STAT 632 FINAL PROJECT

Predicting Song Popularity

By
Mark Nevins
Florencia Irene
Odbileg Davaadorj

1.Introduction

In today's world, the music industry has become increasingly data-driven. Companies like Spotify have made it their business goal to quantify the sense of music, as they believe they can generate more revenue by using algorithms based on quantitative data.

The purpose of this project is to predict the popularity of a track. Specifically, we want to identify the qualities that are most strongly associated with popularity. By doing so, we can provide insights into what makes a hit. So, if we want to make a hit, what qualities should we include? Through our analysis, we hope to shed light on this question.

2.Data Description

2.1 Data Index

The dataset used in this project is Spotify's track dataset with a range around 114 genres and has dimensions of 21 columns and 114000 rows. Among 21 variables in the dataset, 6 of them are nuisance variables so in further analysis only 15 variables are included. The response variable is "popularity", measures the popularity of a track using values between 0 and 100 with 100 being the most popular. On the other hand, the predictor variables are: duration, explicit, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, time signature, and track genre. The description for each variables are provided in *Appendix I - Data Index*

2.2 Summary Statistics

2.2.1 Qualitative Variables

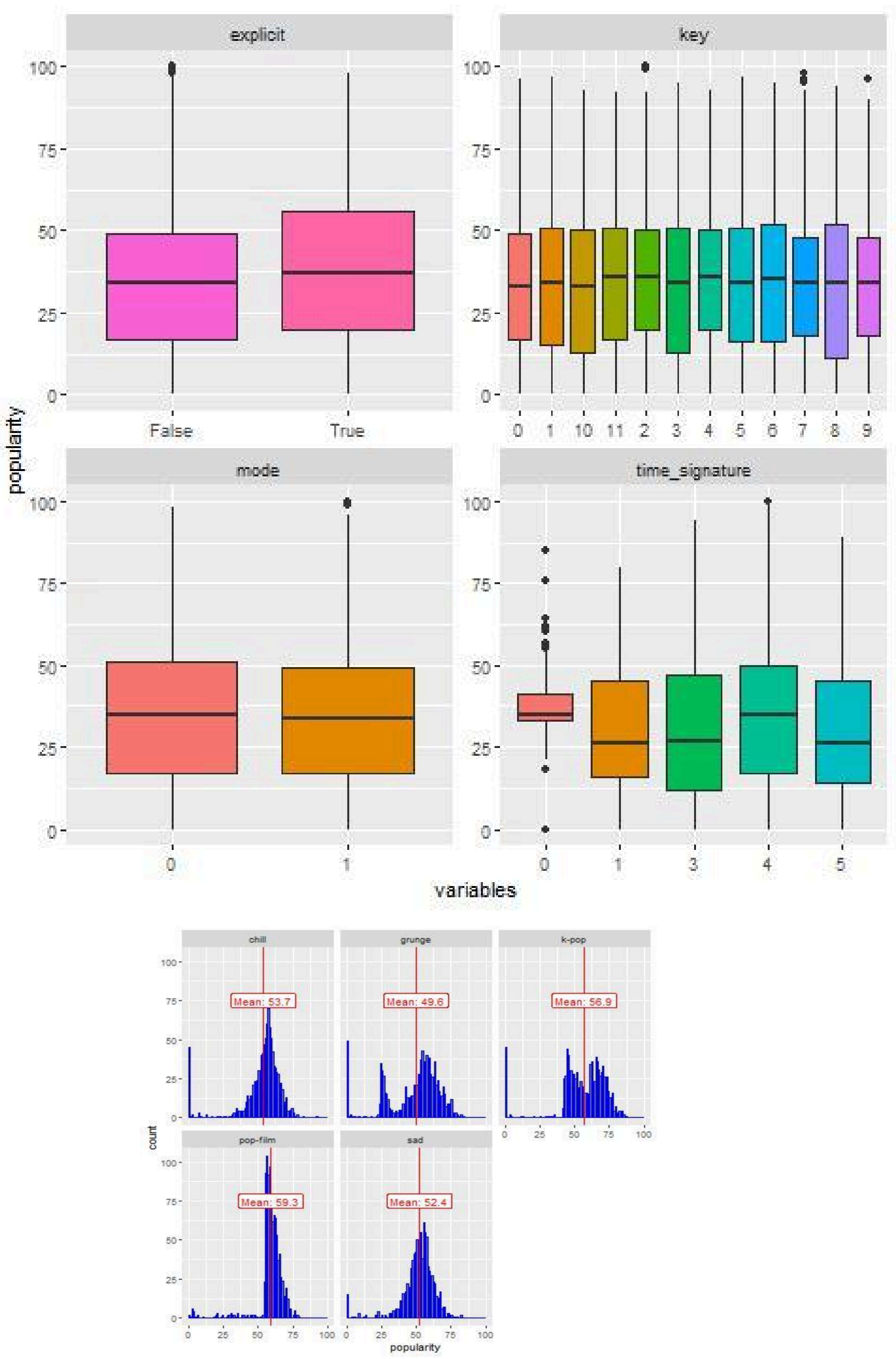
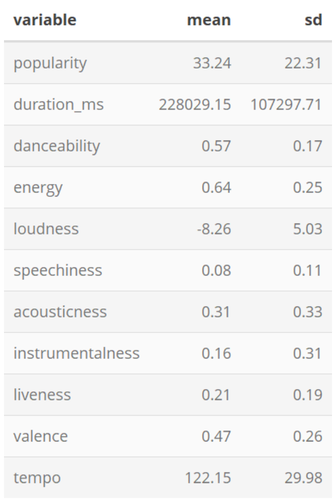


Figure 2.1

Based on figure 2.1 (left), all variables' levels except time signature have the same variance range in terms of popularity. In terms of time signature, songs that have 4/4 time signature(4) are more popular than others. Furthermore, to find the most popular track genre, mean of popularity for each track genres were calculated, Figure 2.1 (right) shows 5 most popular track genres, with pop-film, k-pop and chill are the top three.

2.2.2 Numerical Variables

There are 11 numerical variables founded in the dataset. Table 2.1 below shows the summary statistics (mean and standard deviation) for each of the variables.



variable	mean	sd
popularity	33.24	22.31
duration_ms	228029.15	107297.71
danceability	0.57	0.17
energy	0.64	0.25
loudness	-8.26	5.03
speechiness	0.08	0.11
acousticness	0.31	0.33
instrumentalness	0.16	0.31
liveness	0.21	0.19
valence	0.47	0.26
tempo	122.15	29.98

Table 2.1

Based on summary statistics on table 2.1 it is concluded that most of the songs have high energy and danceability as the mean for these variables is pretty high.

2.2.3 Correlation Matrix

In order to see the relationship between each variable, correlation coefficients for numerical variables were calculated. Figure 2.3 shows the correlation matrix for numerical variables:

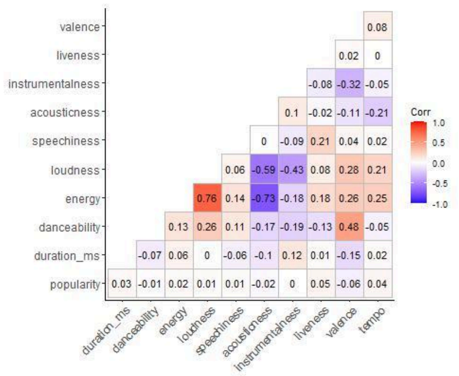
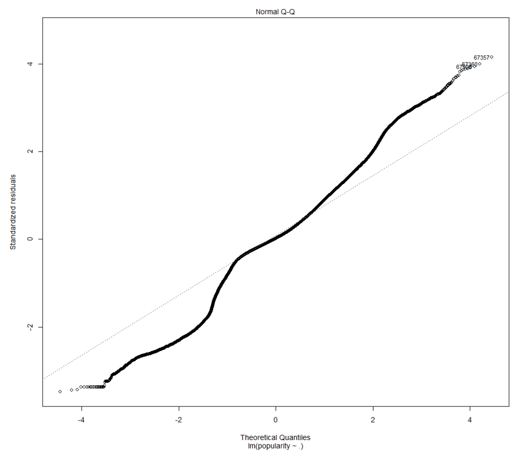


Figure 2.3

Based on figure 2.3 energy and loudness has the largest correlation value (0.76) which indicates that they have the strongest positive linear relationship. On the other hand energy and acoustic ness have large negative correlation values (0.73), which indicates negative relationship between these two variables.

3. Methods and results

The first thing we did to our data was some basic exploratory data analysis. Through this, we saw that our response definitely needed a transformation. Our Box-Cox transformation gave us a lambda value of 0.65, which we then applied to our response. However, the resulting multiple linear regression model showed that equal variance and normality were still violated. In particular, in figure 3.1 the points deviate from the Q-Q line, and in figure 3.2 the points are scattered in a pattern.



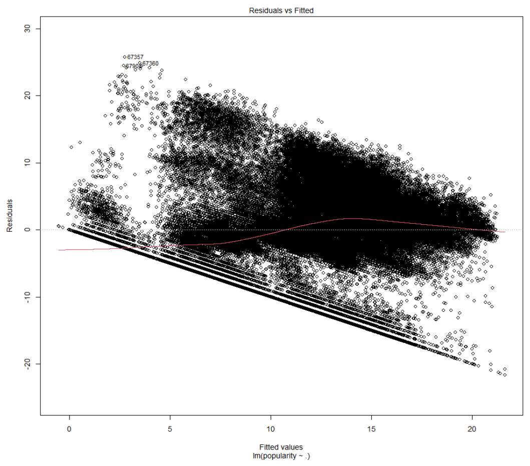


Figure 3.1 *Figure 3.2*

The next thing we did was split our data into an 80-20 testing-training split. We then expanded each of our factor variables into dummy coded columns, so that we can look at the significance of each column individually. We then did a multiple linear regression with all variables, and removed all predictors with a p-value over 0.1. We then repeated this process until no variables were left with a significance of over 0.1. At the end, our lm object had 116 predictors, an adjusted R squared of 0.2619, an AIC of 740423.5, and a mean squared error of 38.75.

Now that we had a solid multiple linear regression model, we wanted to see if we could use machine learning models to surpass the accuracy of our current model. The first of two models we looked at was LASSO. In short, LASSO is a method that provides variable selection and regularization to choose the best multiple linear regression model. Using cross-validation as our metric for accuracy, LASSO chooses which variable when added to the model will reduce our prediction error the most, and then steps forward, adding more variables. The model with the least prediction error corresponds to our minimum lambda value, but in our case we use a lambda value one standard error higher than the minimum as the minimum lambda was virtually indistinguishable from the full model without any variables removed. This LASSO model ended up with a mean squared error of 38.93.

Our other model was a random forest model. The random forest is simply an aggregation of the best performing trees when a random selection of predictors is chosen for it. Our random forest has a mean squared error of 38.02.

4. Conclusion

Based solely on our mean squared error, our random forest is our best model, followed by the multiple linear regression, and finally the LASSO model. The reason for the LASSO model being outperformed by the multiple linear regression is simple; the LASSO model with a lambda of 1.se has less variables than the multiple linear regression. Additionally, the multiple linear regression model does not consider overfit or multicollinearity in the way that the LASSO model does. Regardless, the best model for predicting on our test data is our random forest model.

Looking at the models, the highest predictors were all genres. The most significant predictor was Iranian music, and it was negative, indicating that Iranian music is severely unpopular. By analyzing the models given as well as the variable importance, we found that the optimal song for popularity was a pop song that was featured in a film soundtrack, had high values for loudness and length, had a BPM of around 171, and was in the key of E minor.

We had quite a few issues and limitations while we worked on the paper. For one, there was an issue of multicollinearity. After looking at the variance inflation factor, most variables are between 1 and 4, but the set of variables denoting time signature had very high values, ranging from 12 to 75. One reason for this could be that some genres like Jazz are closely linked to non-standard time signatures, giving a high correlation value.

Another limitation was how popularity was measured. Popularity was measured on a single day. Therefore, songs that were popular a few months ago have a much lower popularity score than music that was in the top 10 on that exact day of measurement. While this makes sense, I would like to think that the world of music isn't so rapid that a song that was popular a few months ago would not be of a similar popularity if released today.

Another oversight was the exclusion of polynomial terms. For many values, such as BPM, there is diminishing returns if not outright negative returns upon reaching a critical value. These need to be mapped in polynomial terms to be more accurate, which could be done in future analyses.

The final limitation that we had was in computing our random forest model. Our model had an mtry value of 3. Mtry is the parameter that controls how many input features we use in our trees. Normally for a regression tree, like the one we use, it is advised to go for around the square root as the number of predictors, which in our case is 12. However, due to a lack of computing power, as well as the sheer size of our dataset, we found that 3 was the upper limit for calculating the tree in a reasonable amount of time.

As a side note, there was a suggestion in class by Professor Kerr to remove all the zero values before fitting our model. However, when doing so, despite getting higher adjusted R squared values, our mean squared error values were actually higher than before! (e.g. MLR had an MSE of 37 with zeros and 96 without). As such, we stick by the idea of keeping in the zeroes.

Regardless, we feel that we have created a worthy model in this project, and hope that we could use this and future analyses to not just understand popular music, but grow a deeper appreciation for it.

Appendix I - Data Index

The description of each variables are:[1]

1. Duration: Track length in milliseconds
2. Explicit: Whether the track contains explicit lyrics (true = yes, false = no, or unknown)
3. Danceability: Measured by tempo, rhythm stability, beat strength, and overall regularity of a track to determine how suitable it is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable
4. Energy: A perceptual measure of intensity and activity from 0.0 to 1.0. Energetic tracks feel fast, loud, and noisy, for example, death metal.
5. Key: The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. *0 = C, 1 = C#, 2 = D*, and so on. The value -1 means there is no key that was detected.
6. Loudness: The overall loudness of a track in decibels
7. Mode: Indicating the modality (major or minor) of a track. Major is represented by 1 and minor is 0
8. Speechiness: Detecting 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
9. Acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic
10. Instrumentalness: 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
11. Liveness: 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
12. Valence: 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)
13. Tempo: 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

[1] "Spotify Tracks Dataset." *Kaggle*, https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset. Accessed 1 May 2023.

14. Time Signature: An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of 3/4, to 7/4.
15. Track Genre: The genre in which the track belongs

Appendix II - Code Appendix

Data Cleaning

Data Visualization Code

Qualitative variable

Figure 2.1 (left)

```
popularity <- spotify1$popularity
x.f <- Filter(is.factor,spotify1)
x.fac <- cbind.data.frame(popularity, x.f)
x.fac <- x.fac[,-6]
box <- melt(x.fac, id = "popularity")

box$value <- as.factor(box$value)
jpeg('factbox.jpeg')
ggplot(data = box, aes(x = value, y = popularity)) +
  geom_boxplot(aes(fill = value))+
  facet_wrap(~variable, scales = "free") + theme(legend.position = "none") +
  xlab("variables")
dev.off()
```

Figure 2.1 (right)

```
spotify1 %>%
  group_by(track_genre) %>%
  summarise(mean = mean(popularity)) %>%
  arrange(desc(mean))
# 5 best
track_best <- spotify1 %>%
  group_by(track_genre) %>%
  summarise(mean = mean(popularity)) %>%
  arrange(desc(mean)) %>%
  filter(mean > 49.539)
jpeg('pop.jpeg')
ggplot(track_best2)+
  geom_histogram(aes(popularity,col=popularity),
              breaks=0:100, color = 'blue', alpha = .75)+
  facet_wrap(~track_genre)+
  geom_vline(data = track_best, col = 'red',
             mapping = aes(xintercept = mean)) +
  geom_label(data = track_best, col = 'red',
             aes(x = mean,y = 75,
                 label = paste('Mean:',
                               round(mean,1))))+
  ylab("popularity")
dev.off()
```

Quantitative Variable

Table 2.1

```
num <- melt(x.num)
spotify1 %>%
  select(where(is.numeric)) %>%
  summarise_all(list(mean = mean, sd = sd))

a <- num %>%
  group_by(variable) %>%
  summarise_all(list(mean = mean, sd = sd))

a <- a %>%
  mutate(across(where(is.numeric), round, 2))
kable(a,
    col.names = c("variables","mean", "sd"),
    align = "lccrr")
kable(a) %>%
  kable_styling(bootstrap_options = "striped", full_width = F, position = "left", fixed_thead = T) %>%
  as_image(file = "table.png", zoom =1.5)
```

Correlation Matrix

Figure 2.3

```
x.num <- Filter(is.numeric, spotify1)
corr <- cor(x.num)
ggcorrplot(corr, type = "lower",  lab = TRUE, ggtheme = theme_classic)
```

Multiple Linear Regression Code

MLR Model

```
pacman::p_load(glmnet, randomForest, recipes, tibble)

dat <- read.csv("spotifydataset.csv", header = TRUE)

cldat <- dat[,c(-1:-5)]

cldat[,c("explicit", "time_signature", "track_genre", "key", "mode")] <- lapply(cldat[,c("explicit",
"time_signature", "track_genre", "key", "mode")], as.factor)

# cldat <- cldat[cldat$popularity != 0,]     #for removing zeroes

cldat$popularity <- cldat$popularity + 1


lm1 <- lm(popularity ~., data = cldat)
x <- powerTransform(lm1)
cldat$popularity <- bcPower(cldat$popularity, x$roundlam)
lm2 <- lm(popularity ~., data = cldat)
# summary(lm2)

#train test
trainind <- sample(1:nrow(cldat), size = floor(0.8*nrow(cldat)), replace = FALSE)
traindat <- cldat[trainind,]
testdat <- cldat[-trainind,]
#data prep
cldatt <- as_tibble(cldat)
dum <- cldatt %>%
  recipe(~ .) %>%
  step_dummy(explicit, time_signature, track_genre, key, mode) %>%
  prep(training = cldatt) %>%
  bake(new_data = cldatt)
#getting the mlr
lm5 <- lm(popularity ~., data = dum[trainind,])
lm6dat <- dum[trainind,c(-unname(which(summary(lm5)$coefficients[,4] > 0.1)))]
lm6 <- lm(popularity ~., data = lm6dat)
mean((cltest$popularity - predict(lm6, cltest))^2)
```

Other Models

```
rf <- randomForest(popularity ~., data = cltrain, mtry = 3)

mean((cltest$popularity - predict(rf, cltest))^2)

cvfit <- cv.glmnet(x = as.matrix(cltrain[,-1]), y = as.matrix(cltrain[,1]), family = "gaussian", alpha = 1)

mean(unlist((cltest[,1] - predict(cvfit, newx = as.matrix(cltest[,-1]))))^2) #38.93298
```