ) %>%
  set_engine( "MASS" ) %>%
  fit(playlist_genre ~ track_popularity+danceability+speechiness+tempo+track_year,data = data_train)

pred_data_lda <- predict(data_lda_tidy,data_test,type = "class")

pred_data_lda <- pred_data_lda %>%
  bind_cols(data_test$playlist_genre)
names(pred_data_lda) <- c("predicted","observed")

# Confusion matrix
tab_lda = table(pred_data_lda$observed,pred_data_lda$predicted,dnn = c("obs","pred"))
```

```
library(kknn)
# KNN model
set.seed(2022)
knn_cv_splits=vfold_cv(data_train,v=5)

knn_tune = parameters(neighbors(range = c(1,100)))

knn_mod = nearest_neighbor() %>%
  set_engine("kknn") %>%
  set_mode("classification") %>%
  set_args(neighbors = tune())

knn_tuned = tune::tune_grid(knn_mod,preprocessor = data_rec,resamples =
                    knn_cv_splits,control = tune::control_resamples(save_pred = TRUE))
knn_tuned %>%
  select_best(metric = "roc_auc",n = 5)
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        15 Preprocessor1_Model10
```

```
knn_mod_best = nearest_neighbor() %>%
  set_engine("kknn") %>%
```

```r
  set_mode("classification") %>%
  set_args(neighbors = 15)

wflow_knn <- workflow() %>%
  add_model(knn_mod_best) %>%
  add_recipe(data_rec)

knn_fit <- fit(wflow_knn, data_juiced)

knn_pred <- predict(knn_fit, new_data = data_bake)

tab_knn = table(data_bake$playlist_genre,knn_pred$.pred_class,dnn = c("obs","pred"))
```

```r
library(ranger)
# Random Forest
tune_spec <- rand_forest(
  mtry = tune(),
  trees = 100,
  min_n = tune()
) %>%
  set_mode("classification") %>%
  set_engine("ranger")

tune_wf <- workflow() %>%
  add_recipe(data_rec) %>%
  add_model(tune_spec)

set.seed(2022)
trees_folds <- vfold_cv(data_train,v = 5)

rf_grid <- grid_regular(
  mtry(range = c(1, 5)),
  min_n(range = c(2, 8)),
  levels = 5
)

# you have to change the levels parameter from 3 to 5 as asked in the problem

rf_grid
```

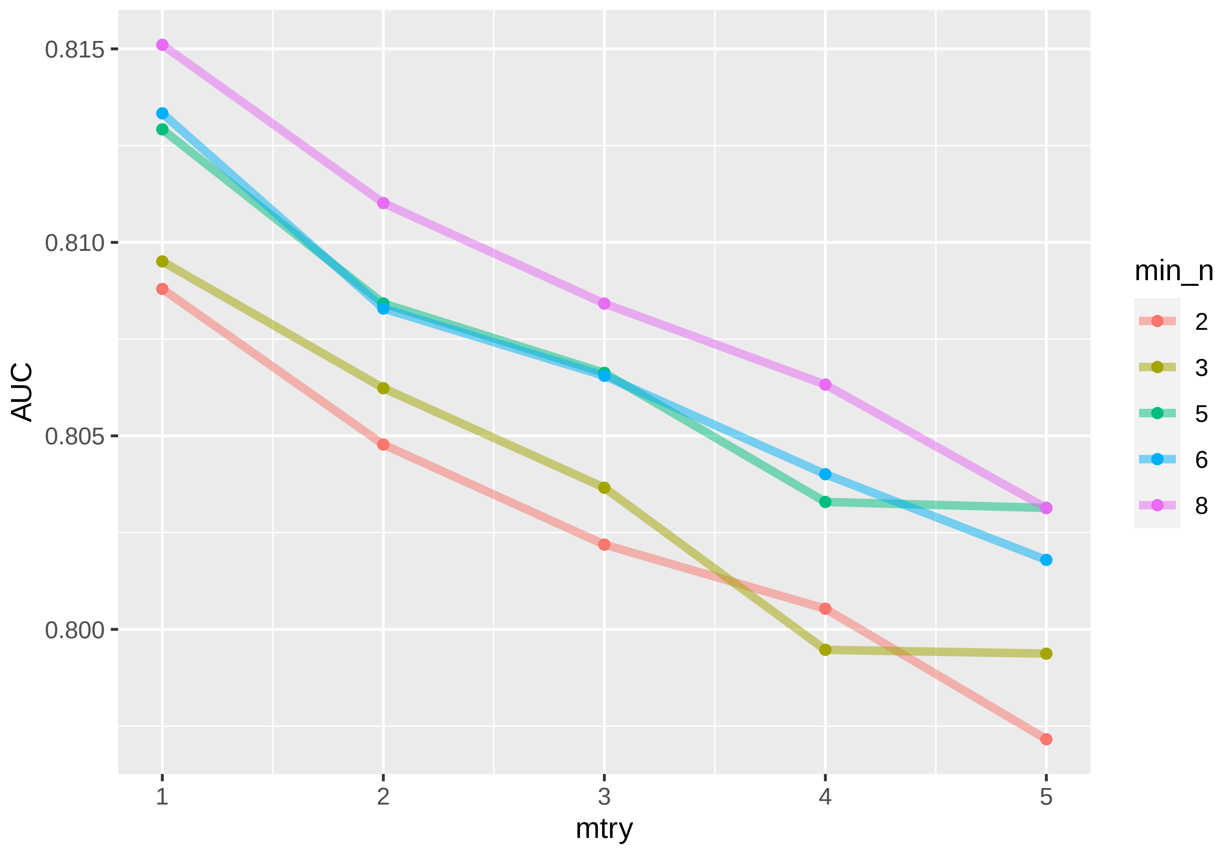
```
## # A tibble: 25 x 2
##     mtry min_n
##    <int> <int>
##  1     1     2
##  2     2     2
##  3     3     2
##  4     4     2
##  5     5     2
##  6     1     3
##  7     2     3
##  8     3     3
##  9     4     3
## 10     5     3
## # ... with 15 more rows
```

```r
regular_res <- tune_grid(
  tune_wf,
  resamples = trees_folds,
  grid = rf_grid
)

regular_res
```

```
## # Tuning results
## # 5-fold cross-validation
## # A tibble: 5 x 4
##   splits             id    .metrics          .notes
##   <list>             <chr> <list>            <list>
## 1 <split [3600/900]> Fold1 <tibble [50 x 6]> <tibble [0 x 3]>
## 2 <split [3600/900]> Fold2 <tibble [50 x 6]> <tibble [0 x 3]>
## 3 <split [3600/900]> Fold3 <tibble [50 x 6]> <tibble [0 x 3]>
## 4 <split [3600/900]> Fold4 <tibble [50 x 6]> <tibble [0 x 3]>
## 5 <split [3600/900]> Fold5 <tibble [50 x 6]> <tibble [0 x 3]>
```

```r
regular_res %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  mutate(min_n = factor(min_n)) %>%
  ggplot(aes(mtry, mean, color = min_n)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "AUC")
```




```
best_auc <- select_best(regular_res, "roc_auc")

final_rf <- finalize_model(
  tune_spec,
  best_auc
)

final_rf
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 1
##   trees = 100
##   min_n = 8
##
## Computational engine: ranger
```

```
rf_best <- rand_forest(
  mtry = 1,
  trees = 100,
  min_n = 8
) %>%
  set_mode("classification") %>%
  set_engine("ranger")

wflow_rf <- workflow() %>%
```

```
  add_model(rf_best) %>%
  add_recipe(data_rec)

rf_fit <- fit(wflow_rf, data_juiced)

rf_pred <- predict(rf_fit, new_data = data_bake)

tab_rf = table(data_bake$playlist_genre,rf_pred$.pred_class,dnn = c("obs","pred"))
```

```
# Model Evaluation
# Comparison
tab_lda;tab_knn;tab_rf
```

```
##        pred
## obs     pop rap rock latin r&b edm
##   pop   104  21   21    47   7  56
##   rap    22 113    6    46   9  34
##   rock   28   4  158     4   7  44
##   latin  53  36   15    91  19  54
##   r&b    50  47   46    43  39  27
##   edm    55  23    2    53   4 112

##        pred
## obs     pop rap rock latin r&b edm
##   pop    84  23   32    33  27  57
##   rap    24 101   10    35  30  30
##   rock   24   8  168     8  12  25
##   latin  45  34    9   104  38  38
##   r&b    39  52   40    38  59  24
##   edm    27   8    8    23   9 174

##        pred
## obs     pop rap rock latin r&b edm
##   pop    98  24   37    27  28  42
##   rap    15 128    9    33  23  22
##   rock   25  10  182     6   9  13
##   latin  56  42   18   101  22  29
##   r&b    38  42   41    37  69  25
##   edm    29  11   11    21   9 168
```

```
# Misclassification rates
(1500 - sum(diag(tab_lda)))/1500
```

```
## [1] 0.589
```

```
(1500 - sum(diag(tab_knn)))/1500
```

```
## [1] 0.54
```

```
(1500 - sum(diag(tab_rf)))/1500
```

```
## [1] 0.503
```

```
library(mltest)
per_lda <- ml_test(true = data_bake$playlist_genre,predicted =
                     pred_data_lda$predicted,output.as.table = FALSE)
per_knn <- ml_test(true = data_bake$playlist_genre,predicted =
                     knn_pred$.pred_class,output.as.table = FALSE)
```

```r
per_rf <- ml_test(true = data_bake$playlist_genre,predicted =
                    rf_pred$.pred_class,output.as.table = FALSE)
```

```r
Tab_per = data.frame("LDA" = cbind(per_lda$specificity,per_lda$FPR,per_lda$FNR),"KNN" = cbind(per_knn$sp
names(Tab_per) = c("LDA_spec",
                   "LDA_FPR","LDA_FNR","KNN_spec","KNN_FPR",
                   "KNN_FNR","RF_spec","RF_FPR","RF_FNR")
avg = apply(Tab_per,2,mean)
Tab_per = rbind(Tab_per,"average" = avg)

knitr::kable(Tab_per)
```

	LDA_spec	LDA_FPR	LDA_FNR	KNN_spec	KNN_FPR	KNN_FNR	RF_spec	RF_FPR	RF_FNR
pop	0.712	0.288	0.594	0.792	0.208	0.672	0.799	0.201	0.617
rap	0.794	0.206	0.509	0.825	0.175	0.561	0.827	0.173	0.443
rock	0.836	0.164	0.355	0.841	0.159	0.314	0.829	0.171	0.257
latin	0.732	0.268	0.660	0.811	0.189	0.612	0.839	0.161	0.623
r&b	0.926	0.074	0.845	0.845	0.155	0.766	0.882	0.118	0.726
edm	0.701	0.299	0.550	0.748	0.252	0.301	0.815	0.185	0.325
average	0.783	0.217	0.586	0.810	0.190	0.538	0.832	0.168	0.499