Data Mining Final Project

Analyzing the music taste of a person through audio features of tracks----from Spotify user data

Bilgic, Utku Tarik

Xu, Xin

April 25, 2018

Motivation

There are many music streaming services being used today. Although it is very important for them to offer a variety of different songs from different artists, however it is not enough for making difference. They should offer more personalized services to their customers to increase their satisfaction.

In this project, our goal is to predict the songs a listener will like or dislike by analyzing the audio features of a song. This can be used to analyze the music taste of a person through audio features and offer new songs or playlist to the listener which will help her/him to explore new songs. By this way, more personalized music suggestion can be made.

To accomplish our goal, we used a dataset including 2017 tracks which has audio 15 features such as (acoustics, danceability, mode, …) and a target value indicating the song is liked or disliked by the listener. The descriptions of each feature in the dataset is given in Table A1 in Appendix. By using these features, we created models that will classify a song as 'liked or 'disliked' by the user. We created 2 classifiers one by using logistic regression, and the other by using decision trees.

Before starting our analysis, we preprocessed our data since we don't want some features to be dominant due to scale differences in data. By scaling 'duration', 'loudness' and 'tempo' features between [0, 1], we had all our continuous data between [0, 1] interval.

Also, as we didn't want our models to overfit to the dataset and can be used to predict new songs for the listener, we partitioned 70% of our data for training and remaining 30% for testing our models.

Logistic regression

First, we fitted a model including all the features without making any elimination.

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -10.47772  324.74474  -0.032 0.974261
acousticness      -1.85700    0.32424  -5.727 1.02e-08 ***
danceability       1.49559    0.44319   3.375 0.000739 ***
duration_ms        2.58949    0.77996   3.320 0.000900 ***
energy            -0.08749    0.52679  -0.166 0.868086
instrumentalness   1.31789    0.25176   5.235 1.65e-07 ***
key1              -0.29526    0.23342  -1.265 0.205910
key2               0.42486    0.25355   1.676 0.093809 .
key3              -0.91538    0.39650  -2.309 0.020965 *
key4              -0.07448    0.30730  -0.242 0.808500
key5               0.05598    0.27257   0.205 0.837281
key6              -0.18682    0.26157  -0.714 0.475087
key7               0.02090    0.24665   0.085 0.932464
key8              -0.23046    0.28476  -0.809 0.418341
key9               0.09227    0.25008   0.369 0.712155
key10              0.20664    0.28482   0.726 0.468127
key11             -0.33133    0.25584  -1.295 0.195298
liveness           0.03557    0.38205   0.093 0.925832
loudness          -3.15618    0.88026  -3.586 0.000336 ***
mode1             -0.04172    0.12380  -0.337 0.736108
speechiness        3.14167    0.70101   4.482 7.41e-06 ***
tempo              0.78504    0.38587   2.034 0.041906 *
time_signature3   10.48293  324.74396   0.032 0.974248
time_signature4   10.75411  324.74386   0.033 0.973582
time_signature5   10.83270  324.74410   0.033 0.973389
valence            0.85371    0.28192   3.028 0.002460 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

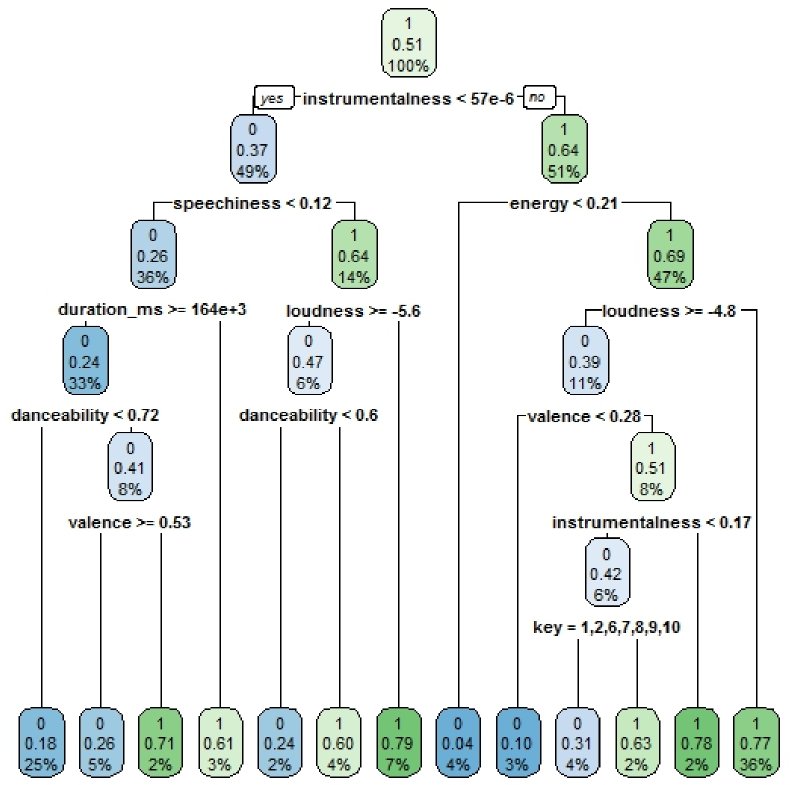
After that, when we check the below table, we realized not all features are important for deciding on the song to be liked or disliked, we decided to create another model with prominent features. Our final model is using 6 important features among of 16 features which determines the listeners music taste. Those features are; ‘acousticness’, ‘danceability’, ‘duration_ms’, ‘instrumentalness’, ‘loudness’, ‘speechiness’.
And when threshold is 0.4, we got the confusion matrix as below:

```
      0   1
  0 118 181
  1  50 256
```

The sensitivity is 70.23% which is fairly good.

Decision Tree

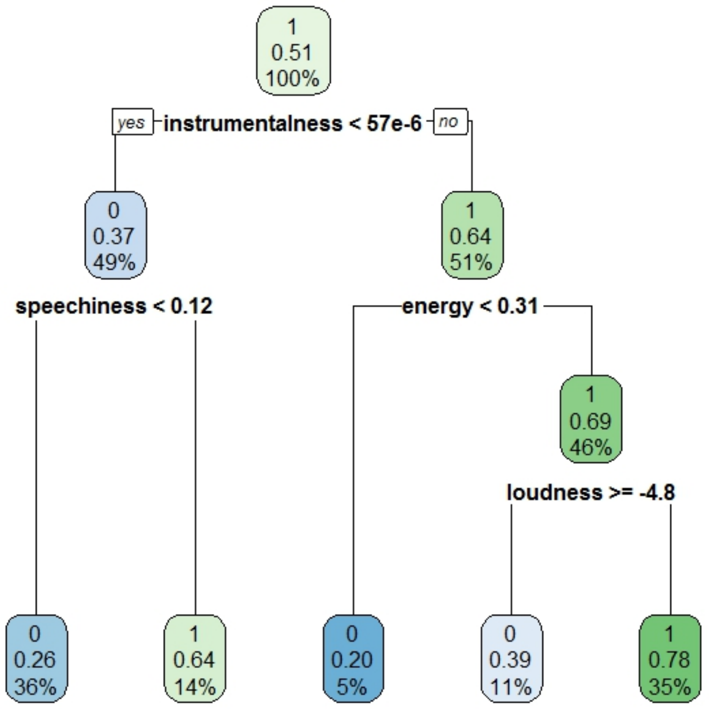
Then, we built a decision tree for this problem. We use all these 17 attributes of a song to analyze the person’s preference.



And we got the confusion matrix below:

```
      0   1
  0 195 104
  1  57 249
```

The sensitivity is 77.38% which is even better than the logistics regression one. However, if we see the decision tree, we would know the tree might be too large. And this can cause overfit. So we added a new attribute minbucket as 5% of test date in decision tree model. And we got the new decision tree model as below:



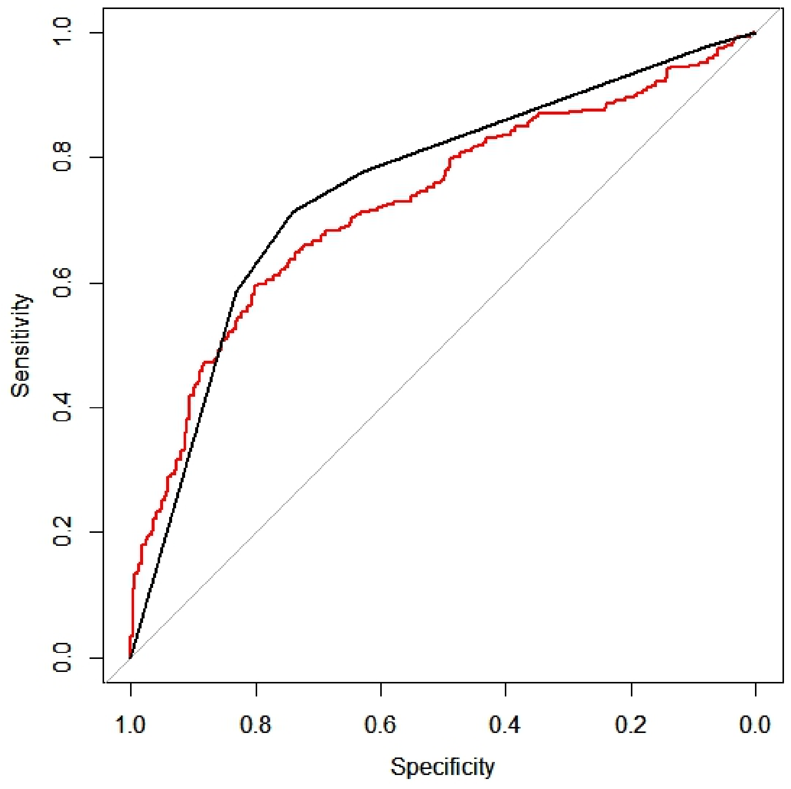
The corresponding confusion matrix is

```
      0   1
  0 222  77
  1  88 218
```

The sensitivity is 71.61% which is a little bit lower, but the tree became much simpler.

Compare Logistics Regression and Decision Tree

Both of these two methods showed good results in general. However, for different requirement we would have different preference. We plotted the ROC for these two models, found that when the specificity is lower, logistics regression line(red line) will be better, and when the specificity is higher, the decision tree(black line) will become better.
All R code is in Table A2 in Appendix



Appendix

Table A1 Description of features in the dataset

Feature	Value Type	Value Description
acousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
danceability	float	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
duration_ms	int	The duration of the track in milliseconds.

energy	float	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
instrumentalness	float	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
key	int	The key the track is in. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on.
liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
loudness	float	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
mode	int	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
speechiness	float	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

tempo	float	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
time_signature	int	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
valence	float	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
target	int	If the track is liked by the use or not. Liked is represented by 1 and disliked is 0.
song_title	char	Title of the song
artist	char	Artist of the song

Table A2 code for building Logistics regression model and decision tree model

```
#preprocessing of data
SpotifyData=read.csv("data.csv", header = TRUE)
SpotifyData$duration_ms=(SpotifyData$duration_ms-
min(SpotifyData$duration_ms))/(max(SpotifyData$duration_ms)-
min(SpotifyData$duration_ms))
SpotifyData$loudness=(SpotifyData$loudness-
min(SpotifyData$loudness))/(max(SpotifyData$loudness)-
min(SpotifyData$loudness))
SpotifyData$tempo=(SpotifyData$tempo-
min(SpotifyData$tempo))/(max(SpotifyData$tempo)-min(SpotifyData$tempo))
SpotifyData$key=as.factor(SpotifyData$key)
SpotifyData$time_signature=as.factor(SpotifyData$time_signature)
SpotifyData$mode=as.factor(SpotifyData$mode)

#training and test set seperation
split = sample.split(SpotifyData$target, SplitRatio = 0.7)
SpotifyData.Train = SpotifyData[split, ]
SpotifyData.Test = SpotifyData[!split, ]

#Logistic regression
glm.target=glm(target ~ acousticness + danceability + duration_ms + energy +
instrumentalness + key + liveness + loudness + mode + speechiness + tempo +
time_signature + valence, data=SpotifyData.Train, family = "binomial")
summary(glm.target)
glm.target=glm(target ~ acousticness + danceability + duration_ms + instrumen
talness + loudness + speechiness, data=SpotifyData.Train, family = "binomial"
)
summary(glm.target)
```

```
SpotifyPredictTest_LR = predict(glm.target, newdata = SpotifyData.Test, type
= "response")
threshold = 0.4
LogReg=table(SpotifyData.Test$target, as.numeric(SpotifyPredictTest_LR >= thr
eshold))
SensLR <- LogReg[1,1]/(LogReg[1,1] + LogReg[2,1])
SpecLR <- LogReg[2,2]/(LogReg[2,2] + LogReg[1,2])
ROC_curve_LR=roc(target~SpotifyPredictTest_LR, data=SpotifyData.Test)
plot(ROC_curve_LR)

#Decision Tree
rtree_fit <- rpart(target ~ acousticness + danceability + duration_ms + energ
y + instrumentalness + key + liveness + loudness + mode + speechiness + tempo
+ time_signature + valence, data=SpotifyData.Train, method = "class",
                   control = rpart.control(minbucket = 0.05*length(SpotifyDat
a.Train$target)))
rpart.plot(rtree_fit)

SpotifyPredictTest <- predict(rtree_fit, newdata = SpotifyData.Test, type = "
prob")
plot(roc(SpotifyData.Test$target, SpotifyPredictTest[,2]))

TbDT=table(SpotifyData.Test$target, as.numeric(SpotifyPredictTest[,2] >= thre
shold))
SensDT <- TbDT[1,1]/(TbDT[1,1] + TbDT[2,1])
SpecDT <- TbDT[2,2]/(TbDT[2,2] + TbDT[1,2])
plot(ROC_curve_LR, col="red")
par(new=TRUE)
plot(roc(SpotifyData.Test$target, SpotifyPredictTest[,2]))
```