Music Prediction Based On Existing Choice

MAT 394 Machine Learning through R

Madhav Samanta
1810110116

Hrishikesh Sriman Narayan
1810110080

Pantho Chakraborty
1910110266

Table of Contents

1. ABSTRACT

Music is said to be a universal language and has the power to control our moods. Everyone has a different taste in music and with so many genres there is something for everyone. Nowadays every music app like Spotify, amazon music, apple, etc have this feature of 'shuffle' which randomizes our playlist which helps us discover new music or genres that we might like but the algorithm behind that shuffled playlist has to be very thorough as we don't want the customer to keep skipping the provided song in search of a 'good song'. We have data for 2017 songs (provided by Spotify) with information describing the song and whether our test subject likes it or not. We have analyzed the dataset with Logistic Regression, Random Forest, QDA (Qualitative Discriminant Analysis) and Naive Bayes. Random Forest has been found to have the best accuracy on this dataset.

2. INTRODUCTION

2.1 DATASET DESCRIPTION

A dataset of 2017 songs with attributes from Spotify's API. Each song is labeled "1" and "0" for songs the subject likes and dislikes respectively. The dataset has other columns other than the 'mode' and 'song number' describing the song in a quantitative manner, these include:

- Acoustics: 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: Describes how suitable a track is for dancing based on a combination of musical elements. A value of 0.0 is least danceable and 1.0 is the most danceable.
- Energy: 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.
- Instrumentals: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context.
- Liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
- Loudness: How loud the song should be for its best experience.
- Key: chords related to the major and minor scales.
- Mode: Each song is labeled "1" meaning the person likes it and "0" for songs he doesn't like.

2.2 MATHEMATICS BEHIND MODELS USED

2.2.1 Logistics Regression

Logistic regression uses an equation as the representation, very much like linear regression. Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is binary values (0 or 1) rather than a numeric value.

$$y = \frac{e^{(b_0+b_1*x)}}{1+e^{(b_0+b_1*x)}}$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

2.2.2 Quadratic Discriminant Analysis

QDA is a variant of LDA in which an individual covariance matrix is estimated for every class of observations. QDA is particularly useful if there is prior knowledge that individual classes exhibit distinct covariances. Since QDA estimates a covariance matrix for each class, it has a greater number of effective parameters than LDA.

2.2.3 Naive Bayes

Naive Bayes Classification is a method that uses the basic Bayes Theorem of Probability for classification problems.

The Bayes Theorem is as follows:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

$P(c|x)$= Posterior Probability | $P(x|c)$= Likelihood | $P(c)$=Class Prior Probability | $P(x)$=Predictor Prior Probability

Naive Bayes uses the combined probability of occurrence of certain predictors to predict the probability of occurrence of an event. It's conceptually a very simple method and very intuitive.

2.2.4 Random Forests:

This is a concept which essentially creates 'n' number of decision trees (also known as classification tree) models on the training set and averages out all the predictions. This is very similar to the concept of Bagging in which learning models are combined to receive a better model. Each tree is built as an individual model(H(x)) with each branch in the tree being decided by a set of 'm' predictors out of the total predictors 'p'. The classification is done on the basis of only one of those 'm' predictors and a fresh batch of 'm' predictors is drawn at each split. The value of 'm' at each split is approximately the square root of 'p'.

$$H(x) = \frac{1}{n}\sum_{i=1}^{n} H_i(x)$$

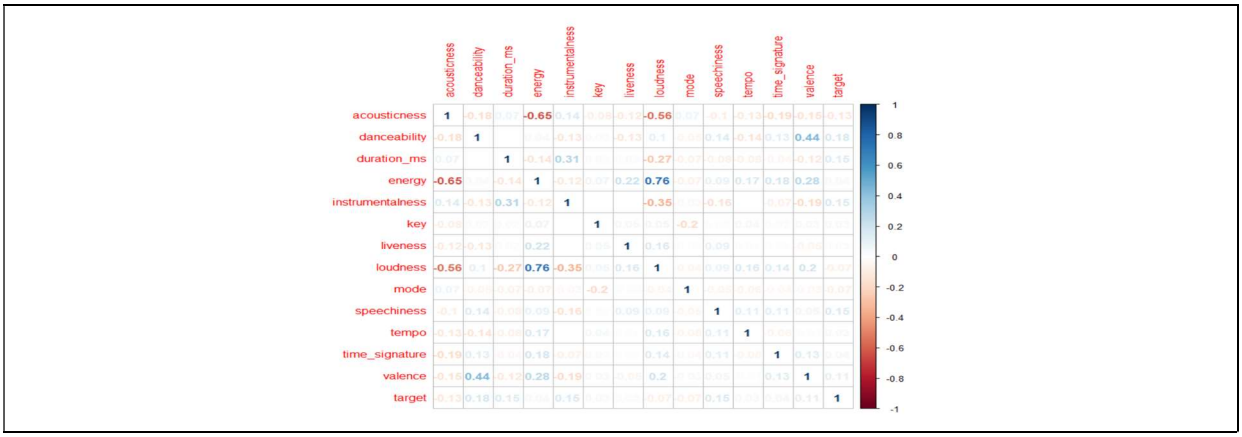
2.3 Experimental Analysis

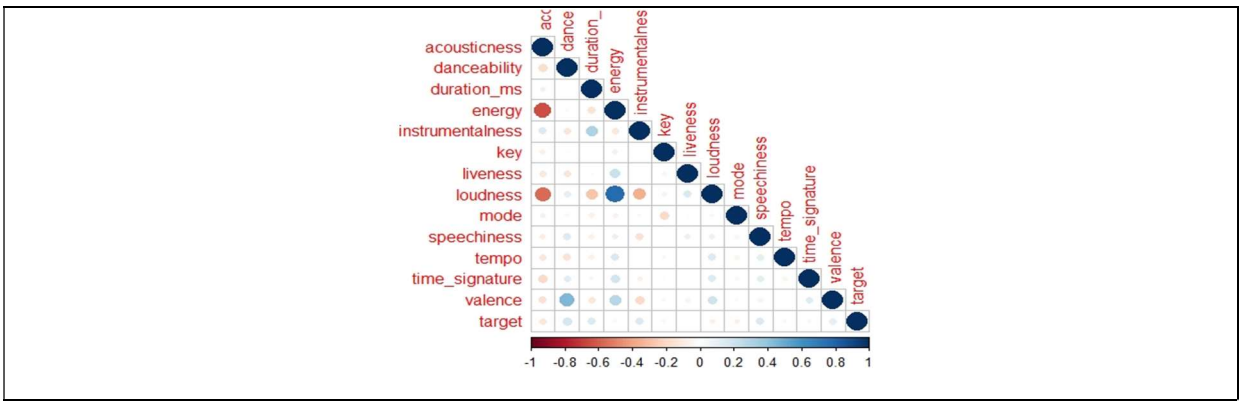
The project was done through 4 methods with 4 different models to try and analyze the best possible method with the highest accuracy.

These 4 methods are -

- Logistic Regression
- Quadratic Discriminant Analysis
- Naive Bayes
- Random Forest

We can visualize data to understand the correlation between the different variables using the corrplot package as -





The way we would begin every analysis would be first to load the dataset, then spit the dataset into two subsets. These are the training and testing. They would be split in the ratio of 0.8:0.2. By doing this we would have our model train with the training dataset and then use it on the testing dataset to predict the values. We check the accuracy of the model by calling in the confusion matrix.

Let us first look at

2.3.1 logistic regression

After loading the dataset, we performed a preliminary analysis of all the columns of the dataset to get a better understanding of the kind of process we would be performing.

To split the dataset into a training dataset and testing dataset we call upon the library(caTools) and use the split function. Our model named glm.fit invokes the glm function and compares the target variable (which in our case happens to be target) to all other variables and performs a regression. It takes in the training dataset to give out a binomial output.

The summary function helps us understand everything in it.

```
library(caTools)
split <- sample.split(data2,SplitRatio = 0.8)
split
training.glm <- subset(data2,split =="TRUE")
testingP.glm <- subset(data2,split =="FALSE")

glm.fit <- glm(target~.,training.glm,family="binomial")
summary(glm.fit)
```

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -4.533e+00  1.118e+00  -4.055 5.02e-05 ***
acousticness    -1.553e+00  2.985e-01  -5.201 1.98e-07 ***
danceability     1.788e+00  4.202e-01   4.254 2.10e-05 ***
duration_ms      2.188e-06  7.656e-07   2.858  0.00426 **
energy           6.454e-01  4.887e-01   1.321  0.18661
instrumentalness 1.394e+00  2.441e-01   5.710 1.13e-08 ***
key              4.830e-03  1.526e-02   0.316  0.75165
liveness         2.562e-01  3.592e-01   0.713  0.47557
loudness        -1.256e-01  2.727e-02  -4.608 4.07e-06 ***
mode            -1.289e-01  1.147e-01  -1.123  0.26125
speechiness      4.441e+00  6.980e-01   6.362 1.99e-10 ***
tempo            4.115e-03  2.158e-03   1.907  0.05659 .
time_signature   1.065e-01  2.277e-01   0.468  0.64011
valence          7.740e-01  2.683e-01   2.885  0.00391 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

It is important to understand what the *, **, *** stand for. These are the significance code that R provides us with. It tells how significant a particular variable is to the data.

It is clear that acousticness, danceability, instrumentalness, speechiness, and loudness play a very significant role in the like of a song.

*** - 99.9% confident

** - 99% confident

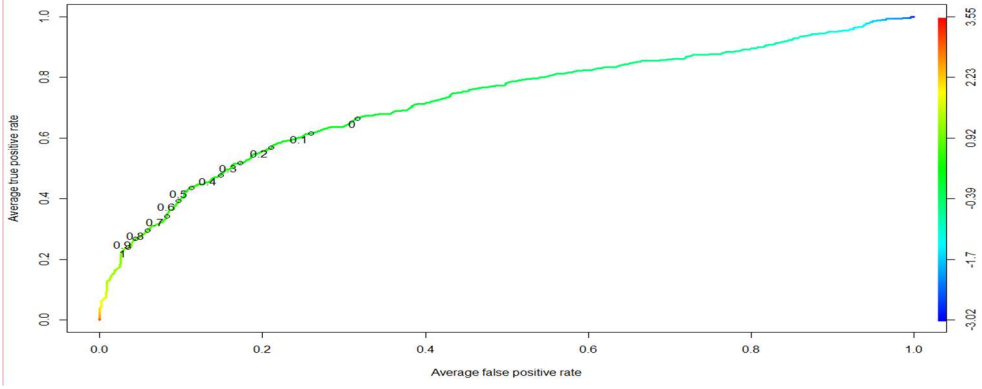
* - 95% confident

. - 90% confident

We now predict the values for the test data set then categorize them according to the threshold set.

In our case this threshold was set to 0.2. So, the values >0.2 will be rounded to 1 and <0.2 will be 0.A confusion matrix is then used to identify the accuracy.

We identified this threshold by plotting the ROCR curve. Given below is the plot -



2.3.2 Quadratic Discriminant Analysis

Here instead of using the split function we employ set.seed and then use the sample function to split our data set.

We proceed in a similar way to the previous method by defining a model and using it to predict the testing dataset.

```
qda.fit <- qda(target ~., data = training)

qda.pred <- predict(qda.fit,testing)
qda.class <- qda.pred$class
```

We are interested in what class we categorize into. Our classes are 0 and 1 hence with this we create a confusion matrix with only the class variable.

2.3.3 Naive Bayes

So, after creating a model which compares the target to the other variables taking in the training dataset, we print this. We get all the variables represented as

```
  danceability
Y        [,1]      [,2]
  0 0.5868222 0.1518962
  1 0.6487391 0.1614636

  duration_ms
Y       [,1]     [,2]
  0 233800.9 68656.40
  1 256747.3 86128.61

  energy
Y        [,1]      [,2]
  0 0.6687945 0.2403574
  1 0.6835568 0.1743420

  instrumentalness
Y        [,1]      [,2]
  0 0.0945276 0.2445153
  1 0.1638209 0.2901231
```

Here the [,1] and [,2] represent the mean and the standard deviation. For example, if we had to understand the first variable danceability, we would read it as the songs which are disliked (given by 0) has a mean danceability of 0.586822 with a standard deviation of 0.1518962, similarly, the songs which are liked have mean danceability of 0.6487391 with a standard deviation of 0.1614636.

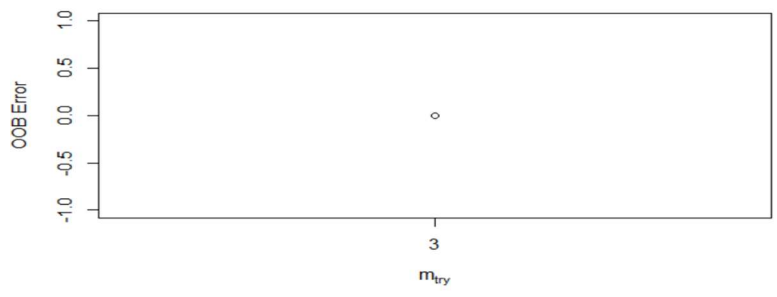
We can then have the model predict the results with the testing dataset and form the confusion matrix.

2.3.4 Random Forest

Proceeding as usual, but here we must take the best mtry value

```
bestmrty <- tuneRF(training.rf,training.rf$target,
                   stepFactor = 1.2,
                   improve = 0.01, trace = TRUE,plot = TRUE)
```

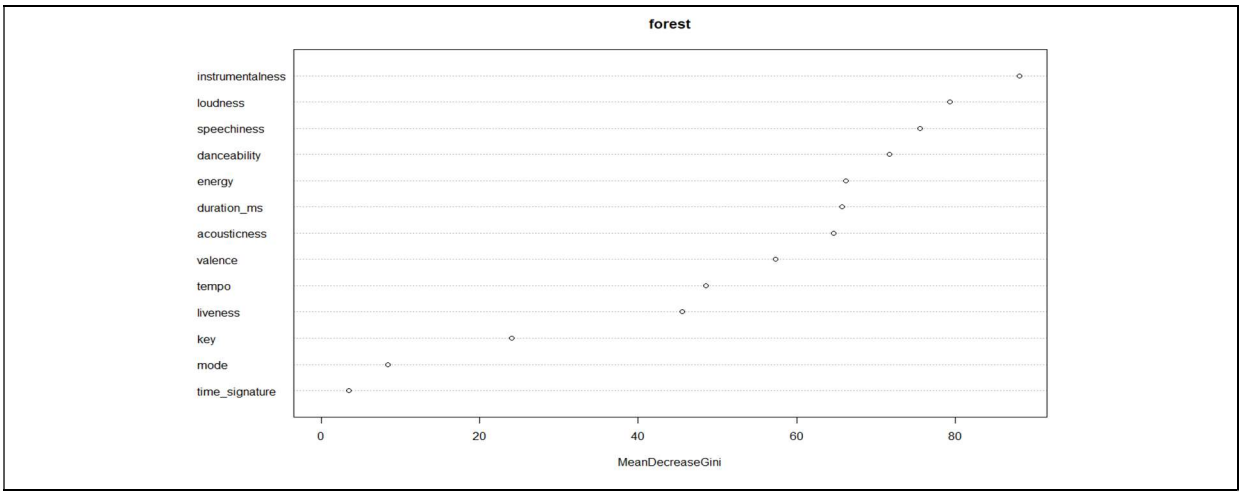
When you implement multiple decision trees in random forest in random forest, you don't actually take all the predictor variables then all decision trees will be similar so the machine does not learn from the multiple decision trees.



Through forest$importance we give priority to the variables to understand how much weight a particular variable has.

	MeanDecreaseGini
acousticness	64.661978
danceability	71.761044
duration_ms	65.739220
energy	66.232925
instrumentalness	88.114235
key	24.043263
liveness	45.536341
loudness	79.324230
mode	8.325514
speechiness	75.636631
tempo	48.501043
time_signature	3.484217
valence	57.333519

If we are to visualize this data, we get



2.4 Result Analysis

We will compare the confusion matrices of all the different methods to conclude best method out of the 4 used in this project.

2.4.1 Logistic Regression

	PredictedValue	
ActualValue	FALSE	TRUE
0	152	61
1	90	129

With an accuracy of 0.65046 or 65%.

2.4.2 Quadratic Discriminant Analysis

qda.class	0	1
0	223	89
1	88	218

Accuracy : 0.7136

Accuracy of 71%

2.4.3 Naive Bayes

```
nb.pred   0   1
      0 145  78
      1 166 229

               Accuracy : 0.6052
```

2.4.4 Random Forest

```
probs_forest   0   1
           0 239  65
           1  72 242

               Accuracy : 0.7783
```

From this, we can conclude that random forest has the highest accuracy and is, therefore, the best model for the given data.

3. Conclusion:

AI and Machine Learning have become a very integral part of the online entertainment industry in terms of User Experience. By using similar techniques to the ones mentioned here various and not to mention, popular software has been generated (Spotify, Netflix, YouTube, and Kindle to name a few). This has both increased the ease of access of the audience to quality content and their overall experience. This has an application not just in music platforms (as discussed here) but various other platforms built for movies, games, books, and so on. Since Art and Entertainment are Subjective in nature, creating accurate mathematical predictors often stands as a challenge. The Goal was to create such a model, and Random Forest Method has given the

4. References:

https://builtin.com/data-science/random-forest-algorithm

Github Link https://github.com/SMadcat/MAT_394MLPROJECT.git