PAC Report

Wenyu Chen

2022-11-29

Introduction

This report summarizes my main work for the PAC, including the data analysis process, what I think I did right and wrong, and what needs more improvement.

Data Analysis Process

1. Data Tidying

After importing the data, I did some tidying work first. I transformed the categorical variables from numeric or character type into factors. Then the null value was checked. The null value only exists in the column "genre", which needs to be further arranged. The processing of "genre" will be demonstrated in the later part.

```
#Read data
songs=read.csv("/Users/angelachen/Desktop/AA challenge/5200/Kaggle Project/lalasongs22/analysisData.csv
scoringdata=read.csv("/Users/angelachen/Desktop/AA challenge/5200/Kaggle Project/lalasongs22/scoringData
#Transform into correct format
library(stringr)
songs$performer=as.factor(str_to_title(songs$performer))
songs$mode=as.factor(songs$mode)
songs$key=as.factor(songs$key)
songs$track_explicit=as.factor(songs$track_explicit)
songs$time_signature=as.factor(songs$time_signature)
```

```
#Check for the null value
apply(songs,
      MARGIN = 2,
      FUN = function(x) 100*sum(is.na(x))/(sum(is.na(x))+ sum(!is.na(x))))
```

```
##                id        performer             song            genre
##         0.0000000        0.0000000        0.0000000        0.5542725
##    track_duration   track_explicit     danceability           energy
##         0.0000000        0.0000000        0.0000000        0.0000000
##               key         loudness             mode      speechiness
##         0.0000000        0.0000000        0.0000000        0.0000000
##      acousticness instrumentalness         liveness          valence
##         0.0000000        0.0000000        0.0000000        0.0000000
##             tempo   time_signature           rating
##         0.0000000        0.0000000        0.0000000
```

I did some descriptive and visual summary of the data, which is displayed below. The data includes numeric and categorical types, and there exists skewness in the distribution of variables, which is one of the reason that I choose to use random forest model.

```r
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
```

```r
library(tidyr)
library(ggplot2)
```

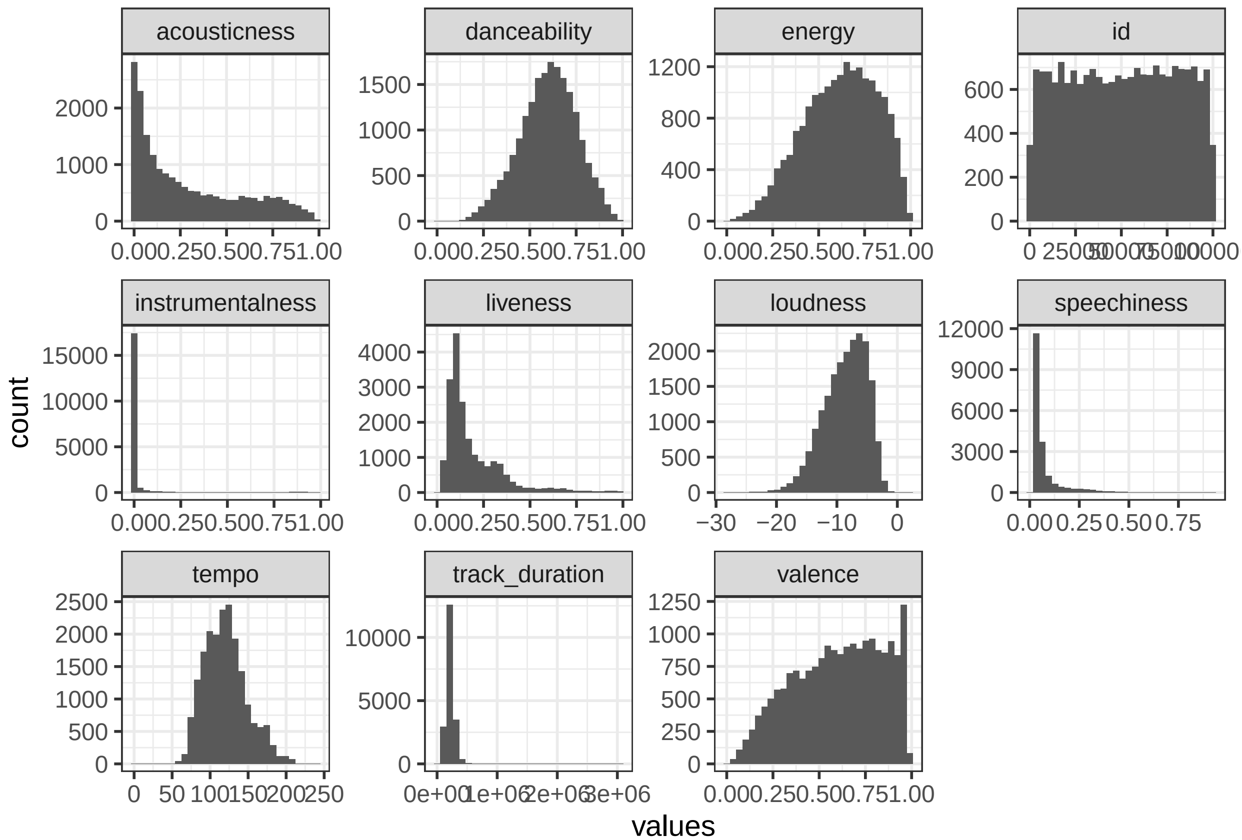
```r
summary(songs)
```

```
##        id                      performer         song
##  Min.   :    3   Glee Cast         :  147   Length:19485
##  1st Qu.:24812   Taylor Swift      :   98   Class :character
##  Median :50487   Drake             :   78   Mode  :character
##  Mean   :50209   The Beatles       :   52
##  3rd Qu.:75544   Aretha Franklin   :   45
##  Max.   :99999   The Rolling Stones:   45
##                  (Other)           :19020
##     genre           track_duration    track_explicit  danceability
##  Length:19485       Min.   :  29688   FALSE:17203    Min.   :0.0000
##  Class :character   1st Qu.: 175173   TRUE : 2282    1st Qu.:0.4990
##  Mode  :character   Median : 214733                  Median :0.6070
##                     Mean   : 220873                  Mean   :0.5994
##                     3rd Qu.: 253306                  3rd Qu.:0.7070
##                     Max.   :3079157                  Max.   :0.9880
##
##      energy              key          loudness        mode       speechiness
##  Min.   :0.000581   0      :2443   Min.   :-28.030   0: 5282   Min.   :0.0000
##  1st Qu.:0.475000   7      :2260   1st Qu.:-11.036   1:14203   1st Qu.:0.0321
##  Median :0.633000   2      :1989   Median : -8.206             Median :0.0412
##  Mean   :0.617599   9      :1965   Mean   : -8.673             Mean   :0.0732
##  3rd Qu.:0.777000   5      :1762   3rd Qu.: -5.856             3rd Qu.:0.0678
##  Max.   :0.997000   1      :1733   Max.   :  2.291             Max.   :0.9240
##                     (Other):7333
##   acousticness       instrumentalness       liveness         valence
##  Min.   :0.0000033   Min.   :0.0000000   Min.   :0.0130   Min.   :0.0000
##  1st Qu.:0.0465000   1st Qu.:0.0000000   1st Qu.:0.0907   1st Qu.:0.4140
##  Median :0.1940000   Median :0.0000046   Median :0.1310   Median :0.6210
##  Mean   :0.2942529   Mean   :0.0328609   Mean   :0.1928   Mean   :0.6011
##  3rd Qu.:0.5070000   3rd Qu.:0.0004570   3rd Qu.:0.2500   3rd Qu.:0.8020
##  Max.   :0.9910000   Max.   :0.9820000   Max.   :0.9990   Max.   :0.9910
##
##      tempo        time_signature     rating
##  Min.   :  0.00   0:    2        Min.   : 0.00
##  1st Qu.: 99.08   1:   71        1st Qu.:24.00
##  Median :119.00   3: 1243        Median :36.00
##  Mean   :120.24   4:18017        Mean   :36.69
##  3rd Qu.:136.39   5:  152        3rd Qu.:50.00
```

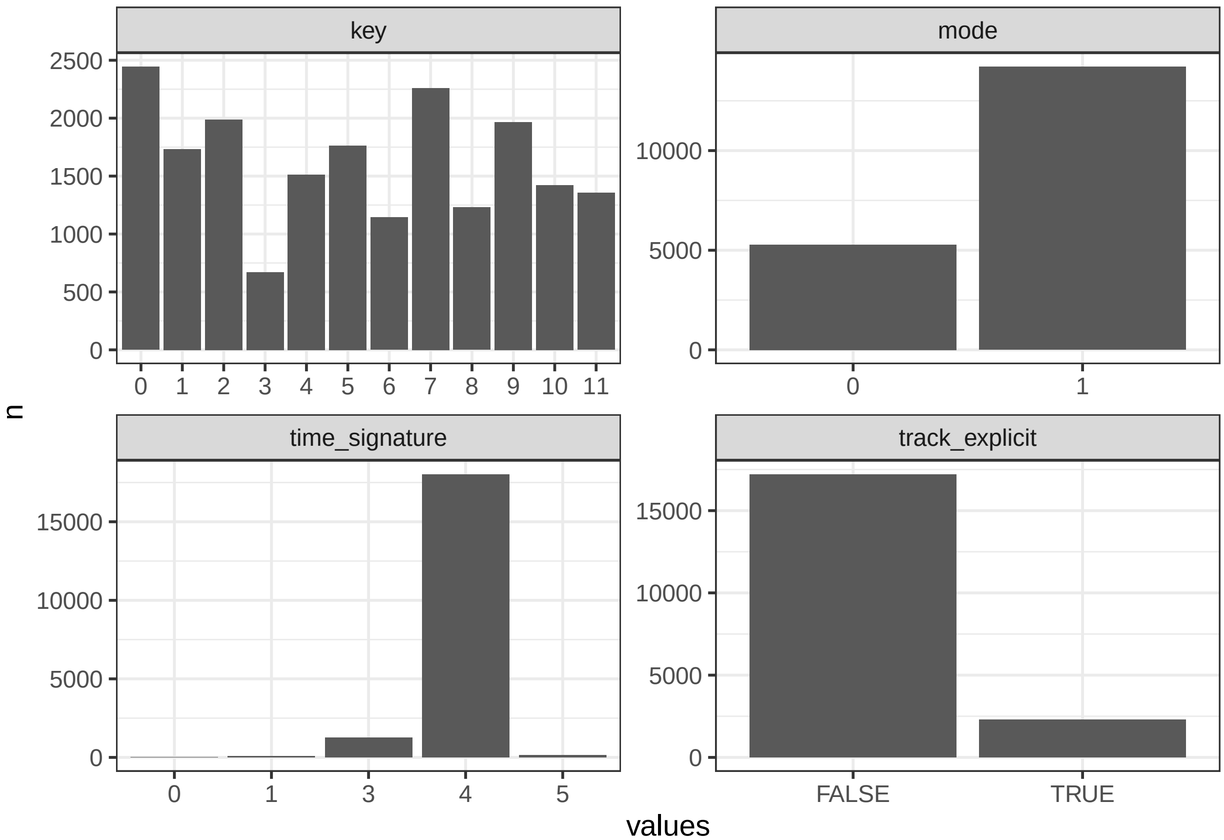
```
##   Max.    :241.01                  Max.    :91.00
##
```

```
songs %>%
  select(-rating)%>%
  select_if(is.numeric)%>%
  pivot_longer(cols = 1:11,names_to = 'numeric_predictor', values_to = 'values'  )%>%
  ggplot(aes(x = values))+
  geom_histogram()+
  facet_wrap(numeric_predictor~., scales = 'free')+
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
songs %>%
  select(-performer)%>%
  select_if(is.factor)%>%
  pivot_longer(cols = 1:4,names_to = 'categorical_predictor', values_to = 'values'  )%>%
  group_by(categorical_predictor, values)%>%
  count()%>%
  ungroup()%>%
  ggplot(aes(x = values, y = n))+
  geom_col()+
  facet_wrap(categorical_predictor~., scales = 'free')+
  theme_bw()
```



2. Exploring More Possibilities of Predictors

- Sort "Speechiness" and "Instrumentalness"

Songs can be classified into different types based on the value of variables "speechiness" and "instrumentalness". Songs with speechiness value larger than 0.66 are probably made entirely of spoken words, and with value less than 0.33 are most likely represent music and other non-speech-like tracks. So I created a new variable named "speechiness_categorical" of 3 categories: nonspeech, mix, and speech. Songs with instrumentalness value larger than 0.5 are regarded as instrumental tracks, and with value less than 0.5 are vocal tracks. The new variable "instrumentalness_categorical" has two categories: instrumental and vocal.

However, my final model still includes the original variables "speechiness" and "instrumentalness", rather than the new-created categorical variables. The possible reason is that treating them as categorical predictors do lose some information. It will be mentioned in the later part.

```
songs$speechiness_categorical=as.factor(ifelse(songs$speechiness<0.33, "nonspeech", ifelse(songs$speechi
songs$instrumentalness_categorical=as.factor(ifelse(songs$instrumentalness>0.5, "instrumental","vocal"))
```

- Length of Song Titles

There may be a relationship between the songs' popularity and length of their titles. The shorter length is beneficial to making songs easy to remember, and to public communication. Rather, the longer length could present listeners a sense of novelty and originality. So it's possible that the length of song titles is a crucial predictors. I created a new independent variable "title_length" by calculating the number of words in each song's title.

```
songs$title_length=str_count(songs$song, boundary("word"))
```

- Collaboration of performers

Collaboration of performers brings more possibilities to songs, and is a form beloved by listeners. Sometimes, collaboration is highlight of songs. So I think a variable depicting whether there is collaboration would be a powerful predictor. Therefore, I created a predictor named "collaboration", based on whether there exists string like "Featuring" in the "performer" column.

```
songs$collaboration=ifelse((grepl("Featuring",songs$performer) |
                            grepl("&",songs$performer)|
                            grepl(",",songs$performer)|
                            grepl("Feat",songs$performer)|
                            grepl("With",songs$performer)), 1,0)
```

- Performers

To Deal with the variable "performer", I firstly splited the column to separate each performer's name, since many songs involve several performers. Honestly, the code in this part is untidy and time-consuming, which I need to improve if I could do the project over.

```
songs<-songs%>%
  separate(col=performer, into = c('p1','p2'), sep = "Featuring", extra = 'merge', fill = 'right')
songs<-songs%>%
  separate(col=p1, into = c('p3','p4','p5','p6','p7'), sep = ",", extra = 'merge', fill = 'right')
#......
#Some same steps are omitted
songs<-songs%>%
  separate(col=p12, into = c('p32','p33'), sep = "&", extra = 'merge', fill = 'right')
songs[,2:22]<-apply(songs[,2:22], MARGIN = 2, FUN = function(x) str_trim(x))
```

Then I calculate the average rating of all songs appearing in the analysis dataset that each performer participated in. And based on the average rating, the performers were equally divided into 3 categories: high, medium and low popularity. After that, I created a new predictors "performer_popularity". If one of the song's performers belong to the high popularity, performer_popularity='High', and so on. However, in the following modeling part, I found that "performer_popularity" didn't have a strong predictive power. Consequently, I went back to the original "performer" variable.

```
performerrating_songs=songs[, c(2:22, 40)]%>%
  pivot_longer(cols = 1:21, names_to = 'names', values_to = 'performer')
performerrating_songs<-performerrating_songs[!is.na(performerrating_songs$performer),c(1,3)]
performerrating_songs$performer=as.factor(performerrating_songs$performer)

#calculate the average rating of songs that each performer participates in
ratingbyperformer_songs=aggregate(performerrating_songs$rating,by=list(performerrating_songs$performer)
  arrange(desc(x))

#classify performers' popularity based on their songs' average rating
popularsinger_high=ratingbyperformer_songs[1:1985,]$Group.1
popularsinger_mid=ratingbyperformer_songs[1986:3971,]$Group.1
popularsinger_low=ratingbyperformer_songs[3971:5957,]$Group.1
songs$performer_popularity=as.factor(
  ifelse(apply(songs[,2:22], MARGIN = 1,FUN = function(x) any(x %in% popularsinger_high)),
         "High",
         ifelse(apply(songs[,2:22], MARGIN = 1,FUN = function(x) any(x %in% popularsinger_mid)),"Mid",
```

- Genre

To include valuable information in the "genre" column, I completed two things. I split each string in the "genre" column first. And the first completion is to classify genres based on their frequency in the analysis data. If one of the song's genre belongs to the group with highest frequency, genre_popularity='High', and so on. To be mentioned, here I split genres into 3 categories just by intuition. A possible improvement is to figure out a more accurate split rule.

```
genretype_songs<-str_split(songs$genre,",", simplify = T)
genretype_songs<-gsub("[[:punct:]]", "", genretype_songs)
genretype_songs<-apply(genretype_songs, MARGIN = 2, FUN = function(x) str_trim(x))
genretype_songs<-apply(genretype_songs, MARGIN = 2, FUN = function(x) str_to_title(x))
genretype_songs<-data.frame(id=songs$id, genretype_songs)
genre_1<-str_split(songs$genre,",", simplify = T)
genre_1<-gsub("[[:punct:]]", "", genre_1)
genre_1<-str_trim(genre_1)
genre_1<-str_to_title(genre_1)
genre_1<-unlist(as.list(genre_1))
genre_freq=data.frame(table(genre_1))
genre_freq=genre_freq%>%arrange(desc(genre_freq$Freq))
populargenre_high=genre_freq[2:200,1]
populargenre_mid=genre_freq[201:600,1]
populargenre_low=genre_freq[800:995,1]
genretype_songs$genre_popularity=as.factor(
  ifelse(apply(genretype_songs[,2:24], MARGIN = 1,FUN = function(x) any(x %in% populargenre_high)),
         "High",
         ifelse(apply(genretype_songs[,2:24], MARGIN = 1,FUN = function(x) any(x %in% populargenre_mid))
```

The second completion is to add in dummy variables about the 25 genres which most frequently appear in the data set. However, the problem that how many of the dummy variables should be included in deserves more consideration.

```
genretype_songs$MellowGold=as.factor(ifelse(
  apply(genretype_songs[,2:24], MARGIN = 1,FUN = function(x) any(x =="Mellow Gold")),1,0))
genretype_songs$SoftRock=as.factor(ifelse(
  apply(genretype_songs[,2:24], MARGIN = 1,FUN = function(x) any(x =="Soft Rock")),1,0))
#......
#Some same steps are omitted
genretype_songs$Trap=as.factor(ifelse(
  apply(genretype_songs[,2:24], MARGIN = 1,FUN = function(x) any(x =="Trap")),1,0))
genretype_songs[,26:50][is.na(genretype_songs[,26:50])]<-0

songs<-songs%>%
  inner_join(genretype_songs[,c(1,25:50)],by="id")
```

3. Model Training

After preparing for predictors, I started to train the tuned random forest model with ranger packages.

```
#Split data
library(caret)
set.seed(617)
split=createDataPartition(songs$rating, p=0.7, list = F)
train=songs[split,]
test=songs[-split,]
#Tune the model
library(ranger)
trControl=trainControl(method="cv",number=5)
tuneGrid = expand.grid(mtry=1:43,
                       splitrule = c('variance','extratrees','maxstat'),
                       min.node.size = c(2,5,10,15,20,25))
```

```
set.seed(617)
cvModel = train(rating~performer+genre_popularity+track_duration+track_explicit+danceability+
                  energy+key+loudness+mode+speechiness+acousticness+instrumentalness+
                  liveness+valence+tempo+time_signature+title_length+collaboration+MellowGold
                +SoftRock+AdultStandards+Rock+DancePop+Pop+BrillBuildingPop+Soul+
                  Motown+FolkRock+PopRap+AlbumRock+Rap+ClassicRock+QuietStorm+HipPop+Funk+
                  ClassicSoul+Rockandroll+BubblegumPop+Country+UrbanContemporary+Rb+
                  Disco+Trap,
                data=train,method="ranger",
                num.trees=1000,
                trControl=trControl,
                tuneGrid=tuneGrid )
cv_forest_ranger = ranger(rating~performer+genre_popularity+track_duration+track_explicit+danceability+
                            energy+key+loudness+mode+speechiness+acousticness+instrumentalness+
                            liveness+valence+tempo+time_signature+characterlength+cooperation+MellowGol
                          +SoftRock+AdultStandards+Rock+DancePop+Pop+BrillBuildingPop+Soul+
                            Motown+FolkRock+PopRap+AlbumRock+Rap+ClassicRock+QuietStorm+HipPop+Funk+
                            ClassicSoul+Rockandroll+BubblegumPop+Country+UrbanContemporary+Rb+
                            Disco+Trap,
                          data=train,
                          num.trees = 1000,
                          mtry=cvModel$bestTune$mtry,
                          min.node.size = cvModel$bestTune$min.node.size,
                          splitrule = cvModel$bestTune$splitrule)
```

Then I calculated RMSE of train data and test data as a reference.I made the same data processing work on the scoring data set, and output predictions.

Summary and Possible Improvement

In summary, through the kaggle project, my main work includes basic data tidying, trying to extract more valuable information from the analysis data set, tuning and building the random forest model. My contribution focuses on building new predicors, including "collaboration", "title_length", and predictors about songs' genres.

However, reviewing the whole process of playing with the data set, I think I could make improvement on the following aspects.

- More works on "performer" and "genre". There should be an accurate information mining regarding the songs' performer. Also, the processing method for genre needs to be more precise and succinct. For example, I would spend time on figuring out the optimal number of binary variables about genres, and the best space division of genres' popularity.
- Perform Feature Selection before training the model. I would choose to research on domain knowledge at first. Then I would make some efforts on dimension reduction, since too many preddictors tend to lead to overfitting and make tuning process time-consuming. Maybe the principal components analysis will be a choice.
- There could be more attempts on other models, like the boosting model which is powerful in many cases.
- Improvement on codes. Not only I'll try to make my codes neater and more time-efficient, but also methodically organized by making clearly comments and modularization.