Spotify-Price-Prediction

Shrey Jaradi

2022-11-28

Predict the year of the song based on different characteristics like album cover etc. (from dataset -1)

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ggplot2)
library(naniar)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8     v purrr   0.3.4
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(corrr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:psych':
##
##     outlier
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(e1071)
library(class)
library(caTools)
print("Library Successfully loaded")
```

```
## [1] "Library Successfully loaded"
```

```
tracks_spotify_df = read.csv(file='../Dataset/dataset_one/tracks.csv')
```

```
head(tracks_spotify_df)
```

```
##                       id                                name popularity
## 1 35iwgR4jXetI318WEWsa1Q                               Carve          6
## 2 021ht4sdgPcrDgSk7JTbKY Capítulo 2.16 - Banquero Anarquista          0
## 3 07A5yehtSnoedViJAZkNnc  Vivo para Quererte - Remasterizado          0
## 4 08FmqUhxtyLTn6pAh6bk45       El Prisionero - Remasterizado          0
## 5 08y9GfoqCWfOGsKdwojr5e                 Lady of the Evening          0
## 6 0BRXJHRNGQ3W4v9frnSfhu                           Ave Maria          0
##   duration_ms explicit             artists                 id_artists
## 1      126903        0             ['Uli'] ['45tIt06XoI0Iio4LBEVpls']
## 2       98200        0 ['Fernando Pessoa'] ['14jtPCOoNZwquk5wd9DxrY']
## 3      181640        0 ['Ignacio Corsini'] ['5LiOoJbxVSAMkBS2fUm3X2']
## 4      176907        0 ['Ignacio Corsini'] ['5LiOoJbxVSAMkBS2fUm3X2']
## 5      163080        0     ['Dick Haymes'] ['3BiJGZsyX9sJchTqcSA7Su']
## 6      178933        0     ['Dick Haymes'] ['3BiJGZsyX9sJchTqcSA7Su']
##   release_date danceability energy key loudness mode speechiness acousticness
## 1   1922-02-22        0.645 0.4450   0  -13.338    1      0.4510        0.674
## 2   1922-06-01        0.695 0.2630   0  -22.136    1      0.9570        0.797
```

```
## 3   1922-03-21        0.434 0.1770   1  -21.180    1      0.0512        0.994
## 4   1922-03-21        0.321 0.0946   7  -27.961    1      0.0504        0.995
## 5         1922        0.402 0.1580   3  -16.900    0      0.0390        0.989
## 6         1922        0.227 0.2610   5  -12.343    1      0.0382        0.994
##   instrumentalness liveness valence   tempo time_signature
## 1           0.7440   0.1510  0.1270 104.851              3
## 2           0.0000   0.1480  0.6550 102.009              1
## 3           0.0218   0.2120  0.4570 130.418              5
## 4           0.9180   0.1040  0.3970 169.980              3
## 5           0.1300   0.3110  0.1960 103.220              4
## 6           0.2470   0.0977  0.0539 118.891              4
```

```
dim(tracks_spotify_df)
```

```
## [1] 586672     20
```

```
tracks_spotify_df = tracks_spotify_df[1:50000,]
```

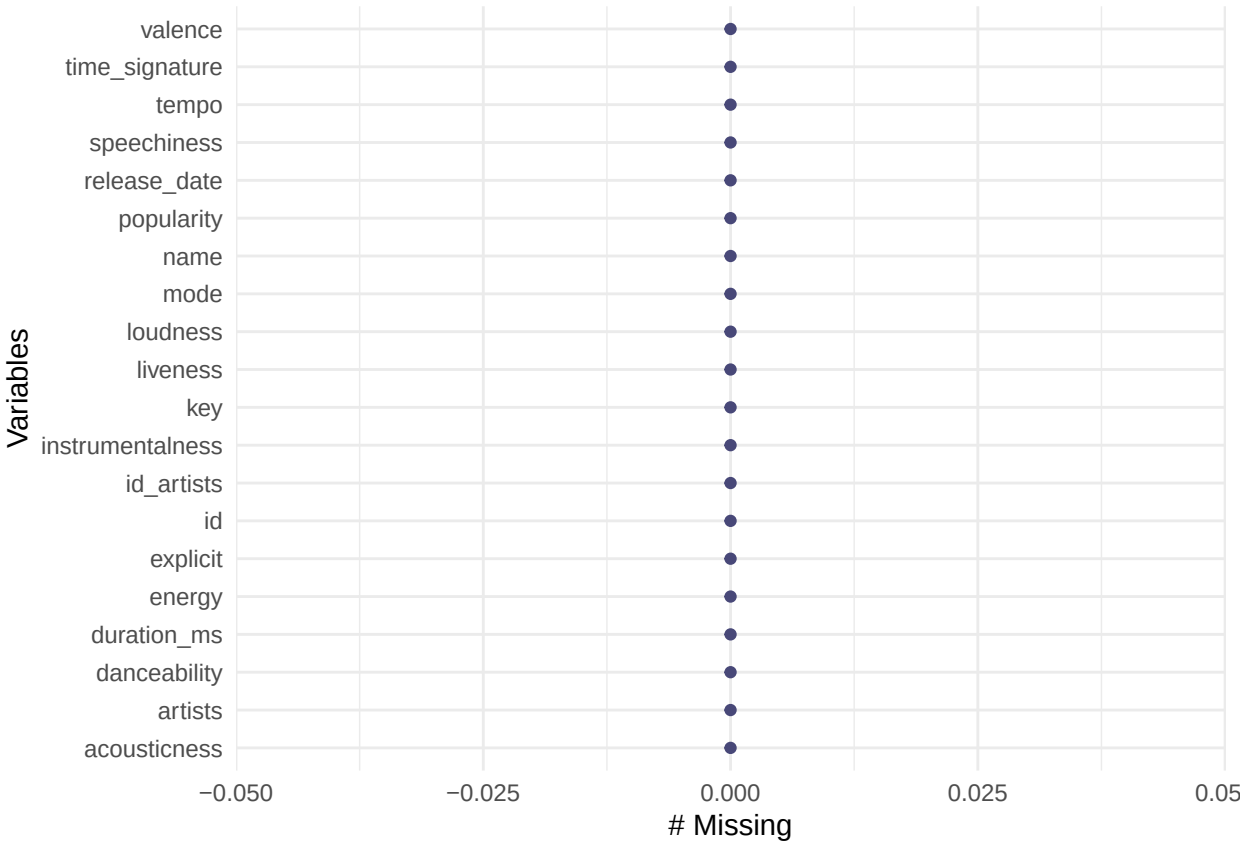
```
summary.default(tracks_spotify_df)
```

```
##                  Length Class  Mode
## id               50000  -none- character
## name             50000  -none- character
## popularity       50000  -none- numeric
## duration_ms      50000  -none- numeric
## explicit         50000  -none- numeric
## artists          50000  -none- character
## id_artists       50000  -none- character
## release_date     50000  -none- character
## danceability     50000  -none- numeric
## energy           50000  -none- numeric
## key              50000  -none- numeric
## loudness         50000  -none- numeric
## mode             50000  -none- numeric
## speechiness      50000  -none- numeric
## acousticness     50000  -none- numeric
## instrumentalness 50000  -none- numeric
## liveness         50000  -none- numeric
## valence          50000  -none- numeric
## tempo            50000  -none- numeric
## time_signature   50000  -none- numeric
```

Missing values

```
gg_miss_var(tracks_spotify_df)
```

```
## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.
```



Checking for Missing Values

```
tracks_spotify_df %>% summarise_all(~ sum(is.na(.)))
```

```
##   id name popularity duration_ms explicit artists id_artists release_date
## 1  0    0          0           0        0       0          0            0
##   danceability energy key loudness mode speechiness acousticness
## 1            0      0   0        0    0           0            0
##   instrumentalness liveness valence tempo time_signature
## 1                0        0       0     0              0
```

Creating new column for year from release_date column and converting the column datatype to datetime

```
tracks_spotify_df$release_year = substring(tracks_spotify_df$release_date,1,4)
tracks_spotify_df$release_year = as.integer(tracks_spotify_df$release_year)
```

```
unique(tracks_spotify_df$release_year)
```

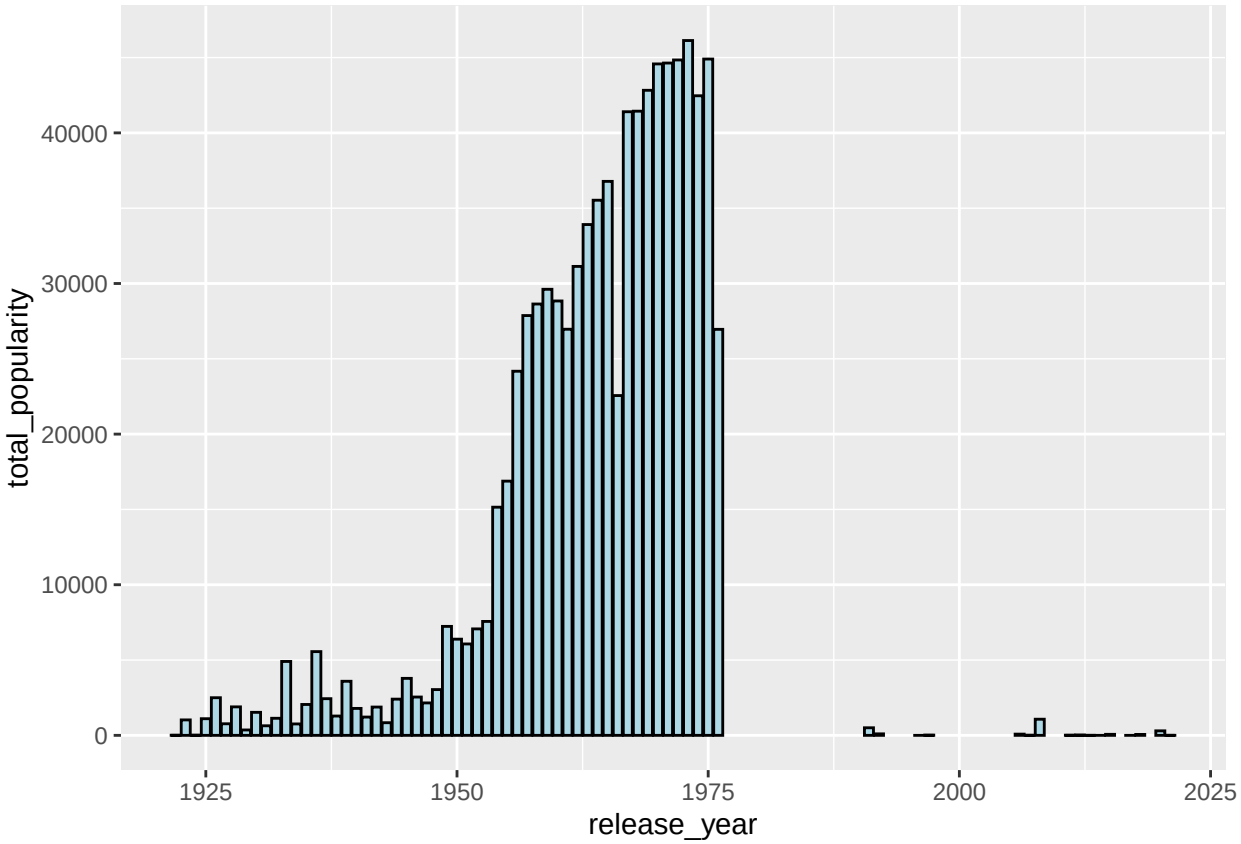
```
##  [1] 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936
## [16] 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951
## [31] 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966
## [46] 1968 2008 2020 2018 1997 2006 1991 2012 2015 2011 1992 2007 1996 2021 2013
## [61] 2014 2017 1967 1969 1970 1971 1972 1973 1974 1975 1976
```

group by year, and check for the popularity for each year

```
popularity_by_year = tracks_spotify_df %>% group_by(release_year) %>% dplyr::summarize(total_popularity=
```

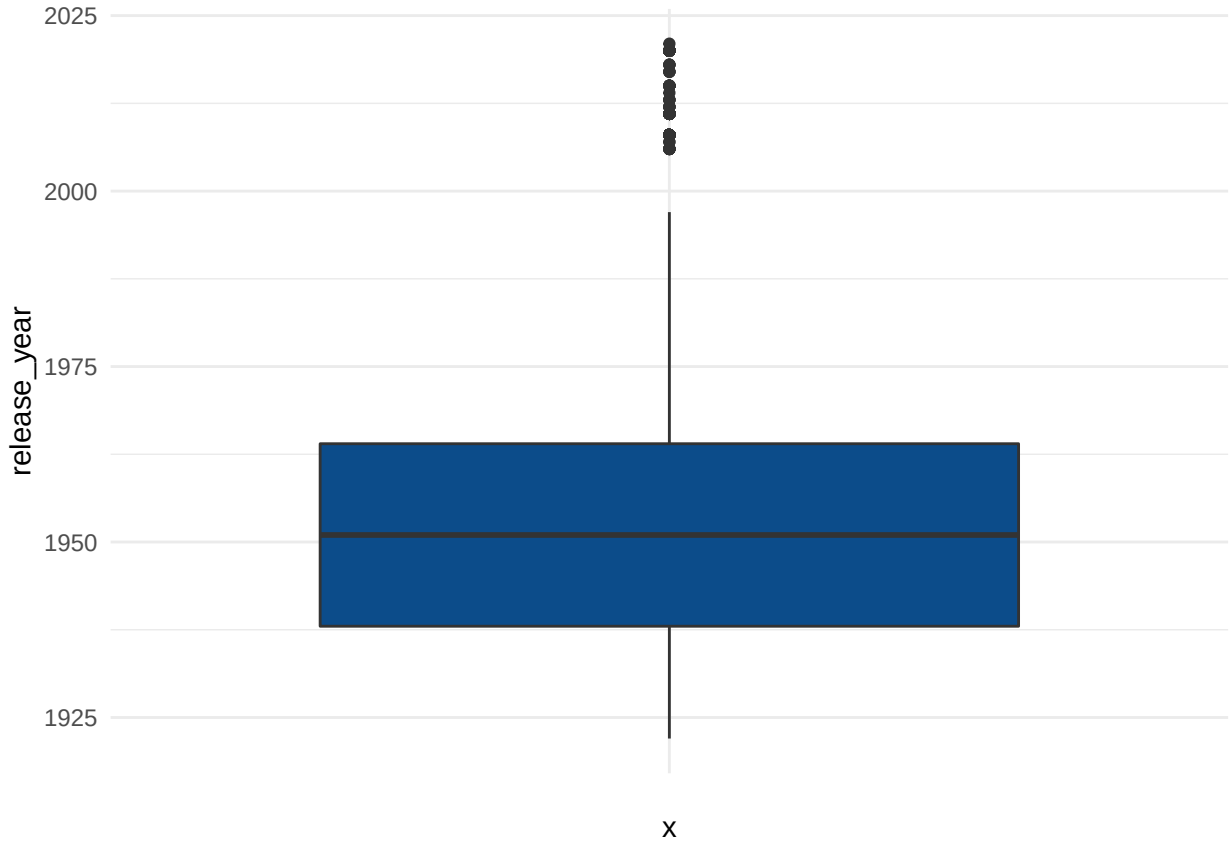
Popularity by year

```
ggplot(popularity_by_year, aes(x = release_year, y = total_popularity)) + geom_bar(position = "dodge", s
```



check for the outlier

```
ggplot(tracks_spotify_df) +aes(x = "", y = release_year) +geom_boxplot(fill = "#0c4c8a") + theme_minimal
```



Finding out the outlier in the release_year, if there are any and removing !!

```
Quant = quantile(tracks_spotify_df$release_year, probs=c(.25, .75), na.rm = T)
Quant
```

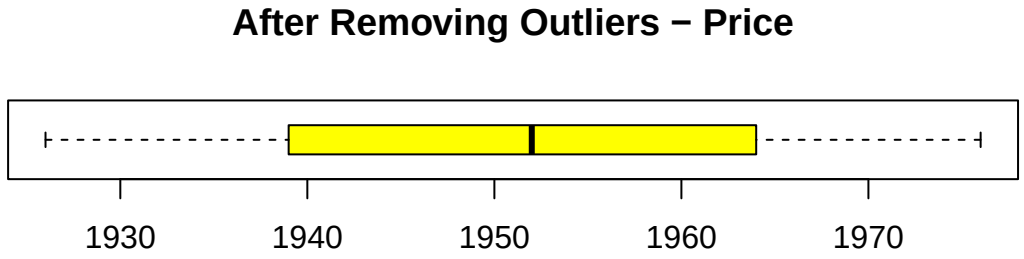
```
##  25%  75%
## 1938 1964
```

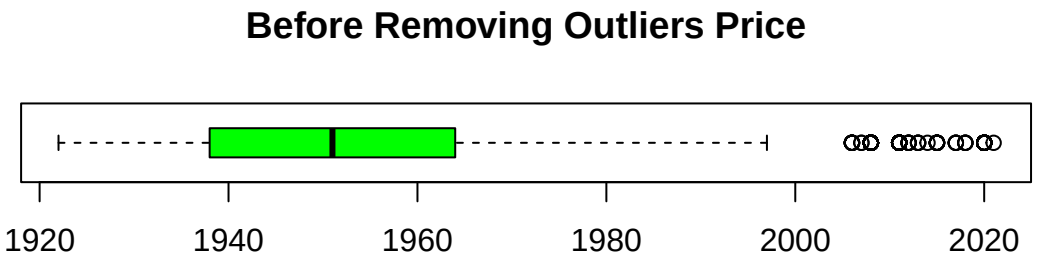
```
iqr_val = IQR(tracks_spotify_df$release_year, na.rm = T)
iqr_val
```

```
## [1] 26
```

```
tracks_spotify_df_new = tracks_spotify_df %>% filter(release_year > (Quant[1] - 0.5*iqr_val) & release_y
```

```
par(mfrow=c(2,1))
options(repr.plot.width=12, repr.plot.height=6)
boxplot(tracks_spotify_df_new$release_year, col = "yellow", horizontal = T, main = "After Removing Outli
boxplot(tracks_spotify_df$release_year, col = "green", horizontal = T, main = "Before Removing Outliers
```





Excluding unique id variable that aren't needed for our problem so, and aren;t much important.

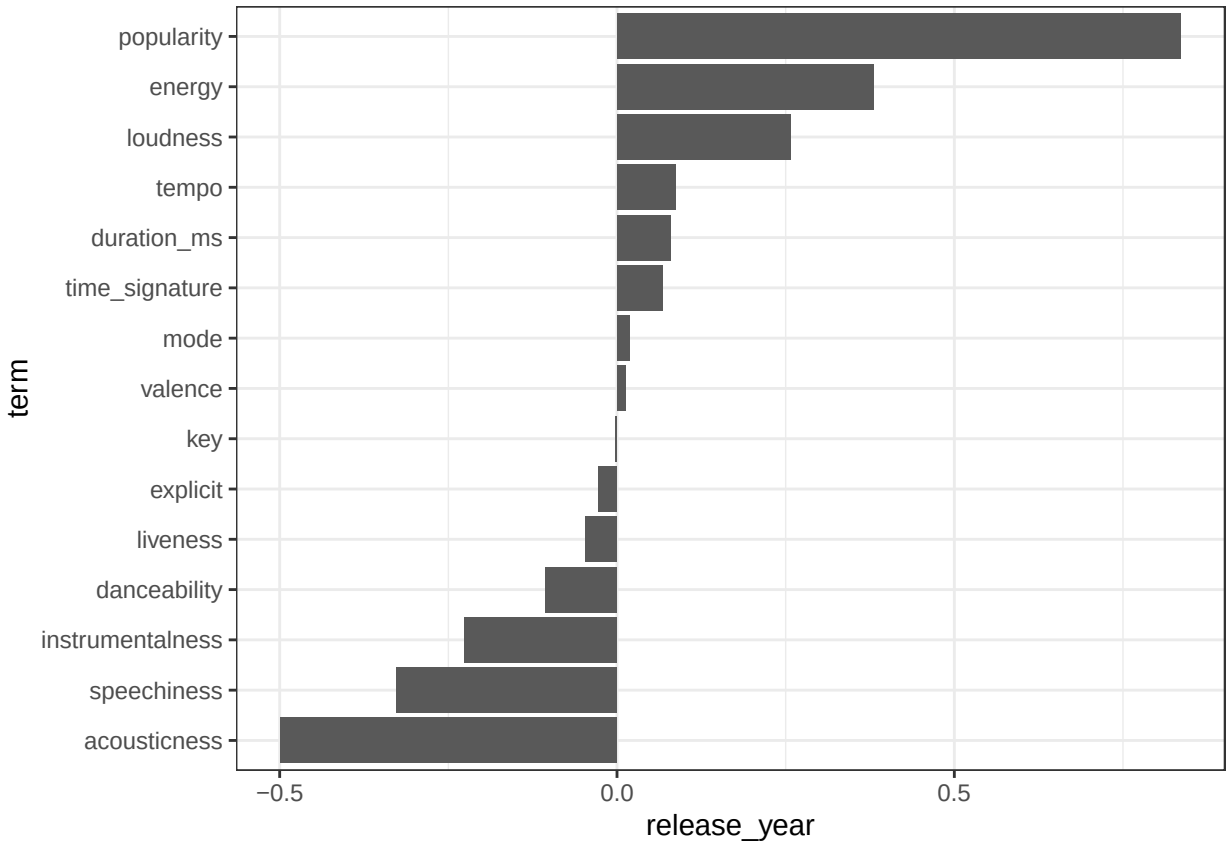
```
tracks_spotify_df_filter = tracks_spotify_df_new[,c("popularity", "duration_ms","explicit", "danceabil
```

Correlation of each data focused on release_year as that will be our response variable

```
corr_mat = correlate(tracks_spotify_df_filter[,c("popularity", "duration_ms","explicit", "danceability"
```

```
## Correlation computed with
## * Method: 'pearson'
## * Missing treated using: 'pairwise.complete.obs'
```

```
corr_mat %>%focus(release_year) %>% mutate(term = reorder(term, release_year)) %>% ggplot(aes(term,relea
```



Splitting into train and test dataset, taking on 1lakh records because of the memory issue

```
track_spotify_filter_index = createDataPartition(tracks_spotify_df_filter$release_year, p=.70, list=FALS
track_spotify_filter_train = tracks_spotify_df_filter[track_spotify_filter_index,]
track_spotify_filter_test =  tracks_spotify_df_filter[-track_spotify_filter_index,]
```

###Multiple Linear Regression model To predict the price, to check if we can predict using the regression classifier

```
lm_model = lm(release_year ~ popularity +  explicit  + danceability  + energy + loudness  + acousticnes
summary(lm_model)
```

```
##
## Call:
## lm(formula = release_year ~ popularity + explicit + danceability +
##     energy + loudness + acousticness + instrumentalness, data = track_spotify_filter_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.296  -5.333   1.336   5.971  29.150
##
## Coefficients:
##                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)       1.949e+03  3.995e-01 4878.801  < 2e-16 ***
## popularity        5.864e-01  2.661e-03  220.350  < 2e-16 ***
## explicit         -6.456e+00  1.040e+00   -6.210 5.37e-10 ***
## danceability     -1.178e+01  2.610e-01  -45.131  < 2e-16 ***
## energy            3.241e+00  3.465e-01    9.355  < 2e-16 ***
```

```
## loudness          4.353e-02  1.195e-02    3.643  0.00027 ***
## acousticness     -3.690e+00  2.149e-01  -17.176  < 2e-16 ***
## instrumentalness -1.811e-01  1.312e-01   -1.380  0.16752
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.674 on 33366 degrees of freedom
## Multiple R-squared:  0.7195, Adjusted R-squared:  0.7194
## F-statistic: 1.223e+04 on 7 and 33366 DF,  p-value: < 2.2e-16
```

```
rf_model = randomForest(release_year ~ popularity +  explicit  + danceability  + energy + loudness  + a
summary(rf_model)
```

Random Forest Model to predict the year using different predictor variable

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted      33374   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times      33374   -none- numeric
## importance         7   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y              33374   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

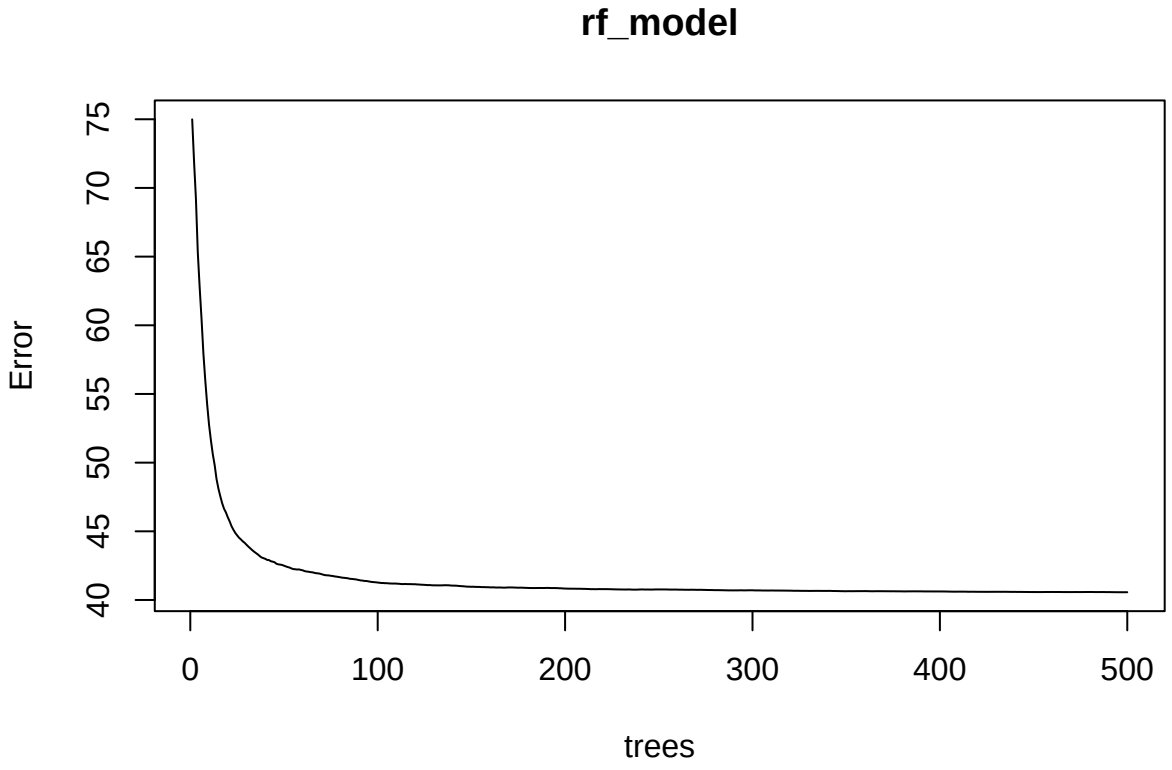
```
predicted_val = predict(rf_model, track_spotify_filter_test[,-16])
predicted_val = ceiling(predicted_val)
```

```
postResample(predicted_val, track_spotify_filter_test$release_year)
```

```
##      RMSE  Rsquared       MAE
## 6.3069904 0.8107416 4.8173554
```

Model Performance

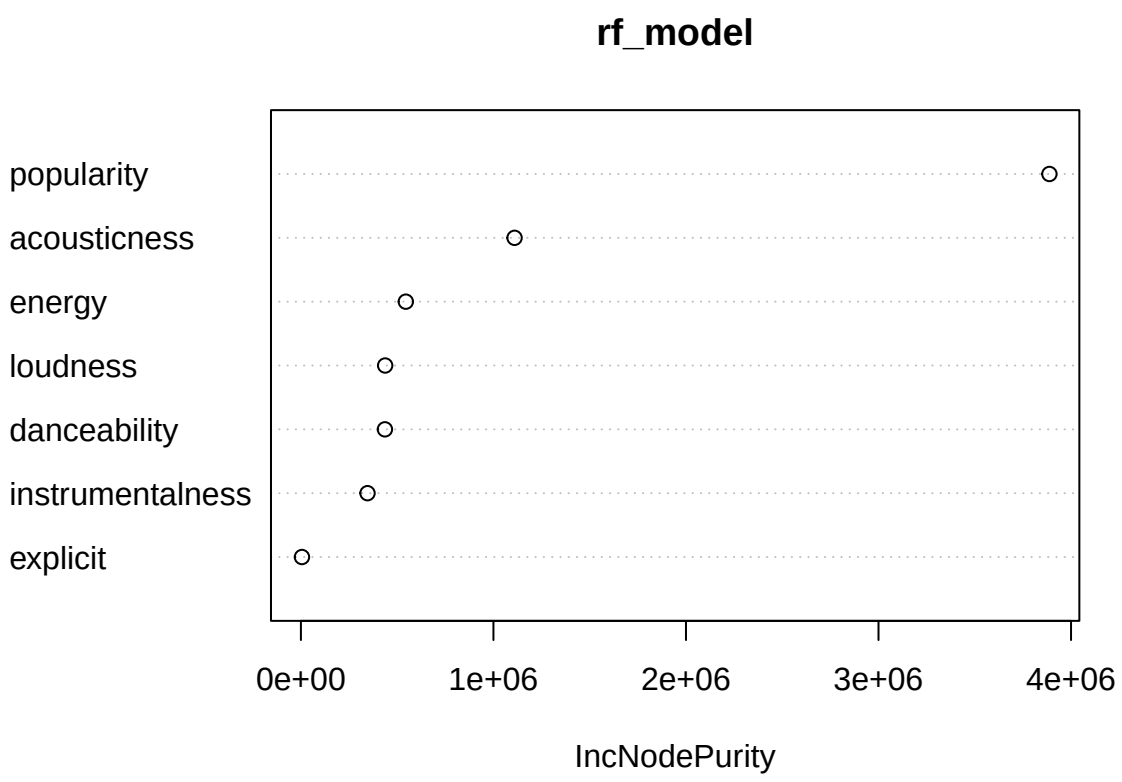
```
plot(rf_model)
```



Impor-

tant variables plot

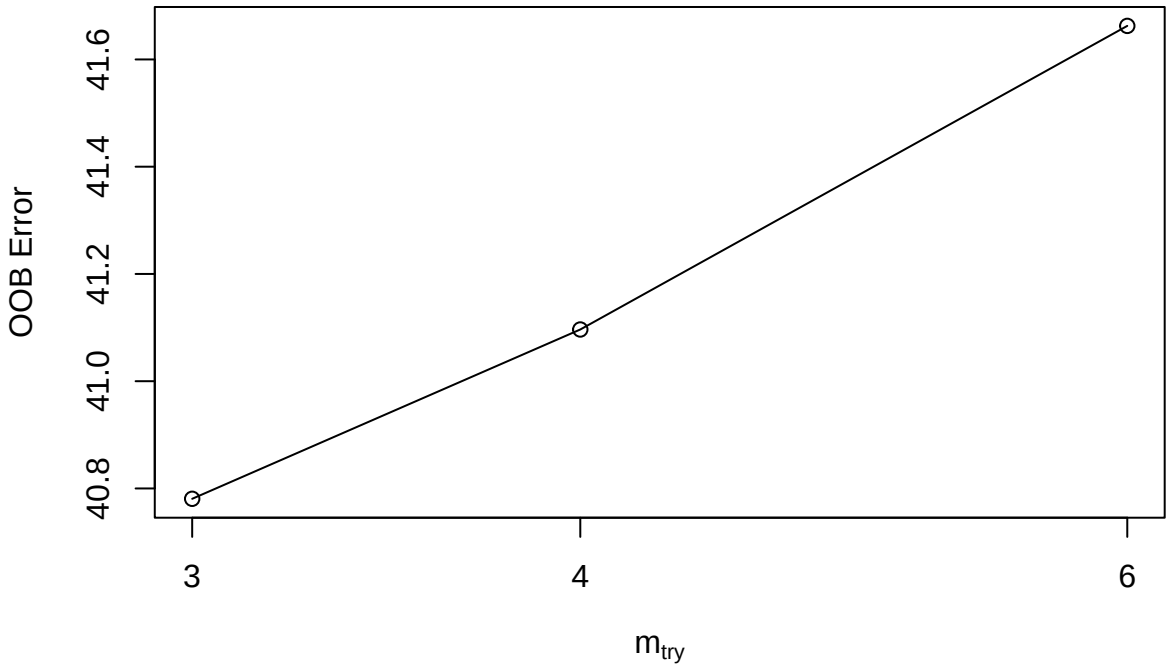
```
varImpPlot(rf_model)
```



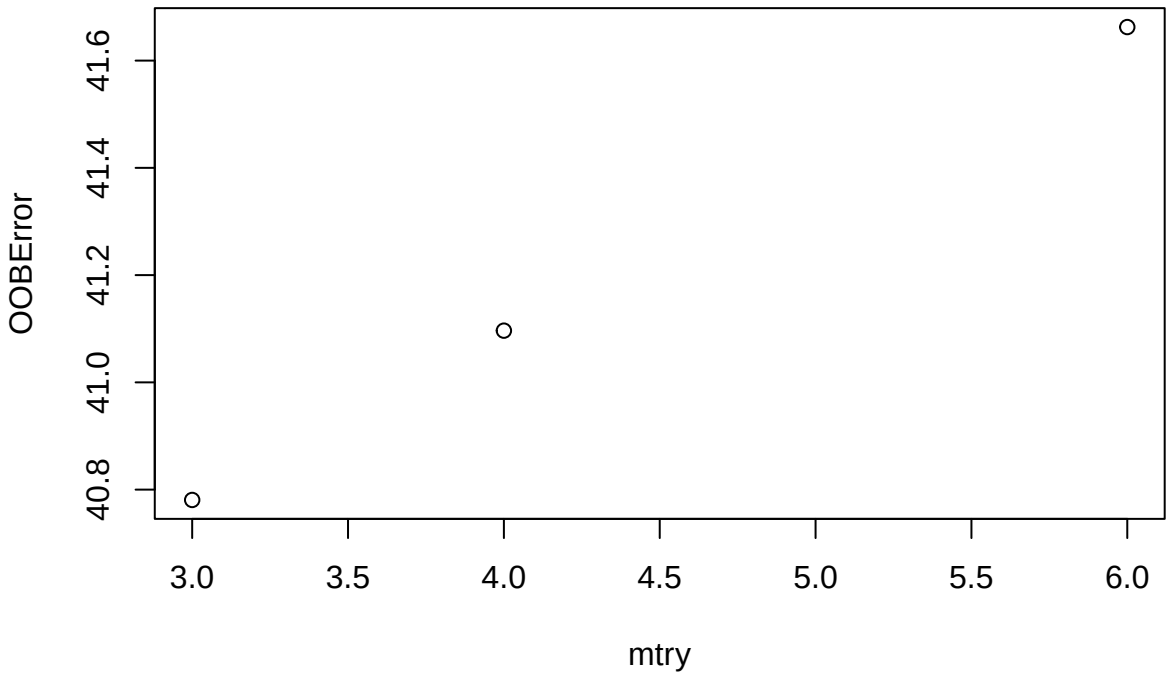
Tunning the Random Forest Model

```
model_tuned = tuneRF(x=track_spotify_filter_train[,c("popularity", "explicit","danceability", "energy",
```

```
## 0.007682417 0.01
## -0.01377554 0.01
```




```
plot(model_tuned)
```




```
predicted_tuned_val = predict(rf_model, newdata=track_spotify_filter_test[,-16])
predicted_tuned_val = ceiling(predicted_tuned_val)
```

```
postResample(predicted_val, track_spotify_filter_test$release_year)
```

```
##      RMSE  Rsquared       MAE
## 6.3069904 0.8107416 4.8173554
```

The answer to our research question, to predict the release_year of the song from it's feature. Yes, as we took some of the correlated features and applied linear regression , Random Forest classifier to predict the year. And our Random Forest classifier shows good R-square value. Overall it looks good as of now. Hopefully we can predict the year from other features in our dataset like "popularity", "explicit","danceability", "energy", "loudness", "acousticness", "instrumentalness".

###Which genre got famous/changed according to year and why? (from dataset -3)

---

https://www.kaggle.com/code/akiboy96/spotify-song-popularity-genre-exploration/data?select=genre_music.csv

```
genre_spotify_df = read.csv(file='../Dataset/dataset_three/genre_music.csv')
head(genre_spotify_df)
```

```
##                   track           artist danceability energy key loudness mode
## 1 Jealous Kind Of Fella    Garland Green        0.417  0.620   3   -7.727    1
## 2         Initials B.B. Serge Gainsbourg        0.498  0.505   3  -12.475    1
## 3          Melody Twist      Lord Melody        0.657  0.649   5  -13.392    1
## 4         Mi Bomba Sonó       Celia Cruz        0.590  0.545   7  -12.058    0
## 5           Uravu Solla      P. Susheela        0.515  0.765  11   -3.515    0
## 6             Beat n. 3  Ennio Morricone        0.697  0.673   0  -10.573    1
##   speechiness acousticness instrumentalness liveness valence   tempo duration_s
## 1      0.0403        0.490         0.00e+00   0.0779   0.845 185.655    173.533
## 2      0.0337        0.018         1.07e-01   0.1760   0.797 101.801    213.613
## 3      0.0380        0.846         4.42e-06   0.1190   0.908 115.940    223.960
## 4      0.1040        0.706         2.46e-02   0.0610   0.967 105.592    157.907
## 5      0.1240        0.857         8.72e-04   0.2130   0.906 114.617    245.600
## 6      0.0266        0.714         9.19e-01   0.1220   0.778 112.117    167.667
##   time_signature chorus_hit sections popularity decade genre
## 1              3   32.94975        9          1    60s   edm
## 2              4   48.82510       10          0    60s   pop
## 3              4   37.22663       12          0    60s   pop
## 4              4   24.75484        8          0    60s   pop
## 5              4   21.79874       14          0    60s   r&b
## 6              4   65.48604        7          0    60s   pop
```

```
dim(genre_spotify_df)
```

```
## [1] 41099    20
```

```
summary.default(genre_spotify_df)
```

```
##                  Length Class  Mode
## track            41099  -none- character
## artist           41099  -none- character
## danceability     41099  -none- numeric
## energy           41099  -none- numeric
## key              41099  -none- numeric
## loudness         41099  -none- numeric
## mode             41099  -none- numeric
## speechiness      41099  -none- numeric
## acousticness     41099  -none- numeric
## instrumentalness 41099  -none- numeric
```

```
## liveness         41099  -none- numeric
## valence          41099  -none- numeric
## tempo            41099  -none- numeric
## duration_s       41099  -none- numeric
## time_signature   41099  -none- numeric
## chorus_hit       41099  -none- numeric
## sections         41099  -none- numeric
## popularity       41099  -none- numeric
## decade           41099  -none- character
## genre            41099  -none- character
```

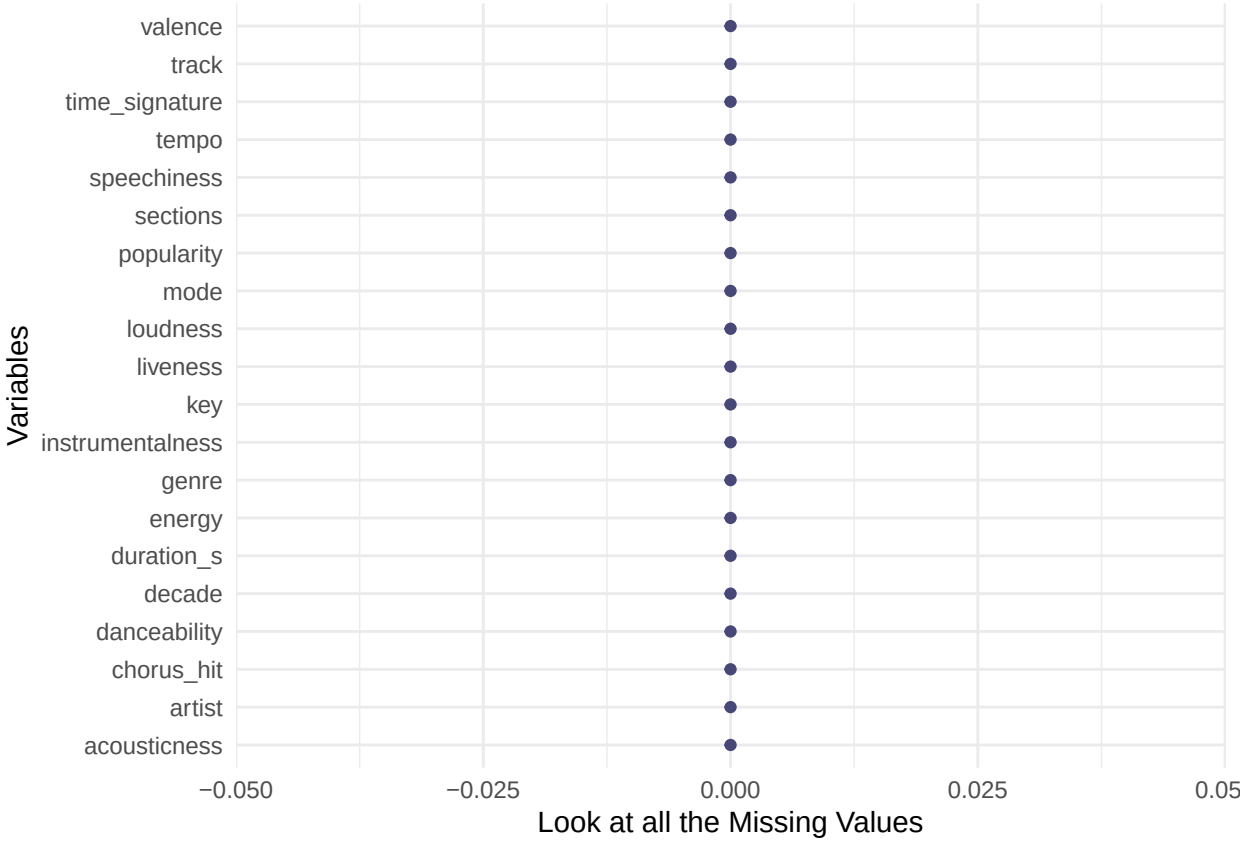
```
colnames(genre_spotify_df)
```

```
##  [1] "track"            "artist"           "danceability"     "energy"
##  [5] "key"              "loudness"         "mode"             "speechiness"
##  [9] "acousticness"     "instrumentalness" "liveness"         "valence"
## [13] "tempo"            "duration_s"       "time_signature"   "chorus_hit"
## [17] "sections"         "popularity"       "decade"           "genre"
```

check for missing values

```
naniar::gg_miss_var(genre_spotify_df) +
  theme_minimal()+
  labs(y = "Look at all the Missing Values")
```

```
## Warning: It is deprecated to specify `guide = FALSE` to remove a guide. Please
## use `guide = "none"` instead.
```



Find unique values in genre , popularity, decade

```
unique(genre_spotify_df$genre)
```

```
## [1] "edm"   "pop"   "r&b"   "rock"  "rap"   "latin"
```

```
unique(genre_spotify_df$popularity)
```

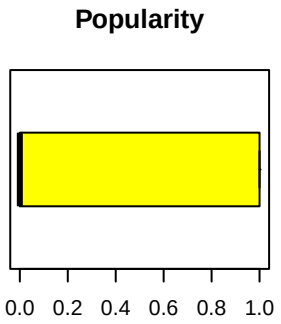
```
## [1] 1 0
```

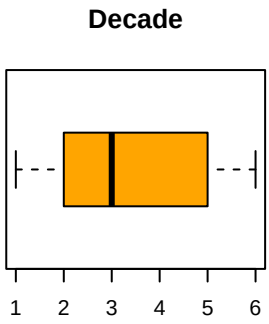
```
unique(genre_spotify_df$decade)
```

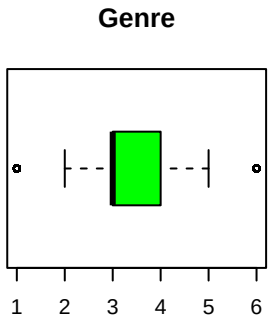
```
## [1] "60s" "70s" "80s" "90s" "00s" "10s"
```

Finding the outlier

```
par(mfcol=c(2,3))
boxplot(genre_spotify_df$popularity, col = "yellow", horizontal = T, main = "Popularity")
boxplot(as.factor(genre_spotify_df$genre), col = "green", horizontal = T, main = "Genre")
boxplot(as.factor(genre_spotify_df$decade), col = "orange", horizontal = T, main = "Decade")
```





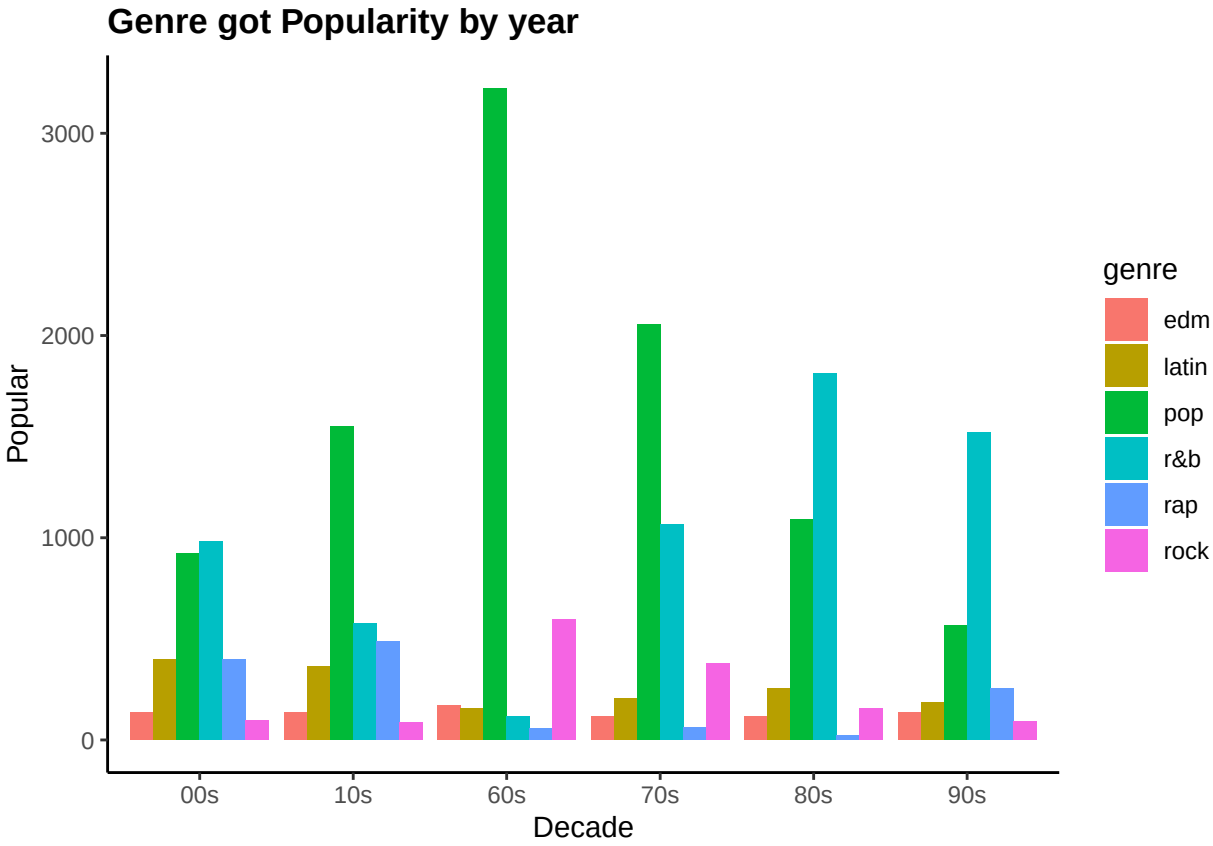


group the data with decade and genre

```
genre_grp = genre_spotify_df %>% group_by(decade, genre)  %>% dplyr::summarise(total_popularity = sum(p
```

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

```
ggplot(genre_grp, aes(x = decade, y = total_popularity, fill = genre )) +
  geom_bar(position = "dodge", stat = "identity") +
  theme_classic() +
  labs(title = "Genre got Popularity by year", x = "Decade", y = "Popular") +
  theme(plot.title = element_text(face = "bold"))
```



This plot answer our question of Popular genre by year This are the genre that were famous in each decade, - the most popular genre is pop in 1960s - in 2000s r&b genre was famous - in 2010s pop was famous - in 1970s again pop was famous - in 1980s r&b was famous - 1990s r&b was famous