6414 Project Report

Group 4

SONG POPULARITY PREDICTION

Asha Gutlapalli, Diwash Bajracharya, Shruthi Laya Hariharan

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Music is the most effective form of art that has the capacity to make humans feel all types of emotions and deeply impacts the minds and hearts of people. The project aims to comprehend the parameters involved in making a song popular and understand the factors that contribute to a song's success or failure and reveals which attributes influence a listener's inclination to a song the most. From the literature study the following parameters were considered as the majority parameters impact the popularity of a song: Genre, Year, Beats per minute, Energy, Danceability, Loudness, Liveness, Valence, Length, Acousticness and Speechiness. Through several different feature selection algorithms, the project aims to identify the most influential features in the dataset

Different regression models such as Logarithmic, LASSO, Stepwise, Ridge, Elastic net, K-Nearest Neighbor, RandomForest and Decision Tree are experimented with to narrow down to one model that best fits the data. The models are tested on test data to give an unbiased performance estimate and the models are compared based on accuracy, sensitivity, and specificity. The goal of the project aims to primarily identify the model having the highest prediction accuracy for the given dataset and analyze the features that influence the popularity of a song.

2. MOTIVATION

Comprehending the reasons behind popularity of a song has major business implications to industries that thrive on popular music, namely radio stations, record labels, and digital and physical music marketplaces. The ability to make accurate predictions of song popularity also has implications for customized music suggestions and to help artists and record labels maximize commercial return is to use a model to predict whether their music will be popular on streaming platforms. Music online platforms like Spotify, Apple Music, Pandora train their algorithms to recommend music based on the user's preferences. Predicting popular songs can be extended to identify user preferences among different population segments which can enable the ability to tailor streaming apps and radio stations to understand the needs of the audience and helps in predicting preferred songs for that segment of population.

3. DATASET

The data was sourced from Kaggle, an online platform allowing users to find data sets. The dataset contains the following parameters -

- Index: ID
- Title: Name of the Track
- Artist: Name of the Artist
- Top Genre: Genre of the track

- Year: Release year of the track
- Beats per Minute (BPM): The average beats per minute indicates the tempo of the song. The beats per minute is a direct indication of the tempo of the song.
- Energy: The energy of a song is the sense of forward motion in music and parameters that keeps the listener engaged and listening to the song.
- Danceability: Danceability is measured using a mixture of song features such as beat strength, tempo stability, and overall tempo. The value returned determines the ease with which a person could dance to a song over the course of the whole song. The higher the value, the easier it is to dance to the song.
- Loudness: 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. The higher the value, the louder the song is measured in dB.
- Valence: Valence describes the musical positiveness of the song. The higher the value, the more positive the mood (cheerful, euphoric, and happy) is in the song.
- Length: The duration of the song.
- Acoustic: Acoustic music is music that solely or primarily uses instruments that produce sound through acoustic/natural instruments means, as opposed to electric or electronic means. The higher the value the more acoustic the song is, usually involving the flute, cello, drums, or violins. Typically, songs using EDM/ Electric instruments have a lower acoustic-ness.
- Speechiness: It is a direct relation to the number of spoken words in the song.

The above parameters were measured and analyzed if it has a direct correlation with the popularity of the song.

4. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis is a crucial process of using summary statistics and graphical representations to perform preliminary investigations on data to uncover patterns, detect anomalies, test hypotheses, and verify assumptions. EDA was performed on the dataset to discover trends if any, observed within the chosen set.

The following key observations were uncovered while performing EDA -

- The frequency of the number of hit songs in the data were predominantly present from 1965 and increased in a linear fashion through the horizon until 2015.
- The average beats per minute of popular songs lie between 100-140 BPM.
- The energy of the popular songs has a direct relation, with an average of 60-80 dB

- The dance-ability of songs follows a normal distribution with the average danceability of the song is approximately 50-60%, as shown in Figure 1 below.

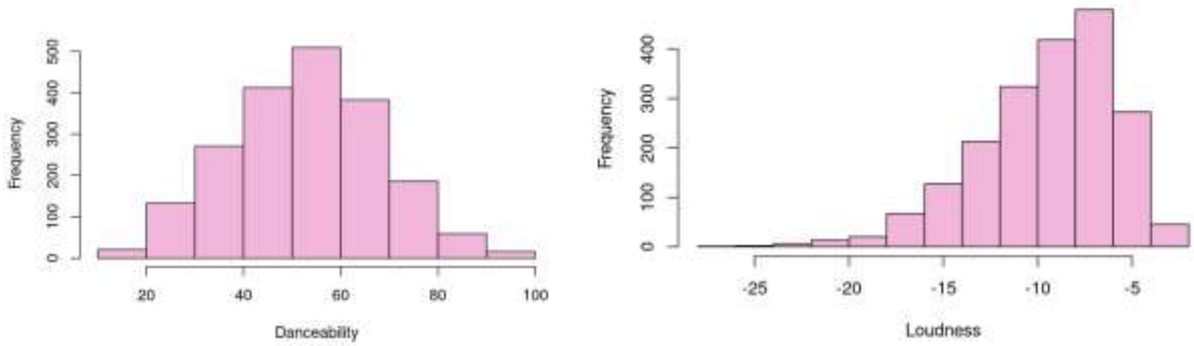


Figure 1. Frequency of Dance-ability and Loudness in popular songs

- Most of the popular songs exhibit a loudness from -13 to -7 dB, as indicated by the graph above.
- Predominantly, popular songs have less liveness, indicating that the songs were recorded in a studio.
- For valence of the song, it indicated that there is no trend observed with valence and popularity of songs.
- Popular songs are usually in the length of 200-300 seconds.
- Electronic music has higher popularity, compared to country music which have higher acousticness.

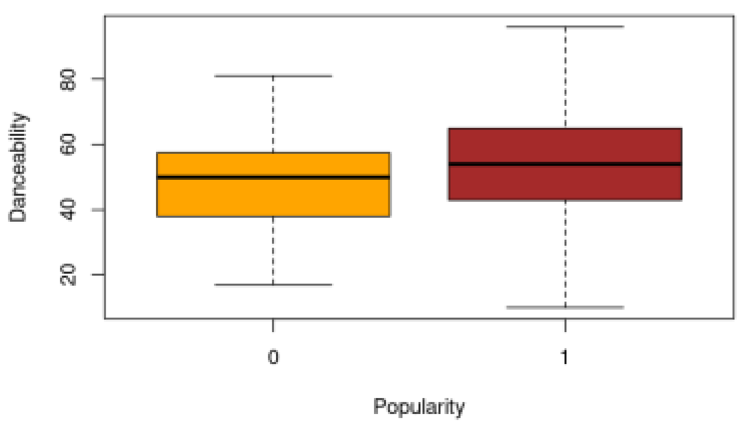


Figure 2. Distribution of Dance-ability

After converting the popularity of the songs into a binary variable with k=0.4, where 0 indicates less popular and 1 indicates a higher popularity, danceability of higher popular songs are distributed throughout the range with a mean danceability higher than less popular songs, as depicted in Figure 2.

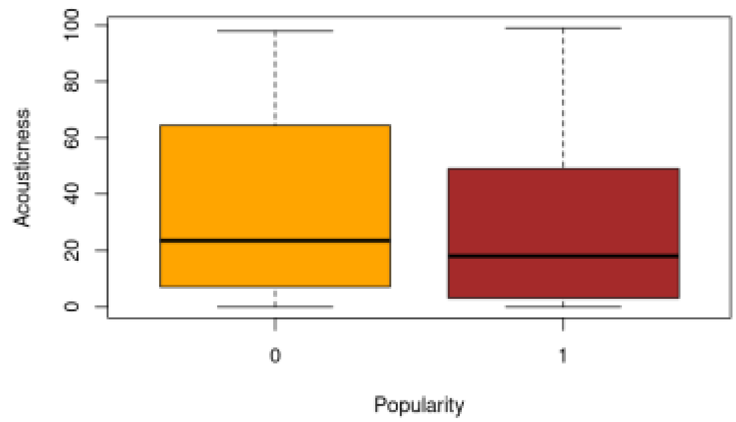


Figure 3. Frequency of Acousticness

The Figure 3 indicates that highly acoustic songs have comparatively less popularity than songs with a lower acousticness. Further data analysis shows that there is no distinct difference in speechiness, length, liveness and beats per minute between high popularity and lesser popular songs.

Cooks distance is used to analyze and calculate influential outliers in a set of predictor variables and indicates a way to identify points that negatively affect regression models. The cooks distance of the dataset was plotted against the number of observations, as shown in Figure 4, and indicates that there are less influential points that affect the regression models.

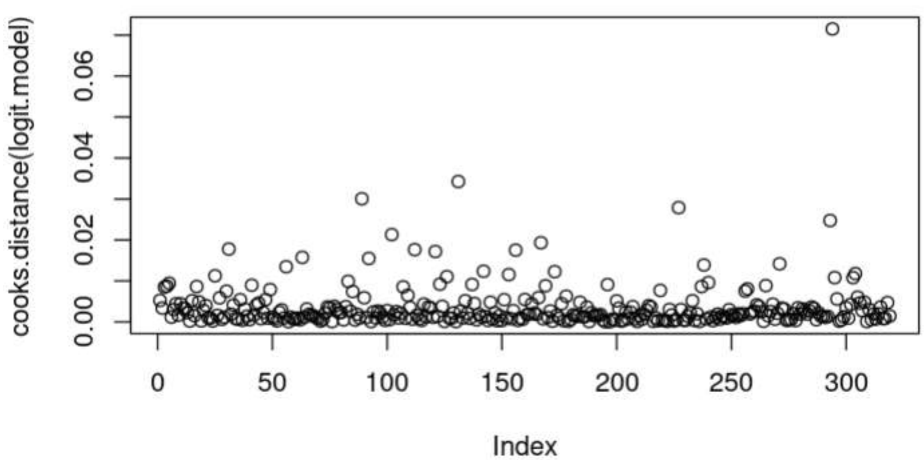


Figure 4. Cooks Distance Plot

For further modeling of the data, missing data points in the dataset were investigated in python to visualize the summary of completeness of the dataset. It was concluded that there were no missing points in the dataset, as shown in the attached figure 5.

```
[6] df_1 = pd.read_csv('/content/Spotify-2000.csv')

[8] df_1.isna().sum()

Index                     0
Title                     0
Artist                    0
Top Genre                 0
Year                      0
Beats Per Minute (BPM)    0
Energy                    0
Danceability              0
Loudness (dB)             0
Liveness                  0
Valence                   0
Length (Duration)         0
Acousticness              0
Speechiness               0
Popularity                0
dtype: int64
```

Figure 5. Analysis of Missing Datapoints

5. APPROACH

Various regression methods were explored to model the data. Since the response variable was transformed from a numerical variable to a binary response variable, regression techniques that have a logit link function were chosen. Linear regression is not appropriate to answer Yes/No questions as it does not adequately capture the behavior of music charts. There are a lot of features in the dataset, and some of them may not be that significant. Hence, variable selection models were also used. Ensemble learning and non-parametric supervised learning have been gaining a lot of popularity due to their high accuracy and ability to generalize well on data. This type of learning was also adapted in this project. The regression models implemented in this project are listed below and explained in detail:

- Logistic Regression
- Stepwise Regression
- Lasso Regression
- Ridge Regression
- Elastic Net Regression
- K-Nearest Neighbor
- Decision Tree
- Random Forest

5.1 LOGISTIC REGRESSION

Logistic Regression is a statistical model that has a logit link function. It models the probability of an event occurring. For example, in this project, the probability of a song being popular after its release. Hence, this model is appropriate for this use case.

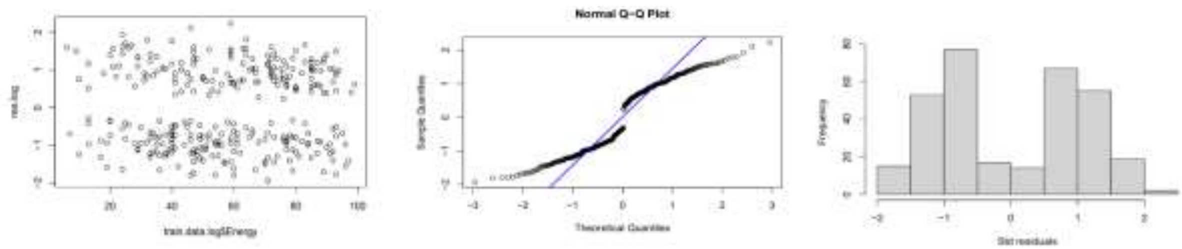


Figure 6. Logistic Regression Residual Analysis

It is known that the constant variance assumption and the normality assumption do not exist for logistic regression. This is shown in Figure 6, residual analysis as it violates these conditions due to bi-modality. According to the person's residuals, the model gives a P-Value of 0.46. Since this is greater than 0.03, the null hypothesis is accepted that the model fits the data. Hence, the goodness of fit test passes. Moreover, the model achieves high accuracy, sensitivity, and specificity of 0.81, 0.80, and 0.81 respectively.

5.2 STEPWISE REGRESSION

Stepwise Regression is a variable selection regression method. It follows a procedure wherein it decides whether to include or discard a particular variable based on a criterion. In the forward stepwise regression, the regression model starts with minimal features and keeps adding to these features if it improves the performance of the overall model. In the backward stepwise regression, the regression model starts with all features and keeps removing features if it improves the performance of the overall model while still keeping the minimal features intact. Bi-directional stepwise regression is a technique that combines both of these methods. Bi-directional stepwise regression was implemented in this project. It selects Year, Danceability, Liveness, Loudness, and Speechiness as its attributes. Year and Danceability are the most significant attributes in the model with P-values very close to zero. Liveness and Loudness are also quite significant with P-values below 0.05. This model reaches the highest accuracy out of all the models that were included in the experimentation. It achieves an accuracy, sensitivity, and specificity of 0.82, 0.73, and 0.90. This makes sense as a few variables are sufficient to explain the variability in the data. Using all the variables might have been overfitting the data and thereby failing to predict well on unseen data. The advantages of stepwise regression are that it is a faster variable selection method and handles even large amounts of predictor variables. The disadvantages of stepwise regression are that it has inconsistencies with variable selection and includes bias in parameter estimation. The stepwise regression model was chosen as the final model based on the evaluation metric, accuracy. It is used to predict new unseen test data.

5.3 LASSO REGRESSION

LASSO Regression or the Least Absolute Shrinkage and Selection Operator is also a regression method that performs variable selection. This is done with the help of L1 regularization. It adds a penalty term to the loss function. This technique forces some of the coefficients of the variables to zero to reduce the complexity of the model.

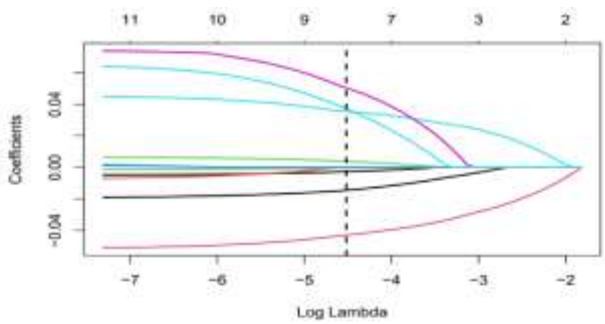


Figure 7. Lasso Regression Plot

In Figure 7, the relationships between log lambda and the coefficient values are observed as the number of variables is decreased in the model. The model selects Top Genre, Year, Beats Per Minute, Danceability, Loudness, Liveness, Length, and Speechiness as its variables. The variables Energy, Valence, and Acouticness are removed from the model. It achieves an accuracy, sensitivity, and specificity of 0.77, 0.70, and 0.88 respectively.

5.4 RIDGE REGRESSION

Ridge Regression is not a variable selection technique like LASSO regression. It is used when the predictor variables are highly correlated. It regularizes the coefficients of the model by making sure that none of the variables highly influence the response variable. This is done using L2 regularization. It adds a penalty term to the loss function.

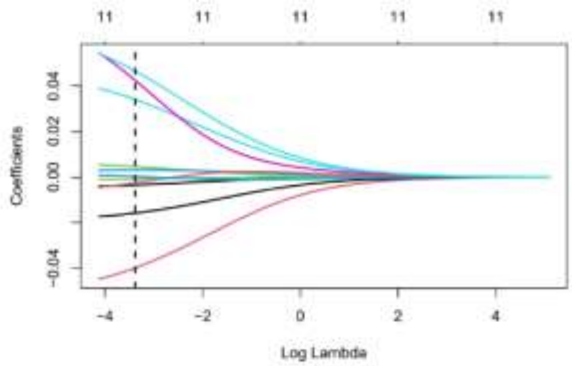


Figure 8.Ridge Regression Plot

In Figure 8, the relationships between log lambda and the coefficient values are observed as the number of variables is maintained the same in the model. The model selects all the variables. The coefficient values of some of the variables are increased while some of them are decreased for regularization. It achieves an accuracy, sensitivity, and specificity of 0.79, 0.75, and 0.83 respectively.

5.5 ELASTIC NET REGRESSION

Elastic Net Regression is also a regularized statistical method. It is a combination of both the LASSO regression and the ridge regression techniques. It is used for variable selection and for regularizing the coefficient values with both L1 and L2 regularizations. It adds a penalty term to the loss function. The tradeoff between LASSO and ridge regressions can be adjusted according to preference.

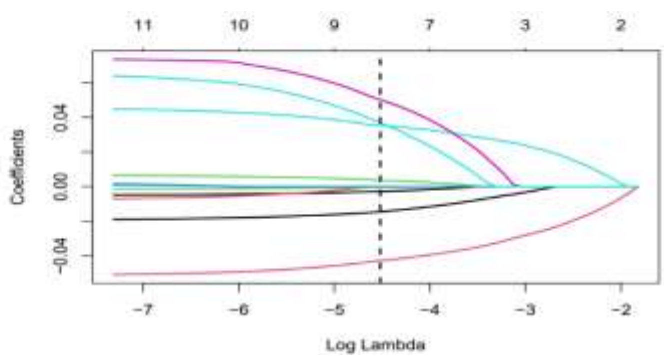


Figure 9. Elastic Net Regression Plot

In Figure 9, the relationships between log lambda and the coefficient values are observed as the number of variables is decreased in the model. The model selects Top Genre, Year, Beats Per Minute, Danceability, Loudness, Liveness, Length, and Speechiness as its variables. The variables Energy, Valence, and Acousticness are removed from the model. It achieves accuracy, sensitivity, and specificity of 0.77, 0.71, and 0.85 respectively.

5.6 K-NEAREST NEIGHBOR

K Nearest Neighbor (KNN) is a supervised machine learning algorithm that is useful for classification and regression. KNN suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new datapoint into one of the categories. KNN classifies the songs into either popular or not popular based on its nearest k data points. Using the KNN package in R, we set the maximum k as 50 given we have 2000 data points.

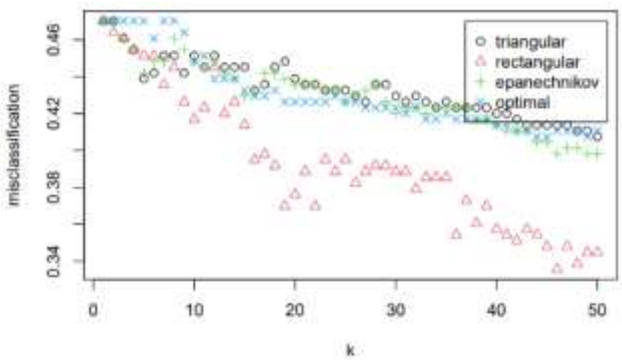


Figure 10.K-Nearest Neighbor Plot

In Figure 10, the misclassification of the song popularity for the different considered values of k from 0-50. As seen on the plot, the number of misclassifications decreases as KNN method increases the k. The algorithm selects best k as 46 and achieves accuracy, sensitivity, and specificity of 0.79, 0.75 and 0.83 respectively.

5.7 DECISION TREE

Decision Tree build regression or classification models in the form of a tree structure. It breaks down a dataset into progressively smaller subsets while at the same time an associated decision tree is incrementally developed. Decision trees seek to find the best split to subset the data and the recursion terminates when the subset at a node having the same value of target variable or when splitting does not add value to the prediction. The final result is a tree with decision nodes and leaf nodes. For each leaf node, a separate regression is run, and it is used to predict whether a given song is classified as popular or not.

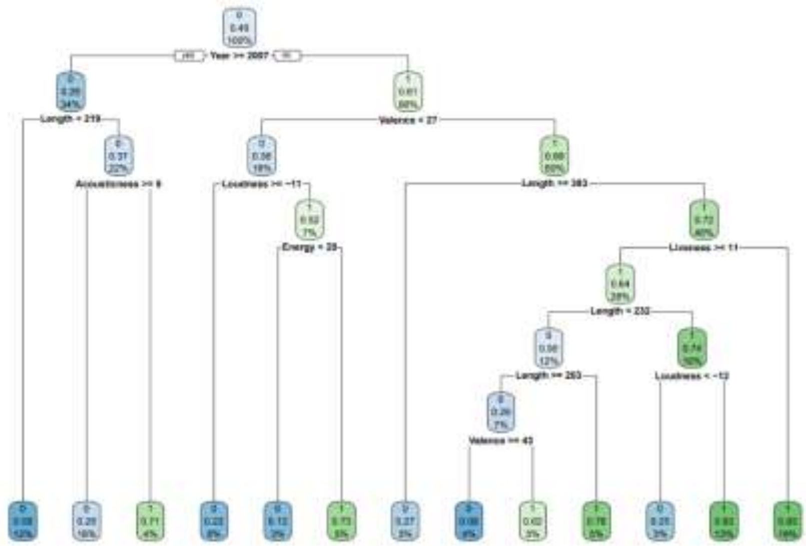


Figure 11. Decision Tree Plot

In Figure 11, the model breaks the dataset first on whether a song is recently released (year 2007 and after) or not. Then it keeps creating further branches and leaves. This method is easy to interpret and explain to others as it generates understandable rules for splitting the dataset. However, it is found that the best model for estimation and prediction tasks as it can disproportionately weigh different attributes and can be prone to bias and overfitting. The model achieves accuracy, sensitivity, and specificity of 0.63, 0.69 and 0.58 respectively, which is the least performing model among the various models we explore in our analysis.

5.8 RANDOM FOREST

Random forests are an ensemble learning method for classification & regression that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Random first algorithm is made up of many decision trees, however, while decision trees consider all possible feature splits, random forests only select a subset of those features.

Random forests have many benefits over using a single decision tree such as reduced risk of overfitting due to it considering multiple decision trees that somewhat neutralizes over-fitting, better overall estimation with a higher degree of accuracy and makes it easy to evaluate attribute importance. This model achieves accuracy, sensitivity, and specificity of 0.77, 0.77 and 0.77 respectively, which is significantly better than for a decision tree. However, performing random forest has a high computing cost compared to a decision tree, and is a time-consuming process for large datasets. Random forest is also difficult to interpret and explain the results compared to decision trees where decisions can be made following the path of the flowchart.

6. DISCUSSION AND RESULTS

One of the first things carried out for the analysis was performing Exploratory Data Analysis (EDA) in order to understand the distribution of our response variable and predicting variables. It is found that while few attributes such as danceability and beats per minute are more or less normally distributed, most of the other attributes are not normally distributed. This observation underlines that normal distribution assumption cannot be made for the analysis hence, linear regression might not yield the best model accuracy as later confirmed during the analysis. It is also concluded from the correlation heat map that some attributes such as Acousticness and Energy, and Energy and Loudness are highly correlated (correlation magnitude >0.5). This suggests that the final model is likely to end up using one of the two highly correlated attributes than both.

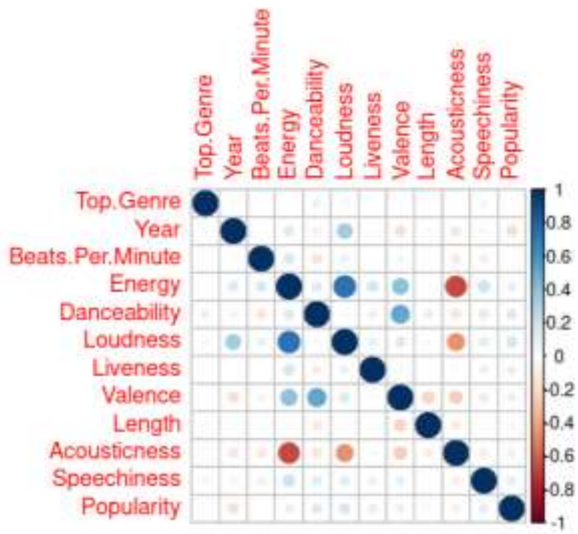


Figure 12. Correlation matrix plot for numerical variables

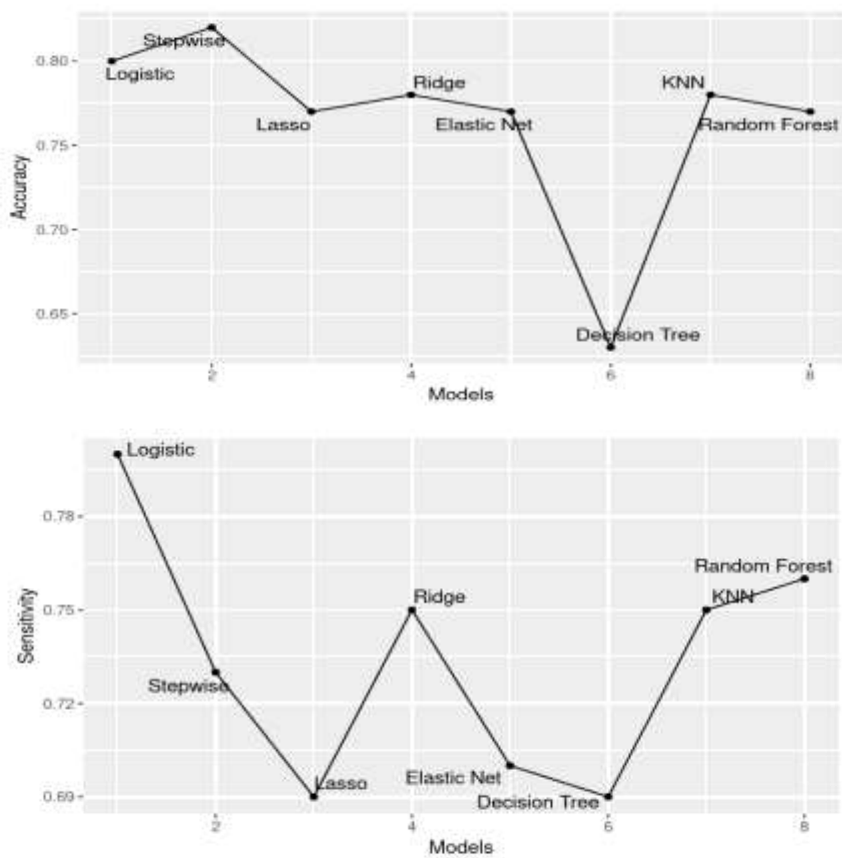
The response variable in the dataset (Popularity) is a continuous variable and had a score between 0-100, with 100 being most popular. Initial analysis involved the creation regression models based on continuous variables with RMSE scores between 12-15. However, the approach is changed to create regression models with binary response variables as it is convenient to compare the various models across standard metrics.

Side-by-side comparison of the various models for determining the best popularity prediction model can be seen in Figure 13 below. It is observed that most models have accuracy, sensitivity, and specificity in the 70-80% range. Based on the results and prediction accuracy of the various models, it is determined that stepwise linear regression has the highest predicting accuracy and decision tree model has the least prediction accuracy.

Name of the Model	Accuracy	Sensitivity	Specificity
Logistic Regression	0.8070	0.8000	0.8125
Stepwise Regression	0.8245	0.7307	0.9032
LASSO Regression	0.7719	0.6969	0.8750
Ridge Regression	0.7894	0.7500	0.8275
Elastic Net Regression	0.7719	0.7096	0.8461
Decision Tree	0.6315	0.6923	0.5806
K-Nearest Neighbor	0.7894	0.7500	0.8275
Random Forest	0.7712	0.7692	0.7741

Figure 13. Comparison of Performance of Models

It is observed that ridge regression selects all the variables, as it is not a variable selection method and the performance of various models are compared in Figure 14.



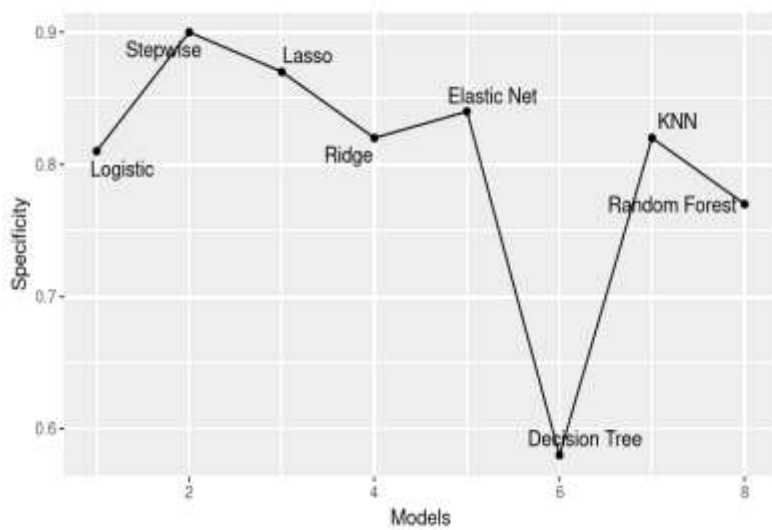


Figure 14. Comparison of various models in predicting the accuracy, sensitivity and specificity

To determine which attributes are most important in determining a song's popularity, different variable selection methods such as Lasso, stepwise regression and decision tree are performed. Side-by-side comparison of the various attributes selected by various models can be seen in Figure 15 below.

It is found that Stepwise regression is the best model for predicting popularity of a song and reduces the most number of variables and selects only 5 of the 11 attributes available for its model, namely Year, Beats Per Minute, Danceability, Loudness and Length. This seems logical as some of the variables were highly correlated to each other as we found in our EDA. It was interesting to find that Year was one of the significant features in predicting a song's popularity. However, this can be explained as a song's year is correlated with other attributes such as Loudness and Danceability.

Model	LASSO	Ridge	Elasti Net	Stepwise	Decision Tree
Top Genre	✓	✓	✓	×	×
Year	✓	✓	✓	✓	✓
Beats Per Minute	✓	✓	✓	✓	×
Energy	×	✓	×	×	✓
Danceability	✓	✓	✓	✓	×
Loudness	✓	✓	✓	✓	✓
Liveness	✓	✓	✓	×	✓
Valence	×	✓	×	×	✓
Length	✓	✓	✓	✓	✓
Acousticness	×	✓	×	×	✓
Speechiness	✓	✓	✓	×	×

Figure 15. Comparison of variable selection for various models

7. FUTURE INVESTIGATION

While we were able to use various models to predict the popularity of a song, and get up to 80% or more accuracy in the prediction, there is a lot of room for improvement in our analysis. For future analysis, we could perform feature engineering by creating interaction features (such as energy*loudness, length*valence, etc), and checking if including such interaction terms in our model significantly improves the prediction accuracy. Similarly, given that some of the dependent variables were not perfectly fitting the normal distribution, methods such as Box-Cox transformation can be used to transform the dependent variables to a normal distribution. To further improve the models, the data can be scaled before implementing variable selection methods.

An important note is that any model is only as good as its underlying data. Hence, to further improve the analysis, it can be performed on a larger data set (compared to our current data size of around 2000 songs) to account for any bias in the current models.

Future analysis can involve delving deeper into some of the insights found for example, Year was not an obvious attribute, during the preliminary modeling stage for predicting a song’s accuracy. However, further investigation indicated that Year is correlated with other attributes in the dataset. Analyzing the trends on how beats per minute, danceability, loudness and length of songs have changed over the years 1956-2019, also proved to be useful as it provided further insight on how Year is an important factor for the models. For future work, dividing the dataset based on the song genres and further analysis of the impact of different attributes on the popularity based on genres. This could enable a more accurate song popularity model as songs belonging to a single genre might have more similarities than songs belonging to a different genre.

8. SUMMARY AND CONCLUSION

In this project, the top 2000 songs in Spotify from the years 1956-2019 are analyzed to predict the popularity of a song given different available attributes such as beats per minute, length, energy, etc. For the analysis, various regression models were created such as logistic regression, Lasso, random forest, etc, and identified that stepwise regression produces the best combination of prediction accuracy and explainability. Stepwise regression achieves an accuracy, sensitivity, and specificity of 0.82, 0.73, and 0.90. From various model selection methods, 5 attributes Year, Beats Per Minute, Danceability, Loudness and Length are the most important factors for predicting a song’s popularity.

Similarly, several ways were identified to further investigate the dataset to improve the model accuracy such as by using feature engineering and interaction terms, performing transformations before modeling. etc. Additionally, there might be other external factors that might be affecting whether a song gets popular or not. Hence, further improvements can be implemented in model and analysis to create accurate models.

APPENDIX

Song Popularity Prediction

Import Packages

```
library(car)
```

```
## Loading required package: carData
```

```
library(kknn)
library(superml)
```

```
## Loading required package: R6
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack
```

```
library(Hmisc)
```

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

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

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

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(caret)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

```
## The following object is masked from 'package:kknn':
##
```

```
##     contr.dummy
```

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

Data Loading

```
data <- read.csv("Spotify.csv")

encoder <- LabelEncoder$new()
data$Top.Genre <- encoder$fit_transform(data$Top.Genre)

data$Length <- as.integer(data$Length)
```

```
## Warning: NAs introduced by coercion
```

```
data <- na.omit(data)

data <- data[, 2:ncol(data)]

head(data)
```

```
##                                  Title            Artist Top.Genre Year
## 1                              Sunrise       Norah Jones         0 2004
## 2                          Black Night       Deep Purple         1 2000
## 3                       Clint Eastwood          Gorillaz         2 2001
## 4                        The Pretender      Foo Fighters         3 2007
## 5              Waitin' On A Sunny Day Bruce Springsteen         4 2002
## 6 The Road Ahead (Miles Of The Unknown)      City To City         5 2004
##   Beats.Per.Minute Energy Danceability Loudness Liveness Valence Length
## 1              157     30           53      -14       11      68    201
## 2              135     79           50      -11       17      81    207
## 3              168     69           66       -9        7      52    341
## 4              173     96           43       -4        3      37    269
## 5              106     82           58       -5       10      87    256
## 6               99     46           54       -9       14      14    247
##   Acousticness Speechiness Popularity
## 1           94           3         71
## 2           17           7         39
## 3            2          17         69
```

```
## 4           0           4         76
## 5           1           3         59
## 6           0           2         45
```

```r
cat("Number of instances in the dataset: ", nrow(data), "\n")
```

```
## Number of instances in the dataset:  1990
```

```r
cat("Number of features in the dataset: ", ncol(data))
```

```
## Number of features in the dataset:  14
```

Descriptive Statistics for Numerical Variables

```r
data.num <- data[, which(sapply(data, is.numeric))]

describe(data.num)
```

```
## data.num
##
##  12  Variables      1990  Observations
## --------------------------------------------------------------------------------
## Top.Genre
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0      149     0.99    27.74    34.96        0        1
##      .25      .50      .75      .90      .95
##        1       13       40       87      117
##
## lowest :   0   1   2   3   4, highest: 144 145 146 147 148
## --------------------------------------------------------------------------------
## Year
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       63        1     1993    18.57     1967     1970
##      .25      .50      .75      .90      .95
##     1979     1994     2007     2014     2017
##
## lowest : 1956 1958 1959 1960 1961, highest: 2015 2016 2017 2018 2019
## --------------------------------------------------------------------------------
## Beats.Per.Minute
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0      145        1    120.2    31.57     79.0     84.9
##      .25      .50      .75      .90      .95
##     99.0    119.0    136.0    162.0    174.0
##
## lowest :  37  49  54  58  60, highest: 200 203 204 205 206
## --------------------------------------------------------------------------------
## Energy
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       98        1    59.68    25.42       22       29
##      .25      .50      .75      .90      .95
##       42       61       78       89       93
##
## lowest :   3   4   5   6   7, highest:  96  97  98  99 100
## --------------------------------------------------------------------------------
## Danceability
```

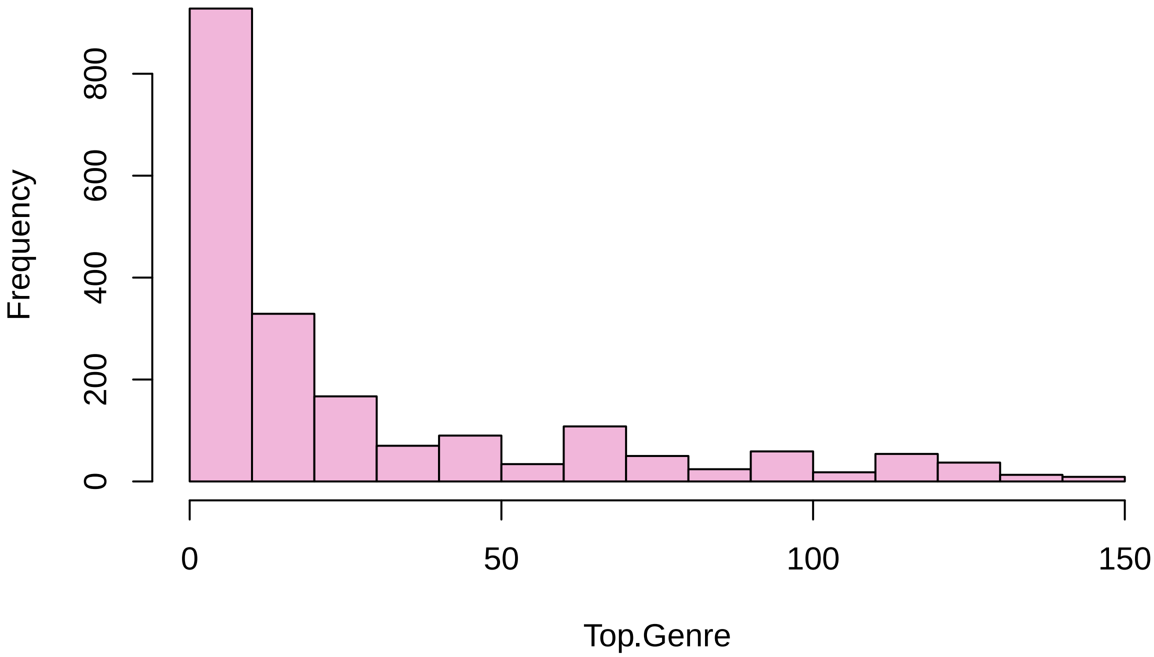
```
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       84        1    53.28    17.42       27       32
##      .25      .50      .75      .90      .95
##       43       53       64       73       79
##
## lowest : 10 12 14 15 16, highest: 91 92 93 95 96
## ---------------------------------------------------------------------------
## Loudness
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       23    0.992   -9.002    4.018      -16      -14
##      .25      .50      .75      .90      .95
##      -11       -8       -6       -5       -4
##
## lowest : -27 -24 -22 -21 -20, highest:  -6  -5  -4  -3  -2
## ---------------------------------------------------------------------------
## Liveness
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       94    0.997    18.98    15.29        6        7
##      .25      .50      .75      .90      .95
##        9       12       23       37       58
##
## lowest :  2  3  4  5  6, highest: 95 96 97 98 99
## ---------------------------------------------------------------------------
## Valence
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       97        1    49.45     28.6       13       18
##      .25      .50      .75      .90      .95
##       29       47       70       86       91
##
## lowest :  3  4  5  6  7, highest: 95 96 97 98 99
## ---------------------------------------------------------------------------
## Length
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0      346        1    260.4     80.8    162.4    181.0
##      .25      .50      .75      .90      .95
##    212.0    245.0    289.0    348.1    405.5
##
## lowest :  93 102 108 119 120, highest: 715 809 811 859 966
## ---------------------------------------------------------------------------
## Acousticness
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0      100    0.998    28.88    31.79        0        0
##      .25      .50      .75      .90      .95
##        3       18       50       75       86
##
## lowest :  0  1  2  3  4, highest: 95 96 97 98 99
## ---------------------------------------------------------------------------
## Speechiness
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       37      0.9    4.996    3.065        3        3
##      .25      .50      .75      .90      .95
##        3        4        5        8       12
##
## lowest :  2  3  4  5  6, highest: 39 41 44 46 55
```

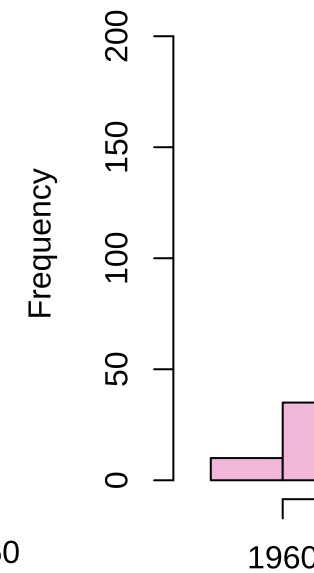
```
## --------------------------------------------------------------------------------
## Popularity
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1990        0       81        1    59.55    16.17       33       40
##      .25      .50      .75      .90      .95
##       50       62       71       76       79
##
## lowest :  11  12  13  14  15, highest:  87  88  95  98 100
## --------------------------------------------------------------------------------
```

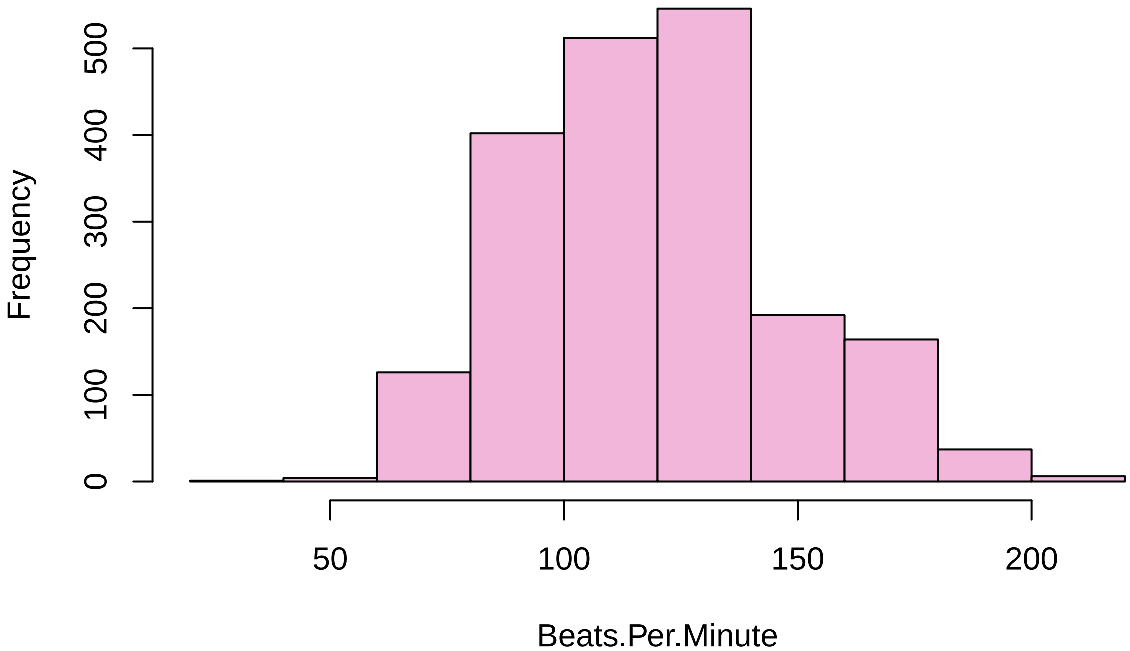
Exploratory Data Analysis

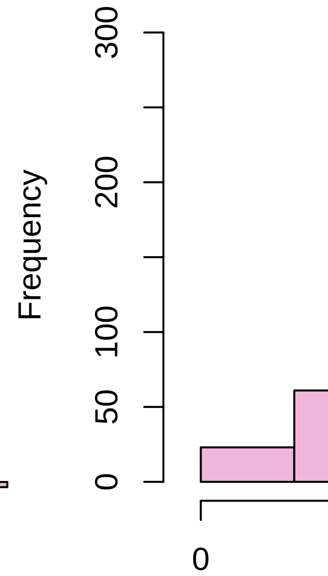
Distribution of Numerical Varibles: Histogram

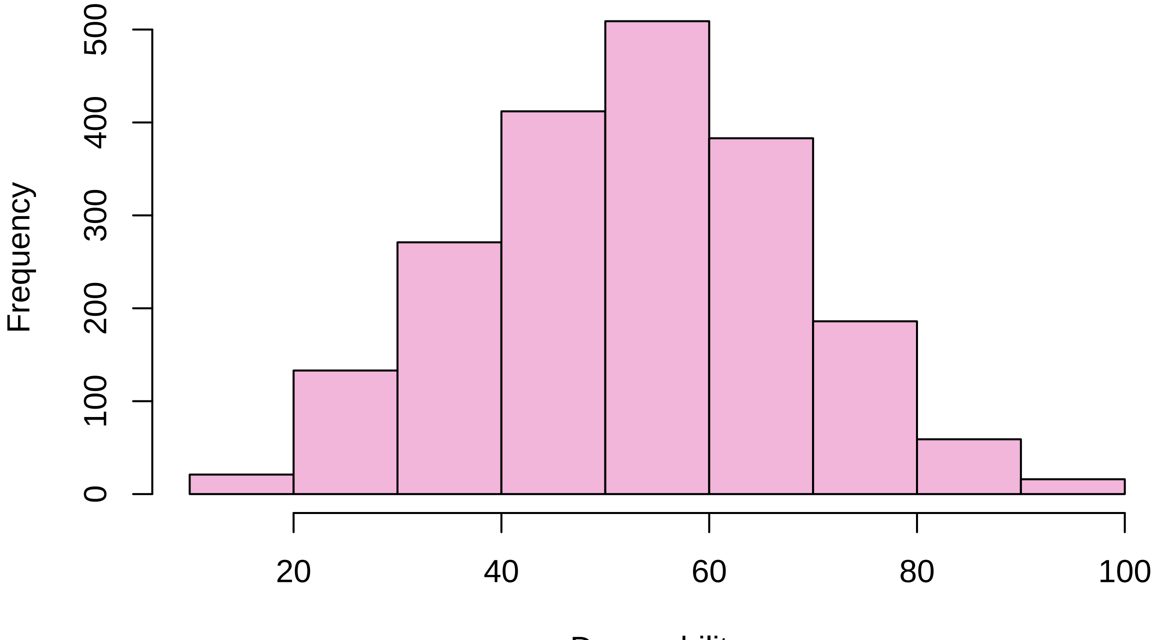
```r
for (i in 1:ncol(data.num)){
  hist(data.num[, i], main = "", xlab = colnames(data.num)[i], ylab = "Frequency", col = c("#F1B6DA"))
}
```

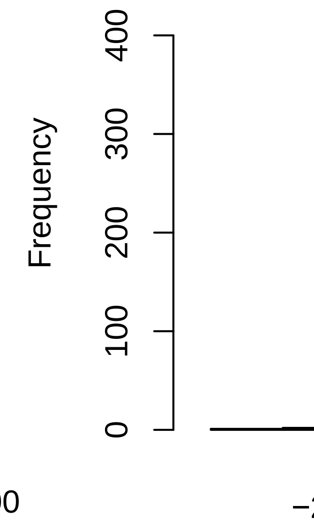


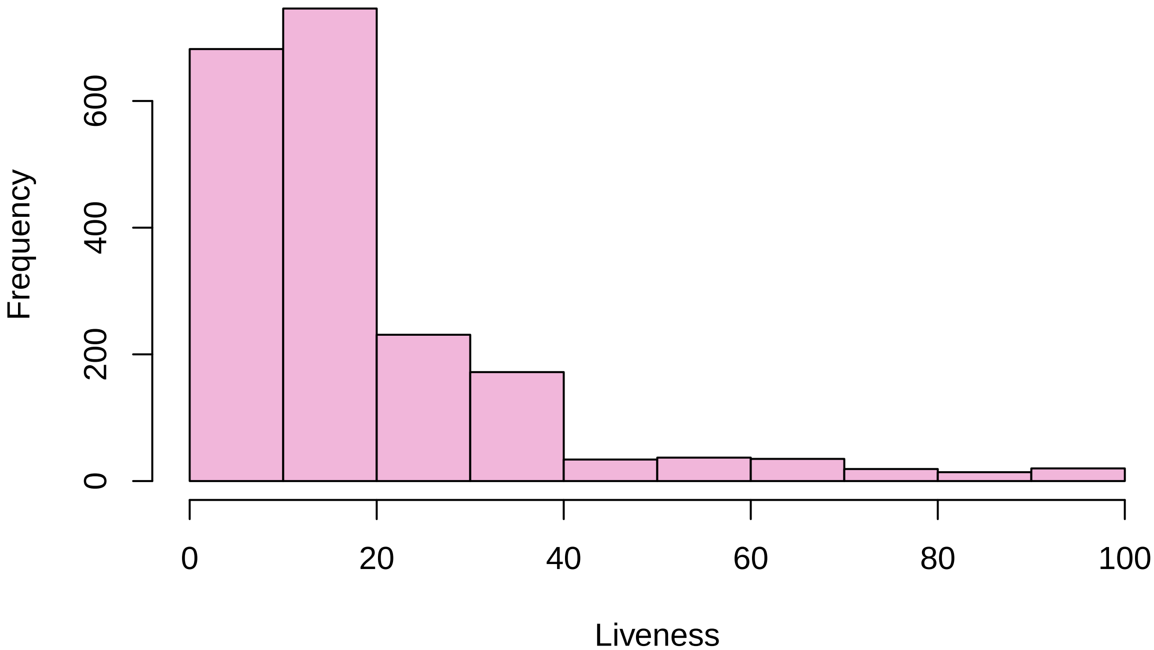


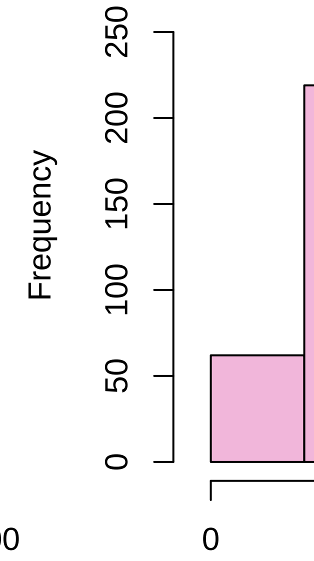

Frequency
0
100
200
300
400
500
50
100
150
200
Beats.Per.Minute

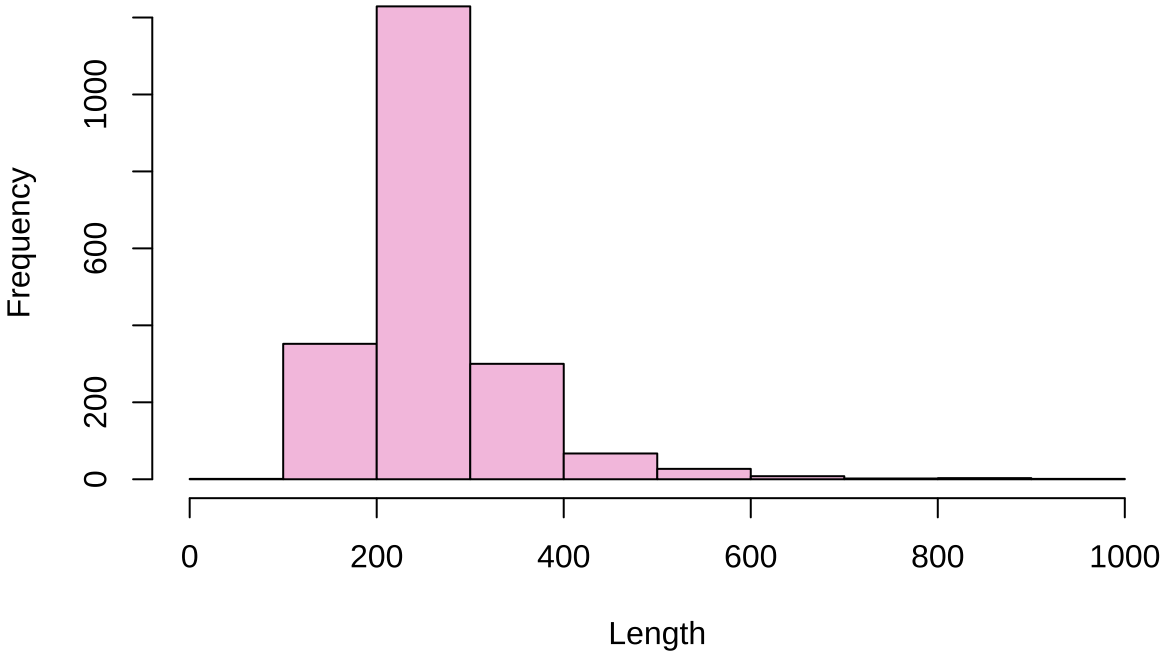

Frequency
0
50
100
200
300
0

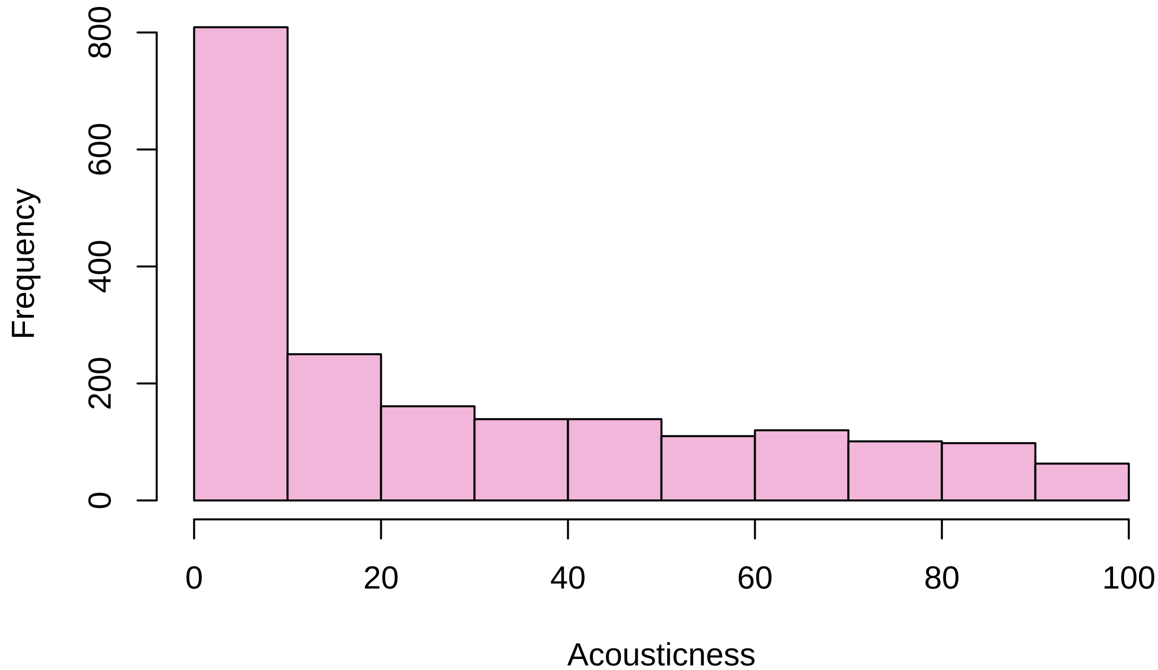

Frequency
0
100
200
300
400
500
20
40
60
80
100
Danceability

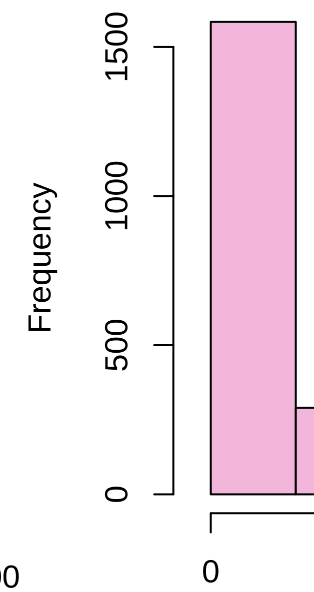

Frequency
0
100
200
300
400

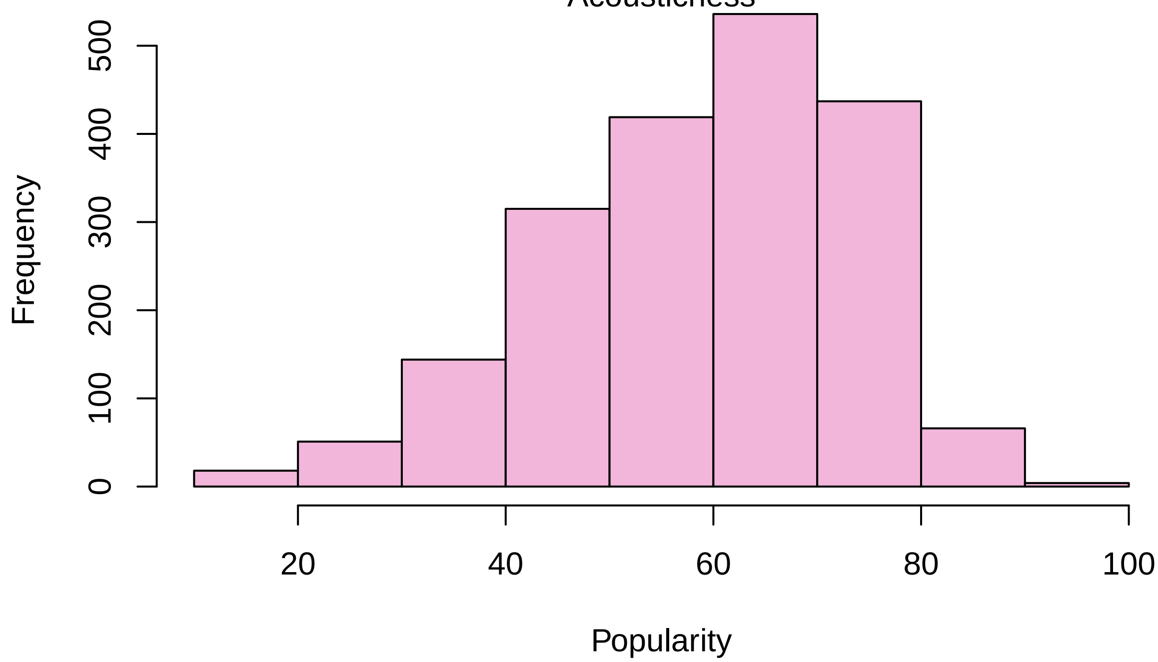

Frequency
0
200
400
600
0
20
40
60
80
100
Liveness


Frequency
0
50
100
150
200
250
0


Frequency
0
200
600
1000
0
200
400
600
800
1000
Length

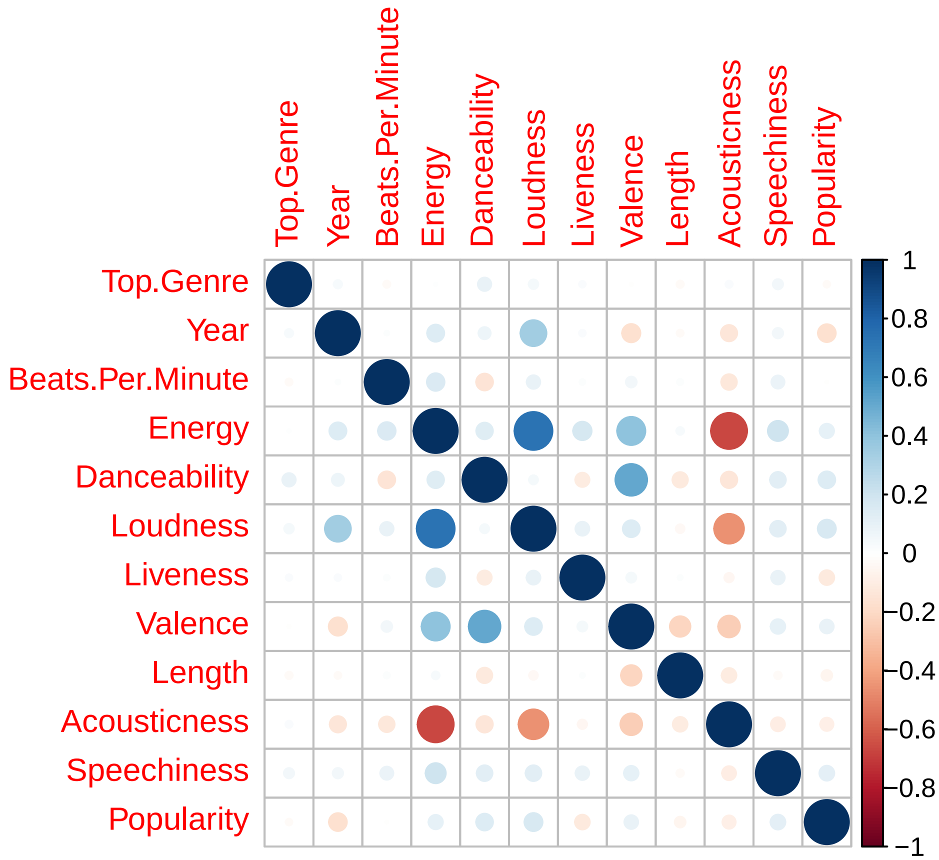






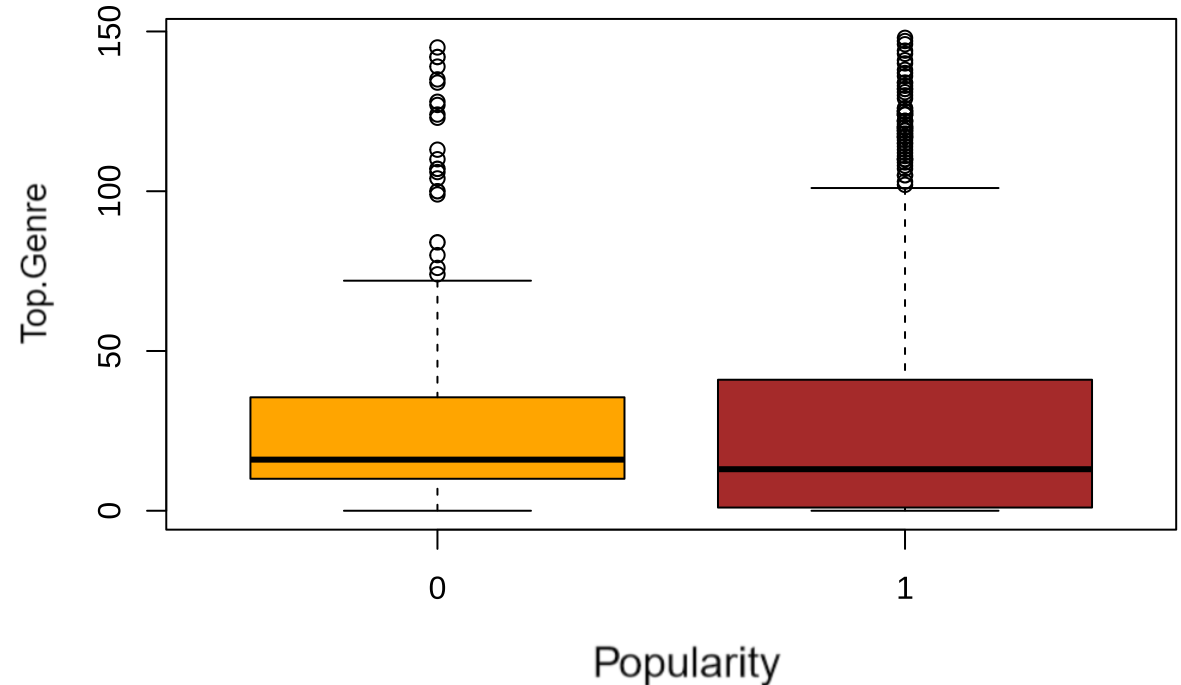
Correlation Matrix for Numerical Variables: Correlation Plot or Heatmap

```
corr <- cor(data.num)

corrplot(corr)
```

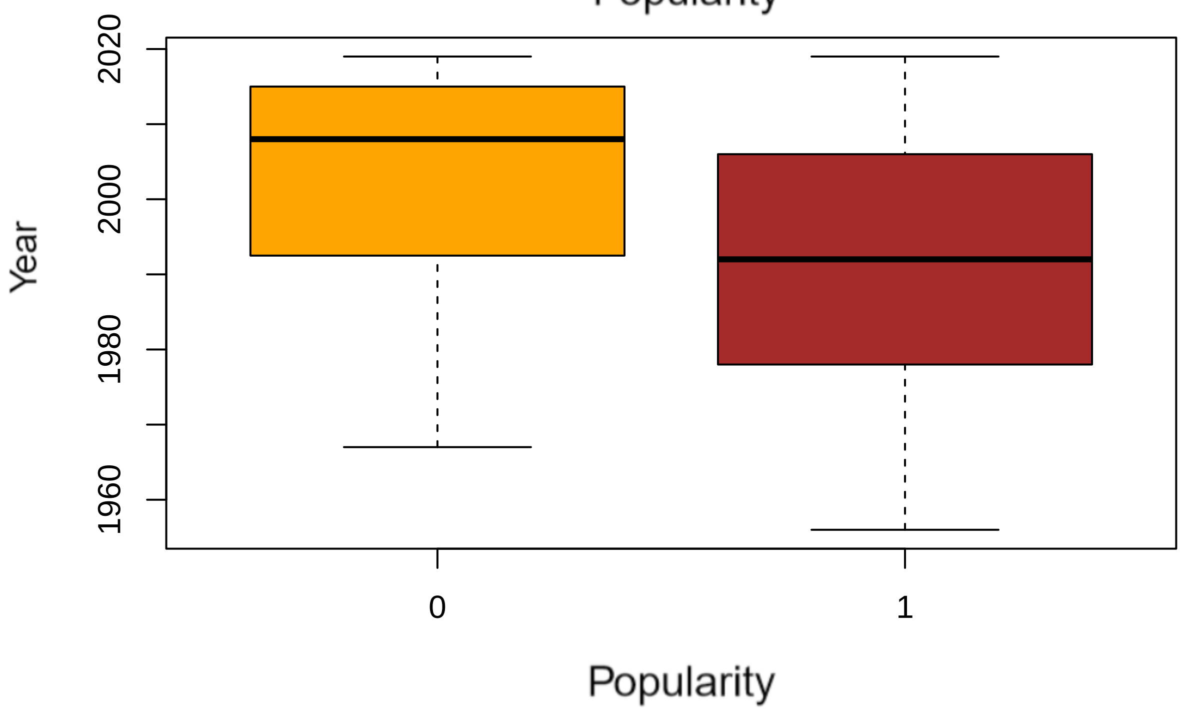


Relationship between Numerical Variables and the Binary Response Variable: Box Plot

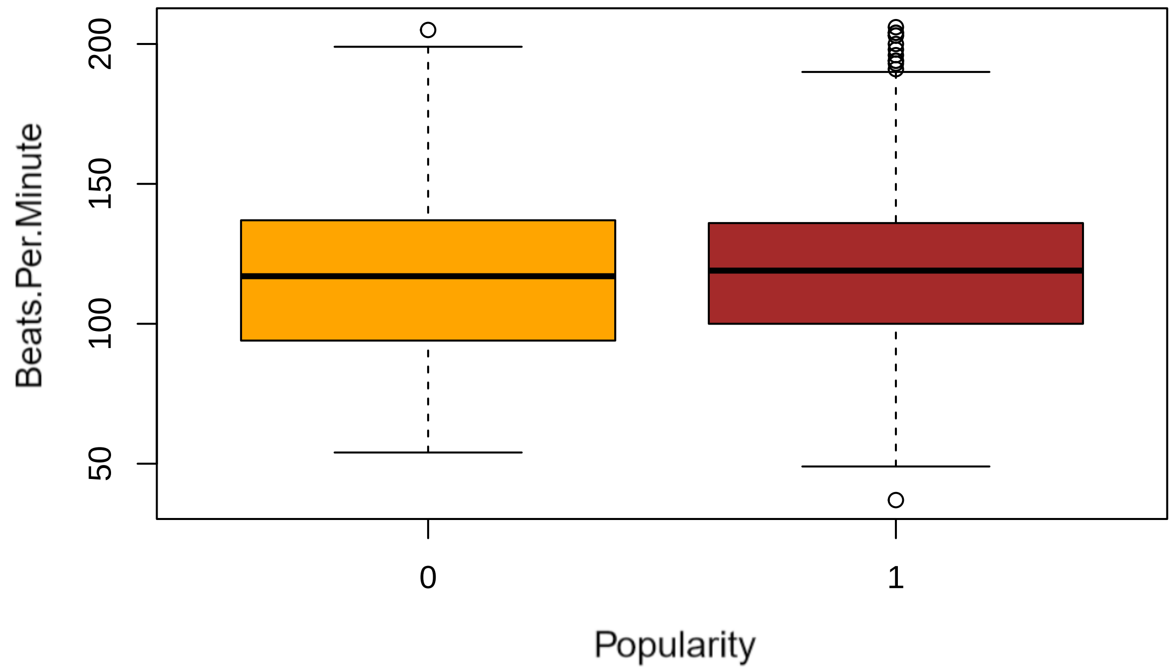
```
data$Popularity_bin <- ifelse(data$Popularity >= 40, 1, 0)
data.num$Popularity_bin <- ifelse(data$Popularity >= 40, 1, 0)

data.num_ <- downSample(data.num[, 1:(ncol(data.num) - 1)], as.factor(data.num[, ncol(data.num)]))

split <- 0.85 * nrow(data.num_)

for (i in 1:(ncol(data.num) - 2)){
  boxplot(data.num[, i] ~ data$Popularity_bin, main="", ylab=colnames(data.num)[i], xlab="Popularity",
  col=c("orange", "brown"), data=data.num)
}
```



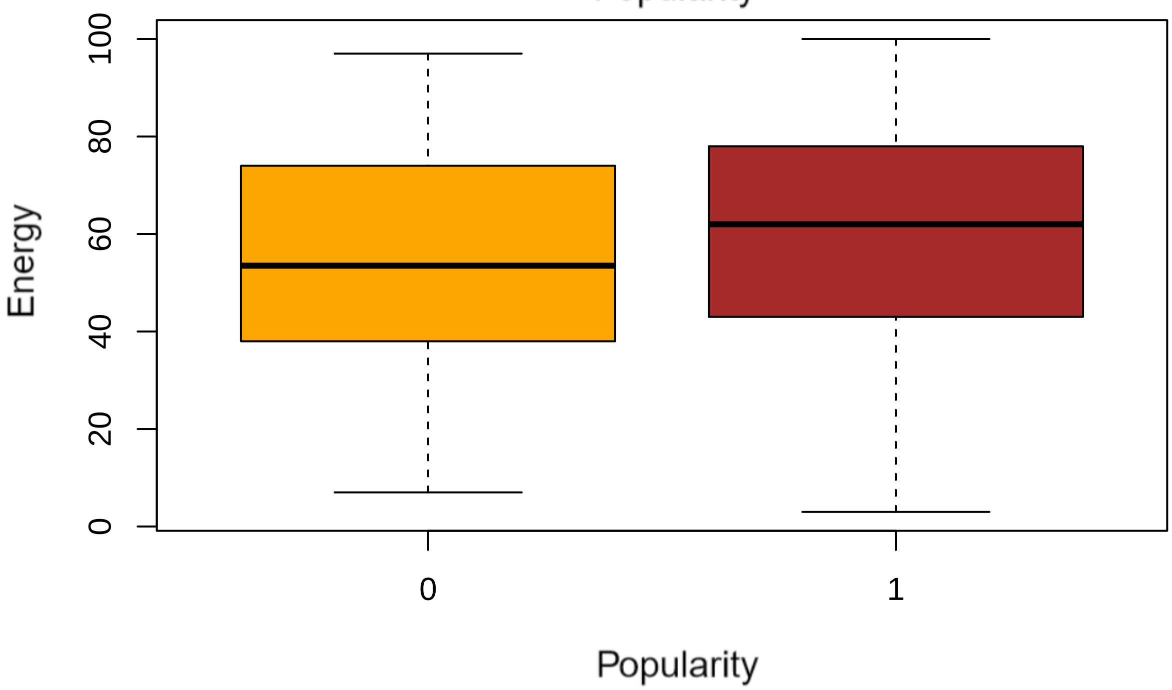
Top.Genre
0
50
100
150
0
1
Popularity



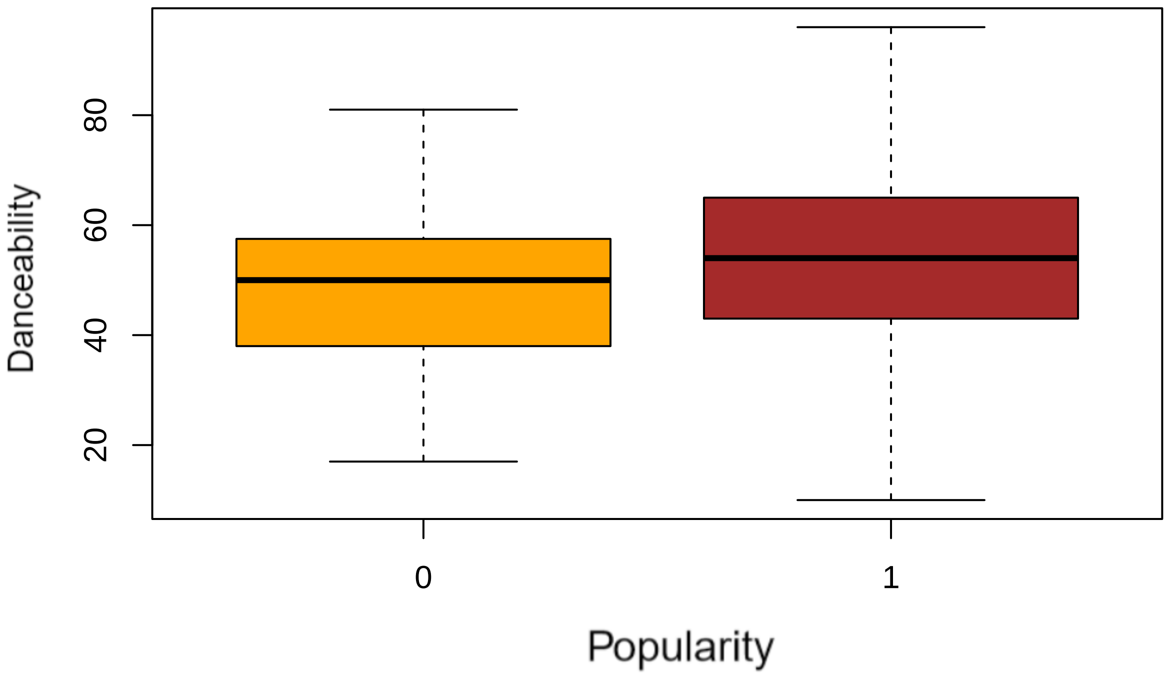
Year
1960
1980
2000
2020
0
1
Popularity



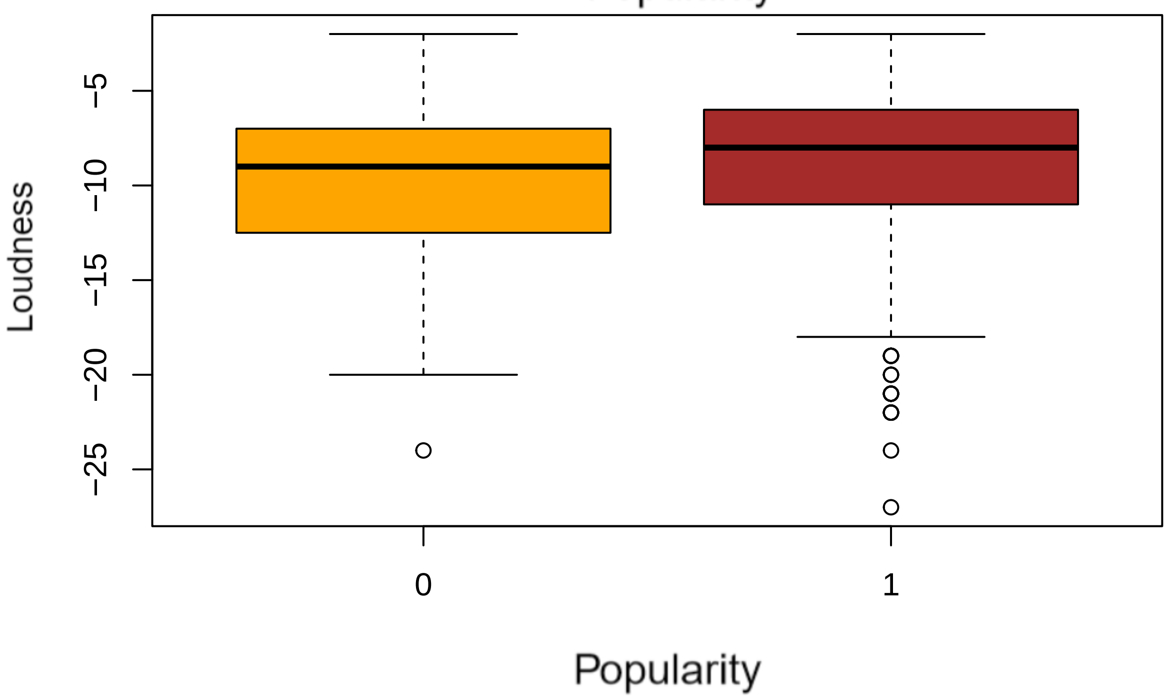
Beats.Per.Minute
200
150
100
50
0
1
Popularity



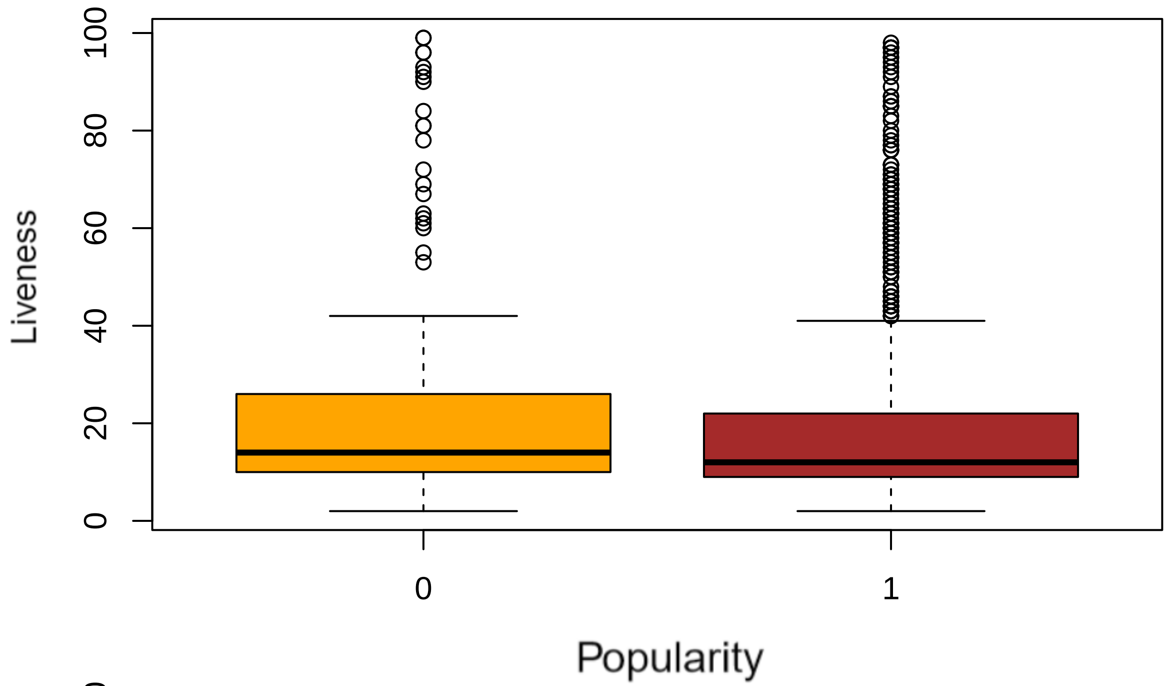
Energy
100
80
60
40
20
0
0
1
Popularity



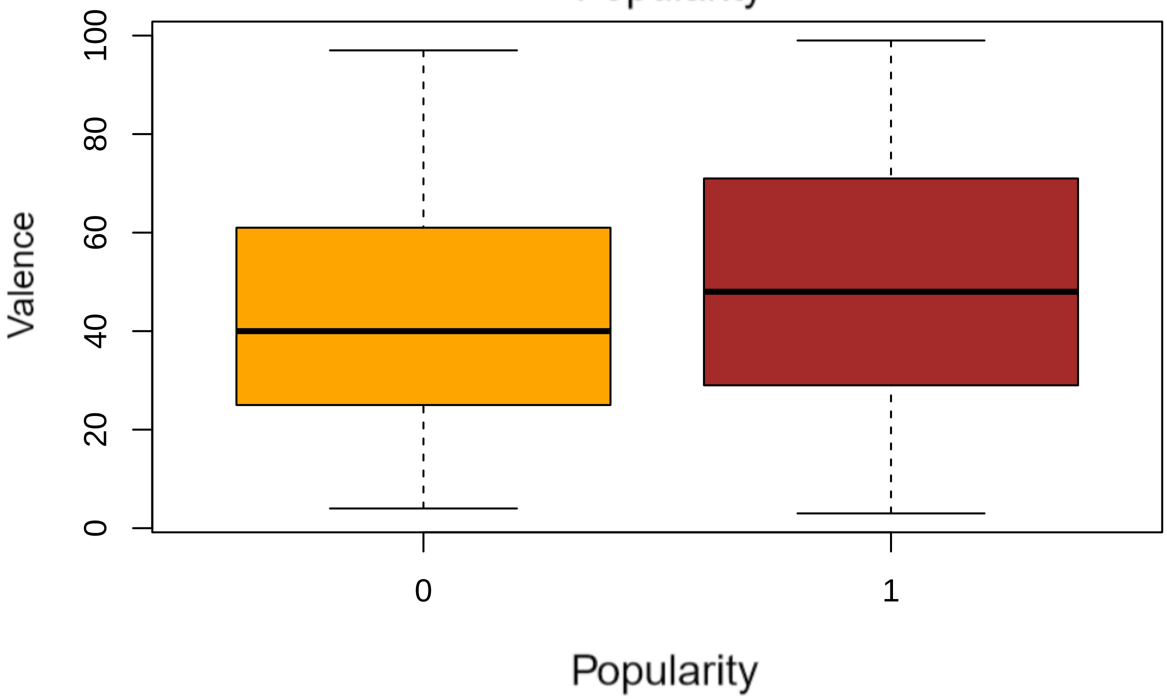
Danceability
80
60
40
20
0
1
Popularity



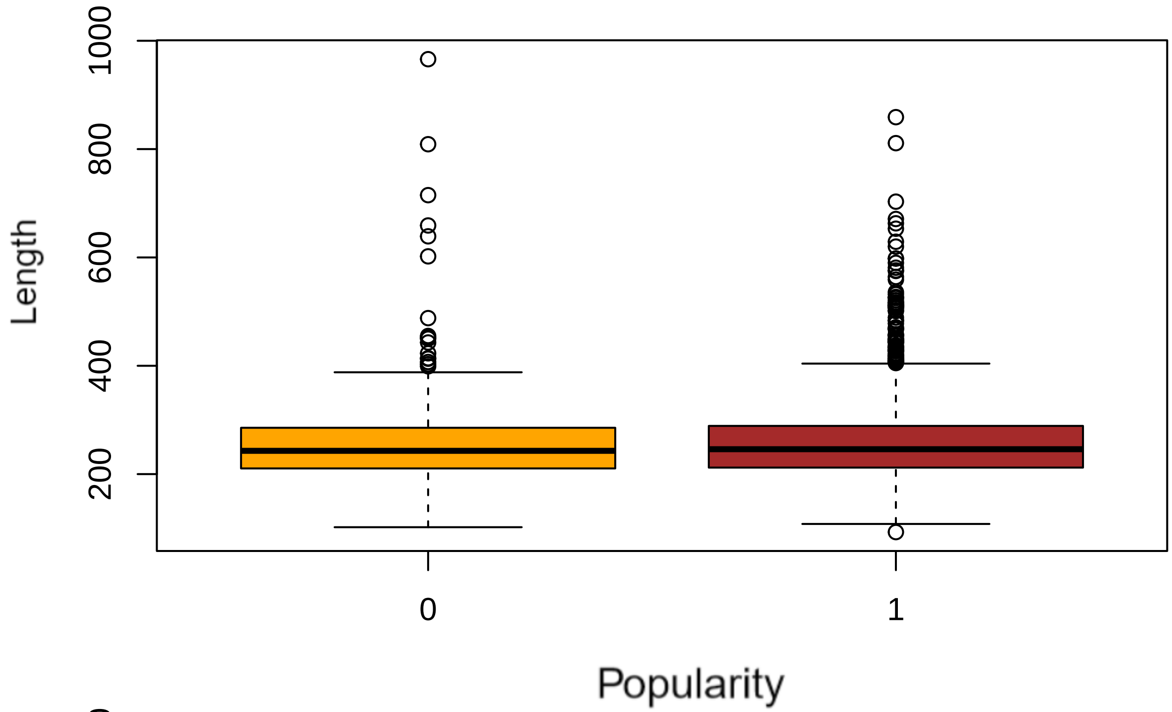
Loudness
−5
−10
−15
−20
−25
0
1
Popularity



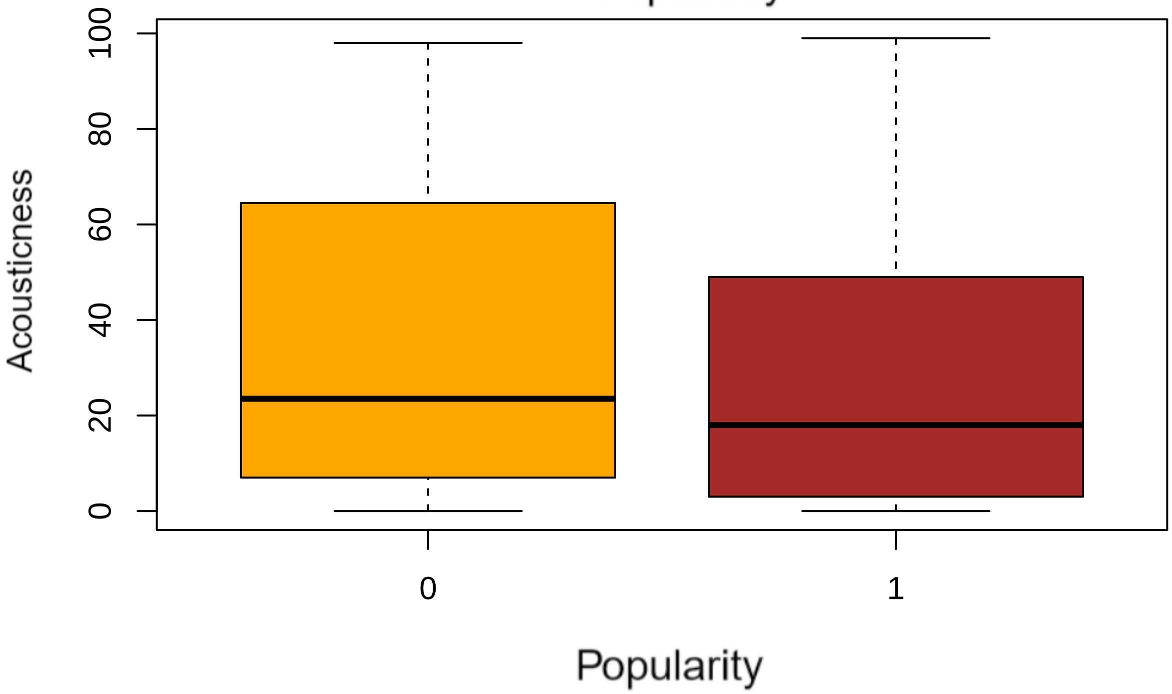
Liveness
100
80
60
40
20
0
0
1
Popularity



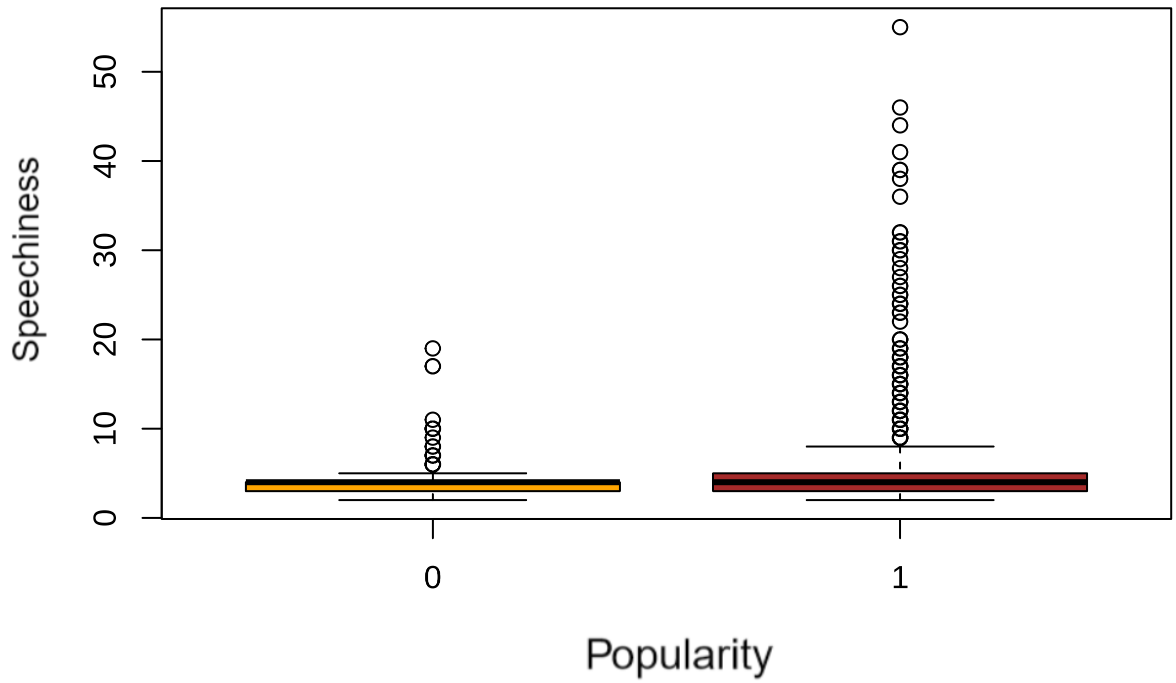
Valence
100
80
60
40
20
0
0
1
Popularity



Length
1000
800
600
400
200
0
1
Popularity



Acousticness
100
80
60
40
20
0
0
1
Popularity



Logistic Regression

```
data.log <- subset(data.num_, select = -c(Popularity))

data.log <- data.log[sample(1:nrow(data.log)), ]

train.data.log <- data.log[1:split, ]
test.data.log <- data.log[split+1:nrow(data.log), ]

logit.model <- glm(Class~ ., family = "binomial", data = train.data.log)

summary(logit.model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = train.data.log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9332  -0.9578  -0.3430   0.9846   2.2337
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      100.900192  19.180292   5.261 1.44e-07 ***
## Top.Genre         -0.004867   0.003568  -1.364   0.1725
## Year              -0.051419   0.009410  -5.465 4.64e-08 ***
## Beats.Per.Minute   0.006584   0.004536   1.451   0.1467
## Energy             0.002295   0.012282   0.187   0.8518
## Danceability       0.045594   0.011167   4.083 4.45e-05 ***
## Loudness           0.074032   0.061306   1.208   0.2272
## Liveness          -0.019382   0.008599  -2.254   0.0242 *
```

```
## Valence           -0.007584   0.007345  -1.033   0.3018
## Length            -0.001313   0.001377  -0.953   0.3404
## Acousticness       0.001465   0.005762   0.254   0.7993
## Speechiness        0.065682   0.047833   1.373   0.1697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 442.15  on 318  degrees of freedom
## Residual deviance: 364.50  on 307  degrees of freedom
## AIC: 388.5
##
## Number of Fisher Scoring iterations: 4
```

Logistic Regression: Multicollinearity

```r
vifs <- vif(logit.model)

vifs
```

```
##        Top.Genre             Year Beats.Per.Minute           Energy
##         1.038572         1.310275         1.126256         4.763514
##     Danceability         Loudness         Liveness          Valence
##         1.539187         3.321821         1.204219         1.975607
##           Length     Acousticness      Speechiness
##         1.158501         1.877285         1.212704
```

Logistic Regression: Prediction

```r
pred.log = predict(logit.model, newdata = test.data.log[, -ncol(test.data.log)], type = "response")

pred.log <- ifelse(pred.log >= 0.4, 1, 0)

head(pred.log)
```

```
## 102  63 122  86  83 320
##   0   0   0   1   0   0
```

Evaluation Function

```r
pred_metrics = function(modelName, actualClass, predClass) {
  cat(modelName, '\n')
  conmat <- confusionMatrix(table(actualClass, predClass))
  c(conmat$overall["Accuracy"], conmat$byClass["Sensitivity"],
  conmat$byClass["Specificity"])
}
```

Logistic Regression: Evaluation

```r
pred_metrics("Logistic Model", test.data.log[, ncol(test.data.log)], pred.log)
```

```
## Logistic Model

##    Accuracy Sensitivity Specificity
##   0.8070175   0.8000000   0.8125000
```

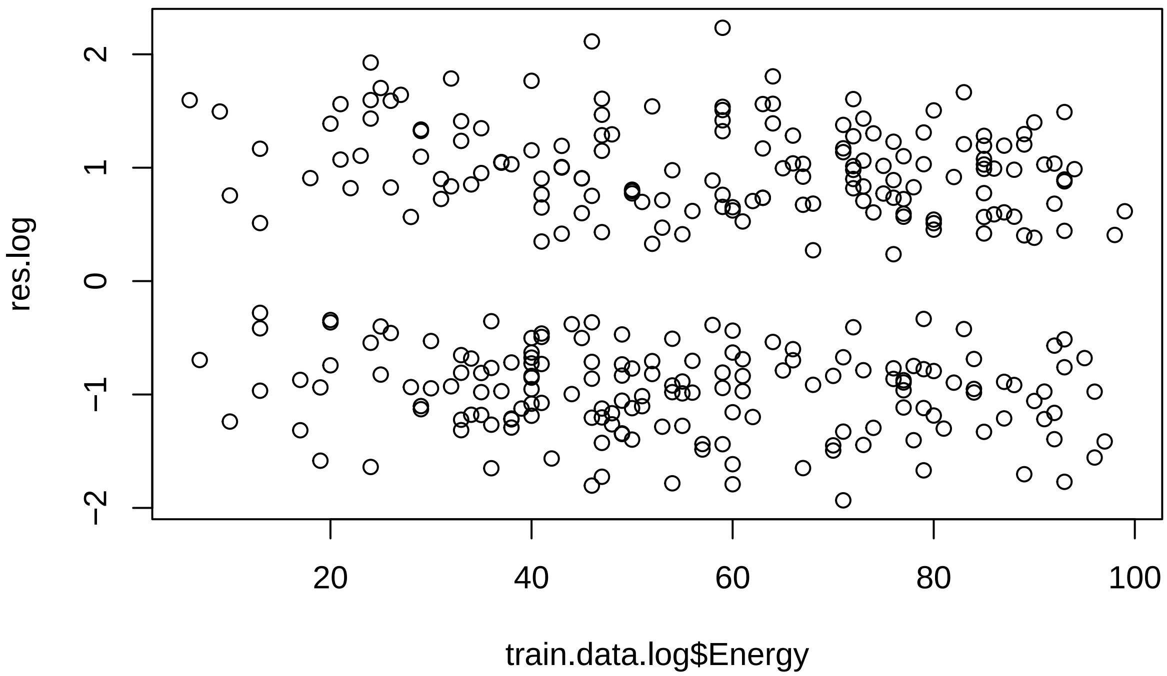
Logistic Regression: Goodness of Fit Test

```r
pearres.log = residuals(logit.model,type="pearson")

pearson.log = sum(pearres.log^2)

round(c(pearson.log, 1-pchisq(pearson.log, 307)), 2)
```

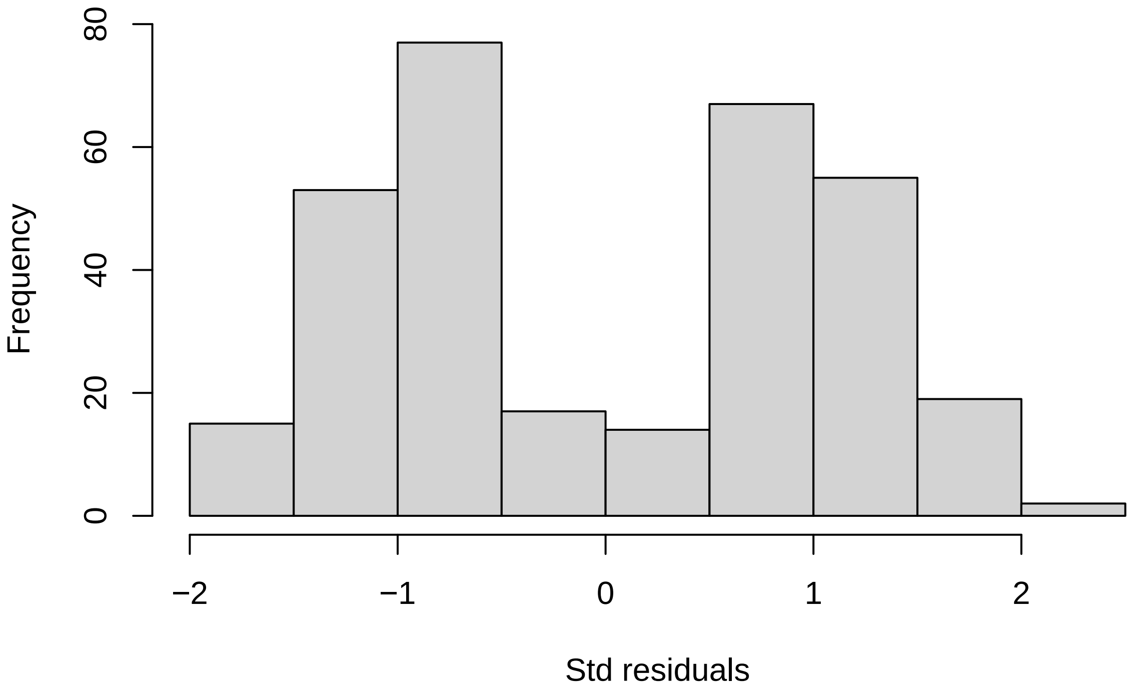
```
## [1] 308.65   0.46
```

Logistic Regression: Residual Analysis

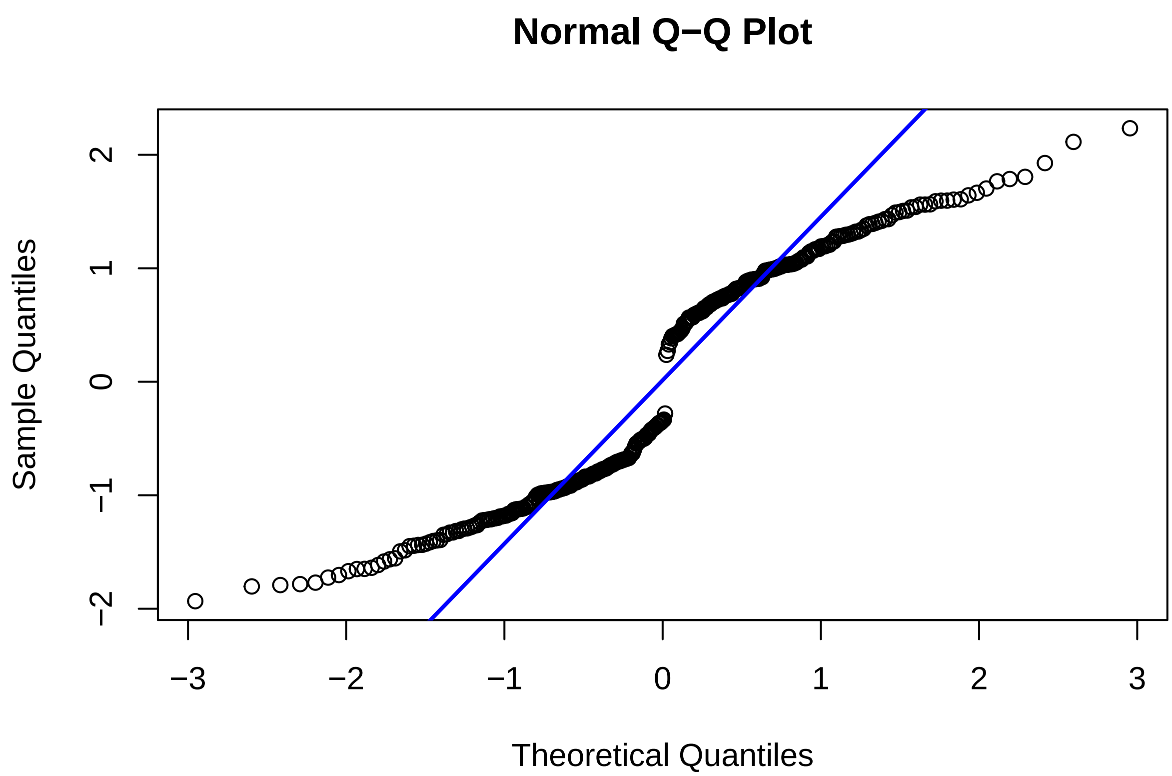
```r
res.log = resid(logit.model, type="deviance")

plot(train.data.log$Energy, res.log)
```



```r
hist(res.log, 10, xlab="Std residuals", main="")
```




```r
qqnorm(res.log)
qqline(res.log, col="blue", lwd=2)
```



Decision Tree

```r
dt <- rpart(Class ~ ., method = "class", data = train.data.log)

summary(dt)
```

```
## Call:
## rpart(formula = Class ~ ., data = train.data.log, method = "class")
##   n= 319
##
##           CP nsplit rel error    xerror       xstd
## 1 0.29936306      0 1.0000000 1.2165605 0.05576050
## 2 0.08917197      1 0.7006369 0.7579618 0.05501657
## 3 0.03184713      2 0.6114650 0.7070064 0.05418738
## 4 0.02229299      3 0.5796178 0.7515924 0.05492171
## 5 0.01910828      5 0.5350318 0.7834395 0.05537141
## 6 0.01273885     11 0.4140127 0.7643312 0.05510895
## 7 0.01000000     12 0.4012739 0.7834395 0.05537141
##
## Variable importance
##             Year           Length          Valence         Loudness
##               24               16               13               11
##           Energy     Acousticness         Liveness Beats.Per.Minute
##               10                9                6                6
##     Danceability
##                5
##
## Node number 1: 319 observations,    complexity param=0.2993631
##   predicted class=0  expected loss=0.492163  P(node) =1
##     class counts:   162   157
##    probabilities: 0.508 0.492
##   left son=2 (110 obs) right son=3 (209 obs)
##   Primary splits:
##       Year         < 2006.5 to the right, improve=17.536410, (0 missing)
##       Danceability < 57.5   to the left,  improve=12.861070, (0 missing)
##       Top.Genre    < 9.5    to the right, improve=11.112510, (0 missing)
##       Liveness     < 9.5    to the right, improve= 6.145657, (0 missing)
##       Valence      < 86.5   to the left,  improve= 5.289976, (0 missing)
##   Surrogate splits:
##       Danceability < 18.5   to the left,  agree=0.665, adj=0.027, (0 split)
##       Liveness     < 77.5   to the right, agree=0.661, adj=0.018, (0 split)
##
## Node number 2: 110 observations,    complexity param=0.01910828
##   predicted class=0  expected loss=0.2636364  P(node) =0.3448276
##     class counts:    81    29
##    probabilities: 0.736 0.264
##   left son=4 (39 obs) right son=5 (71 obs)
##   Primary splits:
##       Length       < 218.5  to the left,  improve=4.212883, (0 missing)
##       Danceability < 47.5   to the left,  improve=3.960553, (0 missing)
##       Year         < 2017.5 to the right, improve=1.995913, (0 missing)
##       Top.Genre    < 77     to the left,  improve=1.941414, (0 missing)
##       Liveness     < 23.5   to the right, improve=1.815538, (0 missing)
##   Surrogate splits:
```

```
##       Valence          < 56.5   to the right, agree=0.736, adj=0.256, (0 split)
##       Liveness         < 20.5   to the right, agree=0.673, adj=0.077, (0 split)
##       Year             < 2017.5 to the right, agree=0.664, adj=0.051, (0 split)
##       Beats.Per.Minute < 82.5   to the left,  agree=0.664, adj=0.051, (0 split)
##       Energy           < 75     to the right, agree=0.664, adj=0.051, (0 split)
##
## Node number 3: 209 observations,    complexity param=0.08917197
##   predicted class=1  expected loss=0.3875598  P(node) =0.6551724
##     class counts:    81   128
##    probabilities: 0.388 0.612
##   left son=6 (50 obs) right son=7 (159 obs)
##   Primary splits:
##       Valence      < 26.5   to the left,  improve=8.376569, (0 missing)
##       Danceability < 56.5   to the left,  improve=6.852674, (0 missing)
##       Top.Genre    < 9.5    to the right, improve=5.072751, (0 missing)
##       Liveness     < 10.5   to the right, improve=4.834359, (0 missing)
##       Length       < 382.5  to the right, improve=4.258092, (0 missing)
##   Surrogate splits:
##       Beats.Per.Minute < 75.5   to the left,  agree=0.780, adj=0.08, (0 split)
##       Danceability     < 32.5   to the left,  agree=0.780, adj=0.08, (0 split)
##       Energy           < 17.5   to the left,  agree=0.775, adj=0.06, (0 split)
##       Loudness         < -19.5  to the left,  agree=0.766, adj=0.02, (0 split)
##       Length           < 391    to the right, agree=0.766, adj=0.02, (0 split)
##
## Node number 4: 39 observations
##   predicted class=0  expected loss=0.07692308  P(node) =0.1222571
##     class counts:    36     3
##    probabilities: 0.923 0.077
##
## Node number 5: 71 observations,    complexity param=0.01910828
##   predicted class=0  expected loss=0.3661972  P(node) =0.2225705
##     class counts:    45    26
##    probabilities: 0.634 0.366
##   left son=10 (57 obs) right son=11 (14 obs)
##   Primary splits:
##       Acousticness < 4.5    to the right, improve=4.225917, (0 missing)
##       Danceability < 47.5   to the left,  improve=4.218385, (0 missing)
##       Year         < 2017.5 to the right, improve=1.972898, (0 missing)
##       Top.Genre    < 77     to the left,  improve=1.860972, (0 missing)
##       Energy       < 62.5   to the left,  improve=1.468858, (0 missing)
##   Surrogate splits:
##       Loudness < -5.5   to the left,  agree=0.845, adj=0.214, (0 split)
##       Energy   < 88     to the left,  agree=0.831, adj=0.143, (0 split)
##
## Node number 6: 50 observations,    complexity param=0.02229299
##   predicted class=0  expected loss=0.36  P(node) =0.1567398
##     class counts:    32    18
##    probabilities: 0.640 0.360
##   left son=12 (27 obs) right son=13 (23 obs)
##   Primary splits:
##       Loudness         < -11.5  to the right, improve=2.228406, (0 missing)
##       Year             < 2003.5 to the right, improve=2.109767, (0 missing)
##       Danceability     < 56.5   to the left,  improve=2.043322, (0 missing)
##       Liveness         < 12.5   to the right, improve=1.191369, (0 missing)
```

```
##       Beats.Per.Minute < 89.5   to the left,  improve=1.051905, (0 missing)
##   Surrogate splits:
##       Energy           < 31     to the right, agree=0.84, adj=0.652, (0 split)
##       Acousticness     < 66     to the left,  agree=0.80, adj=0.565, (0 split)
##       Year             < 1994.5 to the right, agree=0.68, adj=0.304, (0 split)
##       Beats.Per.Minute < 94.5   to the right, agree=0.66, adj=0.261, (0 split)
##       Length           < 288.5  to the right, agree=0.64, adj=0.217, (0 split)
##
## Node number 7: 159 observations,    complexity param=0.03184713
##   predicted class=1  expected loss=0.3081761  P(node) =0.4984326
##     class counts:    49   110
##    probabilities: 0.308 0.692
##   left son=14 (11 obs) right son=15 (148 obs)
##   Primary splits:
##       Length    < 382.5  to the right, improve=4.151322, (0 missing)
##       Liveness  < 10.5   to the right, improve=3.723345, (0 missing)
##       Loudness  < -12.5  to the left,  improve=3.674775, (0 missing)
##       Valence   < 71.5   to the left,  improve=3.352471, (0 missing)
##       Top.Genre < 123    to the right, improve=3.288808, (0 missing)
##
## Node number 10: 57 observations
##   predicted class=0  expected loss=0.2807018  P(node) =0.1786834
##     class counts:    41    16
##    probabilities: 0.719 0.281
##
## Node number 11: 14 observations
##   predicted class=1  expected loss=0.2857143  P(node) =0.04388715
##     class counts:     4    10
##    probabilities: 0.286 0.714
##
## Node number 12: 27 observations
##   predicted class=0  expected loss=0.2222222  P(node) =0.0846395
##     class counts:    21     6
##    probabilities: 0.778 0.222
##
## Node number 13: 23 observations,    complexity param=0.02229299
##   predicted class=1  expected loss=0.4782609  P(node) =0.07210031
##     class counts:    11    12
##    probabilities: 0.478 0.522
##   left son=26 (8 obs) right son=27 (15 obs)
##   Primary splits:
##       Energy           < 19.5   to the left,  improve=3.861594, (0 missing)
##       Top.Genre        < 9.5    to the right, improve=1.278261, (0 missing)
##       Acousticness     < 65.5   to the right, improve=1.278261, (0 missing)
##       Year             < 1994.5 to the right, improve=1.121118, (0 missing)
##       Beats.Per.Minute < 92     to the left,  improve=1.121118, (0 missing)
##   Surrogate splits:
##       Beats.Per.Minute < 87     to the left,  agree=0.826, adj=0.500, (0 split)
##       Acousticness     < 89.5   to the right, agree=0.826, adj=0.500, (0 split)
##       Loudness         < -19.5  to the left,  agree=0.783, adj=0.375, (0 split)
##       Danceability     < 24     to the left,  agree=0.739, adj=0.250, (0 split)
##       Year             < 1994.5 to the right, agree=0.696, adj=0.125, (0 split)
##
## Node number 14: 11 observations
```

```
##   predicted class=0  expected loss=0.2727273  P(node) =0.03448276
##     class counts:     8     3
##    probabilities: 0.727 0.273
##
## Node number 15: 148 observations,    complexity param=0.01910828
##   predicted class=1  expected loss=0.277027  P(node) =0.4639498
##     class counts:    41   107
##    probabilities: 0.277 0.723
##   left son=30 (88 obs) right son=31 (60 obs)
##   Primary splits:
##       Liveness  < 10.5   to the right, improve=3.256511, (0 missing)
##       Top.Genre < 124.5  to the right, improve=2.809620, (0 missing)
##       Year      < 1970.5 to the right, improve=2.753481, (0 missing)
##       Valence   < 72     to the left,  improve=2.727595, (0 missing)
##       Loudness  < -12.5  to the left,  improve=2.378727, (0 missing)
##   Surrogate splits:
##       Danceability < 66.5   to the left,  agree=0.703, adj=0.267, (0 split)
##       Year         < 1968.5 to the right, agree=0.628, adj=0.083, (0 split)
##       Energy       < 18.5   to the right, agree=0.628, adj=0.083, (0 split)
##       Valence      < 81.5   to the left,  agree=0.615, adj=0.050, (0 split)
##       Acousticness < 87.5   to the left,  agree=0.615, adj=0.050, (0 split)
##
## Node number 26: 8 observations
##   predicted class=0  expected loss=0.125  P(node) =0.02507837
##     class counts:     7     1
##    probabilities: 0.875 0.125
##
## Node number 27: 15 observations
##   predicted class=1  expected loss=0.2666667  P(node) =0.04702194
##     class counts:     4    11
##    probabilities: 0.267 0.733
##
## Node number 30: 88 observations,    complexity param=0.01910828
##   predicted class=1  expected loss=0.3636364  P(node) =0.2758621
##     class counts:    32    56
##    probabilities: 0.364 0.636
##   left son=60 (38 obs) right son=61 (50 obs)
##   Primary splits:
##       Length    < 232    to the left,  improve=2.487273, (0 missing)
##       Valence   < 44     to the right, improve=2.437618, (0 missing)
##       Loudness  < -12.5  to the left,  improve=2.125781, (0 missing)
##       Year      < 1991.5 to the left,  improve=1.113070, (0 missing)
##       Top.Genre < 9.5    to the right, improve=1.084571, (0 missing)
##   Surrogate splits:
##       Top.Genre        < 30     to the right, agree=0.636, adj=0.158, (0 split)
##       Year             < 1972.5 to the left,  agree=0.636, adj=0.158, (0 split)
##       Energy           < 38     to the left,  agree=0.625, adj=0.132, (0 split)
##       Beats.Per.Minute < 152.5  to the right, agree=0.614, adj=0.105, (0 split)
##       Loudness         < -14.5  to the left,  agree=0.614, adj=0.105, (0 split)
##
## Node number 31: 60 observations
##   predicted class=1  expected loss=0.15  P(node) =0.1880878
##     class counts:     9    51
##    probabilities: 0.150 0.850
```

```
##
## Node number 60: 38 observations,    complexity param=0.01910828
##   predicted class=0  expected loss=0.5  P(node) =0.1191223
##     class counts:    19    19
##    probabilities: 0.500 0.500
##   left son=120 (21 obs) right son=121 (17 obs)
##   Primary splits:
##       Length       < 202.5  to the right, improve=4.310924, (0 missing)
##       Danceability < 59.5   to the right, improve=3.567050, (0 missing)
##       Year         < 1976   to the right, improve=3.134680, (0 missing)
##       Energy       < 35.5   to the right, improve=2.850000, (0 missing)
##       Valence      < 43     to the right, improve=2.192308, (0 missing)
##   Surrogate splits:
##       Liveness         < 19.5   to the left,  agree=0.711, adj=0.353, (0 split)
##       Year             < 1979.5 to the right, agree=0.684, adj=0.294, (0 split)
##       Beats.Per.Minute < 115.5  to the left,  agree=0.658, adj=0.235, (0 split)
##       Danceability     < 42.5   to the left,  agree=0.658, adj=0.235, (0 split)
##       Loudness         < -13.5  to the right, agree=0.658, adj=0.235, (0 split)
##
## Node number 61: 50 observations,    complexity param=0.01910828
##   predicted class=1  expected loss=0.26  P(node) =0.1567398
##     class counts:    13    37
##    probabilities: 0.260 0.740
##   left son=122 (8 obs) right son=123 (42 obs)
##   Primary splits:
##       Loudness         < -12.5  to the left,  improve=4.573333, (0 missing)
##       Year             < 1990   to the left,  improve=2.246494, (0 missing)
##       Valence          < 73     to the left,  improve=1.690000, (0 missing)
##       Beats.Per.Minute < 131.5  to the left,  improve=1.601905, (0 missing)
##       Energy           < 56.5   to the left,  improve=1.186524, (0 missing)
##   Surrogate splits:
##       Year   < 1975.5 to the left,  agree=0.88, adj=0.25, (0 split)
##       Energy < 27     to the left,  agree=0.88, adj=0.25, (0 split)
##
## Node number 120: 21 observations,    complexity param=0.01273885
##   predicted class=0  expected loss=0.2857143  P(node) =0.06583072
##     class counts:    15     6
##    probabilities: 0.714 0.286
##   left son=240 (13 obs) right son=241 (8 obs)
##   Primary splits:
##       Valence          < 43     to the right, improve=2.9752750, (0 missing)
##       Danceability     < 43     to the right, improve=2.2936510, (0 missing)
##       Beats.Per.Minute < 101.5  to the right, improve=1.1868130, (0 missing)
##       Acousticness     < 18     to the left,  improve=0.7936508, (0 missing)
##       Energy           < 76     to the right, improve=0.4285714, (0 missing)
##   Surrogate splits:
##       Year             < 1969.5 to the right, agree=0.762, adj=0.375, (0 split)
##       Energy           < 56.5   to the right, agree=0.762, adj=0.375, (0 split)
##       Danceability     < 43     to the right, agree=0.762, adj=0.375, (0 split)
##       Beats.Per.Minute < 85     to the right, agree=0.714, adj=0.250, (0 split)
##       Acousticness     < 39     to the left,  agree=0.714, adj=0.250, (0 split)
##
## Node number 121: 17 observations
##   predicted class=1  expected loss=0.2352941  P(node) =0.05329154
```

```
##     class counts:     4    13
##    probabilities: 0.235 0.765
##
## Node number 122: 8 observations
##   predicted class=0  expected loss=0.25  P(node) =0.02507837
##     class counts:     6     2
##    probabilities: 0.750 0.250
##
## Node number 123: 42 observations
##   predicted class=1  expected loss=0.1666667  P(node) =0.1316614
##     class counts:     7    35
##    probabilities: 0.167 0.833
##
## Node number 240: 13 observations
##   predicted class=0  expected loss=0.07692308  P(node) =0.04075235
##     class counts:    12     1
##    probabilities: 0.923 0.077
##
## Node number 241: 8 observations
##   predicted class=1  expected loss=0.375  P(node) =0.02507837
##     class counts:     3     5
##    probabilities: 0.375 0.625
```

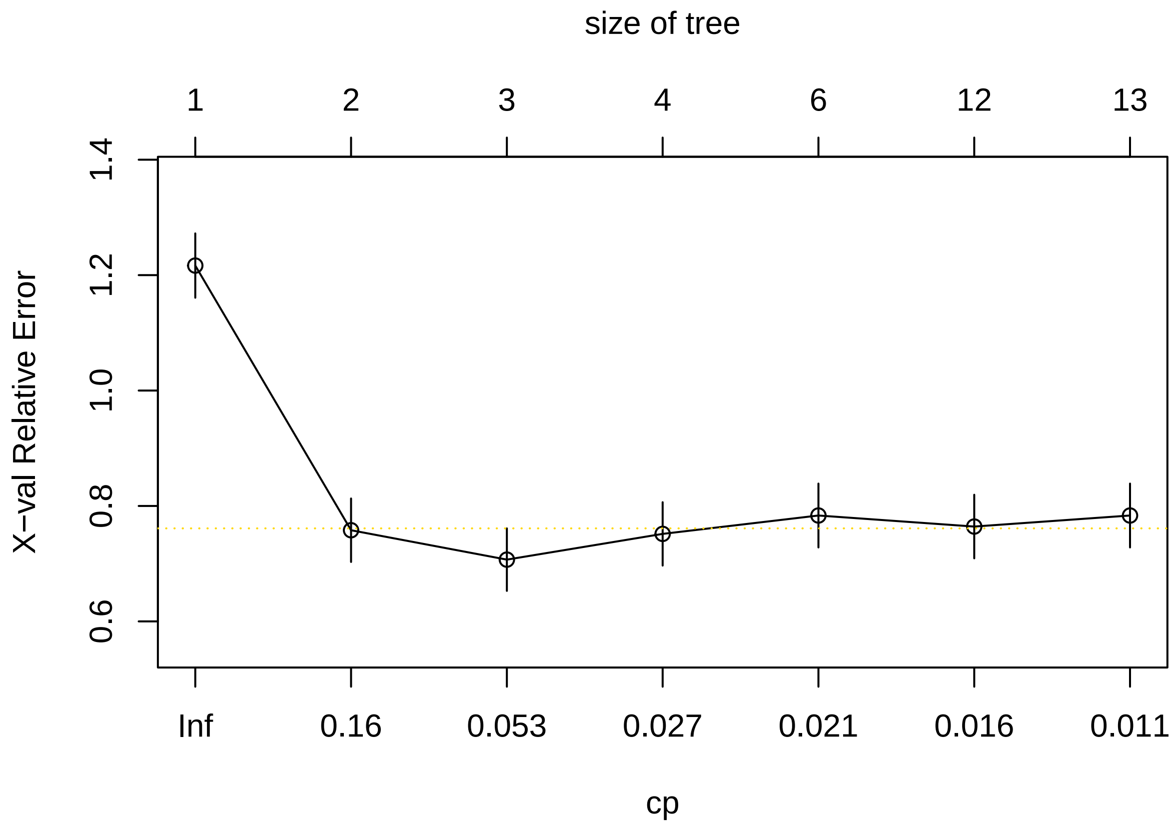
Decision Tree: Cross-Validation Table

```
printcp(dt)
```

```
##
## Classification tree:
## rpart(formula = Class ~ ., data = train.data.log, method = "class")
##
## Variables actually used in tree construction:
## [1] Acousticness Energy       Length       Liveness     Loudness
## [6] Valence      Year
##
## Root node error: 157/319 = 0.49216
##
## n= 319
##
##         CP nsplit rel error  xerror     xstd
## 1 0.299363      0   1.00000 1.21656 0.055761
## 2 0.089172      1   0.70064 0.75796 0.055017
## 3 0.031847      2   0.61146 0.70701 0.054187
## 4 0.022293      3   0.57962 0.75159 0.054922
## 5 0.019108      5   0.53503 0.78344 0.055371
## 6 0.012739     11   0.41401 0.76433 0.055109
## 7 0.010000     12   0.40127 0.78344 0.055371
```

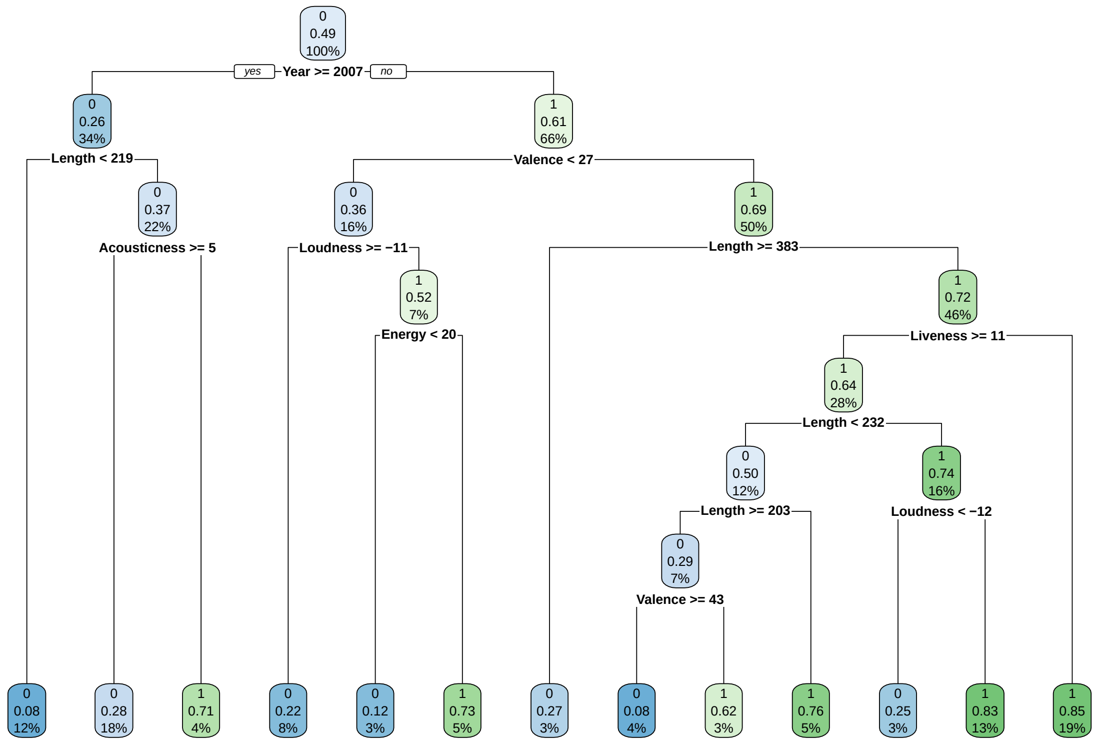
Decision Tree: Complexity Parameters

```
plotcp(dt, minline = TRUE, lty = 3,col = "gold")
```



Decision Tree: Visualization

```
rpart.plot(dt)
```



Decision Tree: Prediction

```r
pred.dt <- predict(dt, test.data.log[, -ncol(test.data.log)], type="class")

head(pred.dt)
```

```
## 102  63 122  86  83 320
##   0   0   1   1   0   0
## Levels: 0 1
```

Decision Tree: Evaluation

```r
pred_metrics("Decision Tree", pred.dt, test.data.log[, ncol(test.data.log)])
```

```
## Decision Tree
```

```
##    Accuracy Sensitivity Specificity
##   0.6315789   0.6923077   0.5806452
```

Stepwise Regression

```
min.model <- glm(Class~ 1, family = "binomial", data = train.data.log)

step.model <- step(min.model, scope = list(lower = min.model, upper = logit.model), direction = "both",

summary(step.model)
```

```
##
## Call:
## glm(formula = Class ~ Year + Danceability + Liveness + Loudness +
##     Speechiness, family = "binomial", data = train.data.log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9913  -0.9561  -0.3836   0.9708   2.0938
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  95.275223  17.329204   5.498 3.84e-08 ***
## Year         -0.048301   0.008633  -5.595 2.21e-08 ***
## Danceability  0.035956   0.009066   3.966 7.31e-05 ***
## Liveness     -0.020412   0.008438  -2.419   0.0156 *
## Loudness      0.070569   0.035346   1.997   0.0459 *
## Speechiness   0.071820   0.049310   1.456   0.1453
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 442.15  on 318  degrees of freedom
## Residual deviance: 369.72  on 313  degrees of freedom
## AIC: 381.72
##
## Number of Fisher Scoring iterations: 4
```

Stepwise Regression: Prediction

```
pred.step = predict(step.model, newdata = test.data.log[, -ncol(test.data.log)], type = "response")

pred.step <- ifelse(pred.step >= 0.4, 1, 0)

head(pred.step)
```

```
## 102  63 122  86  83 320
##   0   0   0   1   1   0
```

Stepwise Regression: Evaluation

```
pred_metrics("Stepwise Regression", pred.step, test.data.log[, ncol(test.data.log)])
```

```
## Stepwise Regression
```

```
##    Accuracy Sensitivity Specificity
##   0.8245614   0.7307692   0.9032258
```

Random Forest

```r
rf <- randomForest(Class~., data = train.data.log)

summary(rf)
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted        319   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes            638   matrix numeric
## oob.times        319   -none- numeric
## classes            2   -none- character
## importance        11   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                319   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

Random Forest: Prediction

```r
pred.rf <- predict(rf, test.data.log[, -ncol(test.data.log)], type="class")

head(pred.rf)
```

```
## 102  63 122  86  83 320
##   0   0   0   1   1   0
## Levels: 0 1
```

Random Forest: Evaluation

```r
pred_metrics("Random Forest", pred.rf, test.data.log[, ncol(test.data.log)])
```

```
## Random Forest
```

```
##    Accuracy Sensitivity Specificity
##   0.7719298   0.7692308   0.7741935
```

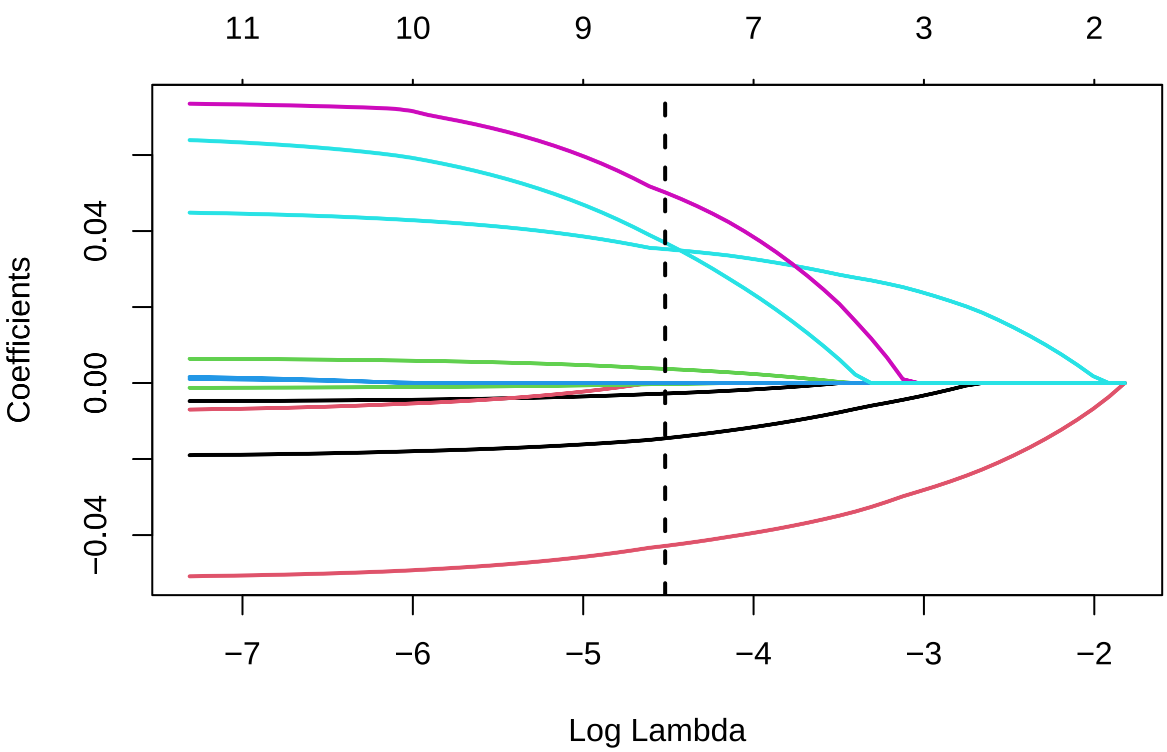
Lasso Regression

```
cv.lasso <- cv.glmnet(as.matrix(train.data.log[, 1:(ncol(train.data.log) - 1)]), train.data.log[, ncol(t

lasso.model <- glmnet(as.matrix(train.data.log[, 1:(ncol(train.data.log) - 1)]), train.data.log[, ncol(t

coef(lasso.model, s=cv.lasso$lambda.min)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)      83.9190880464
## Top.Genre        -0.0027501889
## Year             -0.0428256336
## Beats.Per.Minute  0.0037381507
## Energy            .
## Danceability      0.0352363167
## Loudness          0.0501544287
## Liveness         -0.0145192863
## Valence           .
## Length           -0.0002564901
## Acousticness      .
## Speechiness       0.0369238388
```

Lasso Regression: Visualization

```
plot(lasso.model, xvar="lambda", lwd=2)

abline(v=log(cv.lasso$lambda.min), col='black', lty=2, lwd=2)
```



Lasso Regression: Prediction

```
pred.lasso <- predict(cv.lasso, as.matrix(test.data.log[, -ncol(test.data.log)]), type="class")

head(pred.lasso)
```

```
##     lambda.1se
## 102 "0"
## 63  "0"
## 122 "0"
## 86  "1"
## 83  "0"
## 320 "0"
```

Lasso Regression: Evaluation

```
pred_metrics("Lasso Regression Model", test.data.log[, ncol(test.data.log)], pred.lasso)
```

```
## Lasso Regression Model

##    Accuracy Sensitivity Specificity
##   0.7719298   0.6969697   0.8750000
```

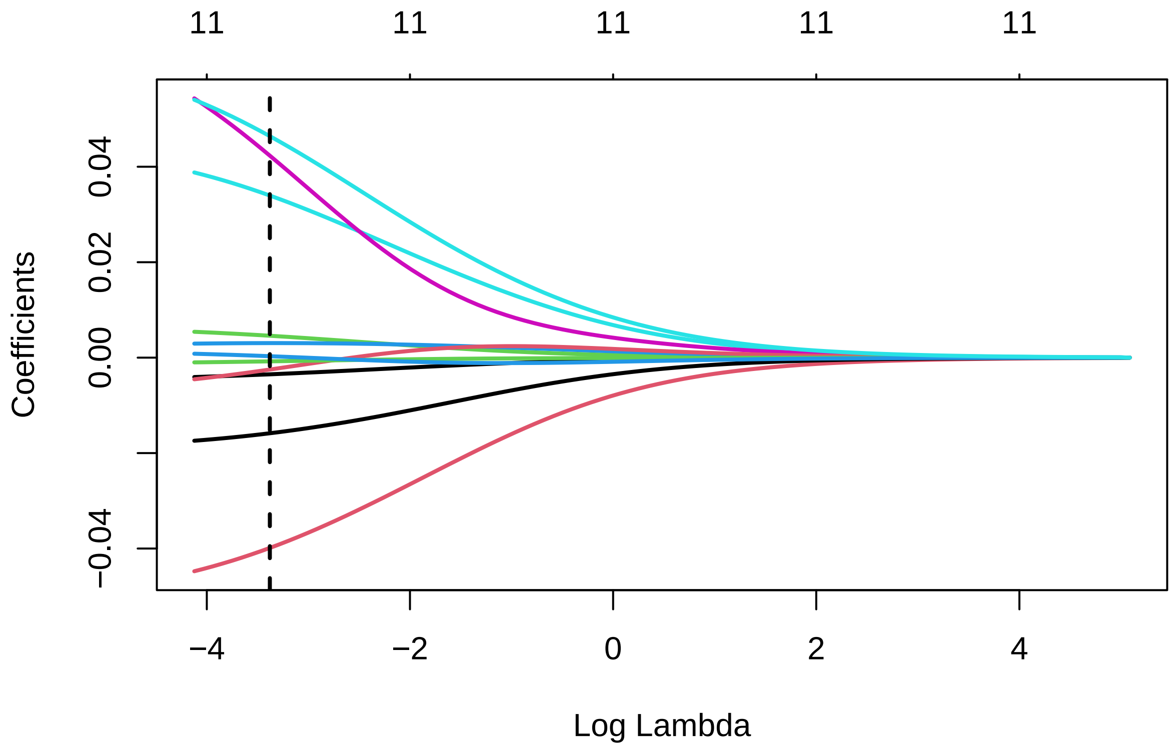
Ridge Regression

```
cv.ridge <- cv.glmnet(as.matrix(train.data.log[, 1:(ncol(train.data.log) - 1)]), train.data.log[, ncol(

ridge.model <- glmnet(as.matrix(train.data.log[, 1:(ncol(train.data.log) - 1)]), train.data.log[, ncol(

coef(ridge.model, s=cv.ridge$lambda.min)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)      77.9782306359
## Top.Genre        -0.0034914179
## Year             -0.0398689898
## Beats.Per.Minute  0.0045884310
## Energy            0.0030483618
## Danceability      0.0339685072
## Loudness          0.0423200080
## Liveness         -0.0158440391
## Valence          -0.0024905395
## Length           -0.0007740176
## Acousticness      0.0003019409
## Speechiness       0.0464087120
```

Ridge Regression: Visualization

```
plot(ridge.model, xvar="lambda", lwd=2)

abline(v=log(cv.ridge$lambda.min), col='black', lty=2, lwd=2)
```



Ridge Regression: Prediction

```
pred.ridge <- predict(cv.ridge, as.matrix(test.data.log[, -ncol(test.data.log)]), type="class")

head(pred.ridge)
```

```
##     lambda.1se
## 102 "0"
## 63  "0"
## 122 "0"
## 86  "1"
## 83  "0"
## 320 "0"
```

Ridge Regression: Evaluation

```
pred_metrics("Ridge Regression Model", test.data.log[, ncol(test.data.log)], pred.ridge)
```

```
## Ridge Regression Model
```

```
##    Accuracy Sensitivity Specificity
##   0.7894737   0.7500000   0.8275862
```

Elastic Net Regression

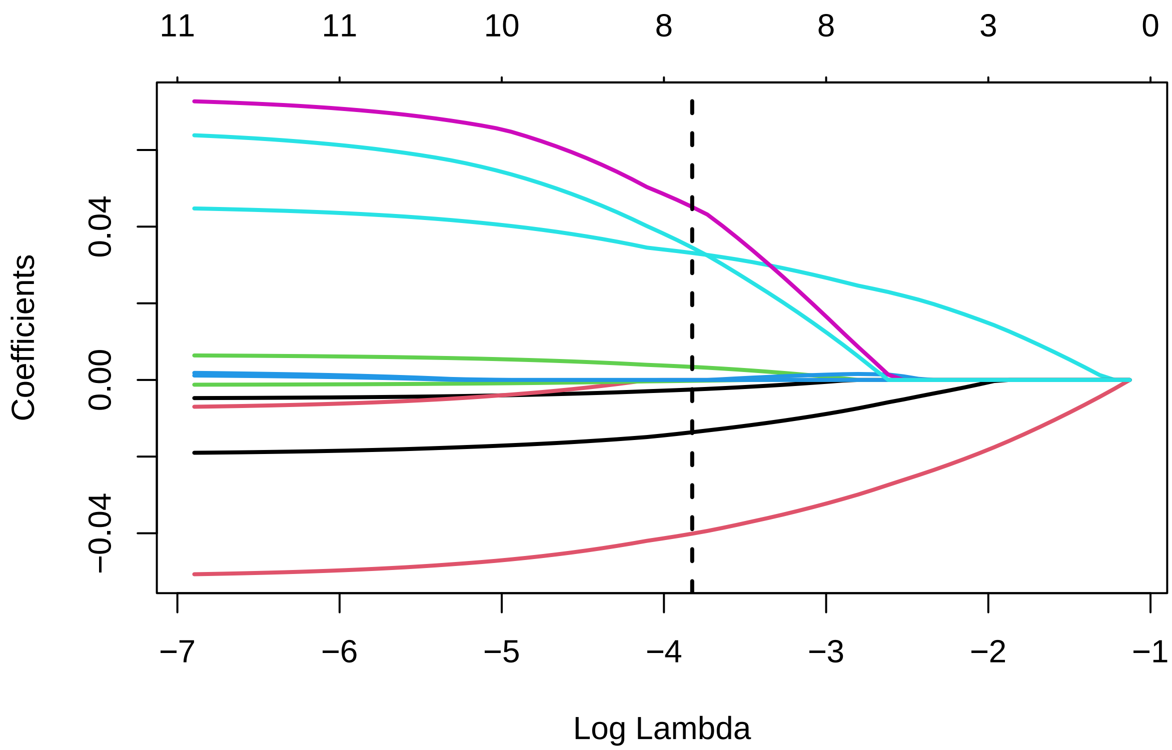
```
cv.el <- cv.glmnet(as.matrix(train.data.log[, 1:(ncol(train.data.log) - 1)]), train.data.log[, ncol(tra
```

```r
el.model <- glmnet(as.matrix(train.data.log[, 1:(ncol(train.data.log) - 1)]), train.data.log[, ncol(trai

coef(el.model, s=cv.el$lambda.min)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)      78.5761625214
## Top.Genre        -0.0025019416
## Year             -0.0401089606
## Beats.Per.Minute  0.0034033634
## Energy            .
## Danceability      0.0331366649
## Loudness          0.0451387682
## Liveness         -0.0136482973
## Valence           .
## Length           -0.0002265023
## Acousticness      .
## Speechiness       0.0345353198
```

Elastic Net Regression: Visualization

```r
plot(el.model, xvar="lambda", lwd=2)

abline(v=log(cv.el$lambda.min), col='black', lty=2, lwd=2)
```



Elastic Net Regression: Prediction

```
pred.el <- predict(cv.el, as.matrix(test.data.log[, -ncol(test.data.log)]), type="class")

head(pred.el)
```

```
##     lambda.1se
## 102 "0"
## 63  "0"
## 122 "0"
## 86  "1"
## 83  "0"
## 320 "0"
```

Elastic Net Regression: Evaluation

```
pred_metrics("Elastic Net Regression Model", test.data.log[, ncol(test.data.log)], pred.el)
```

```
## Elastic Net Regression Model

##    Accuracy Sensitivity Specificity
##   0.7719298   0.7096774   0.8461538
```

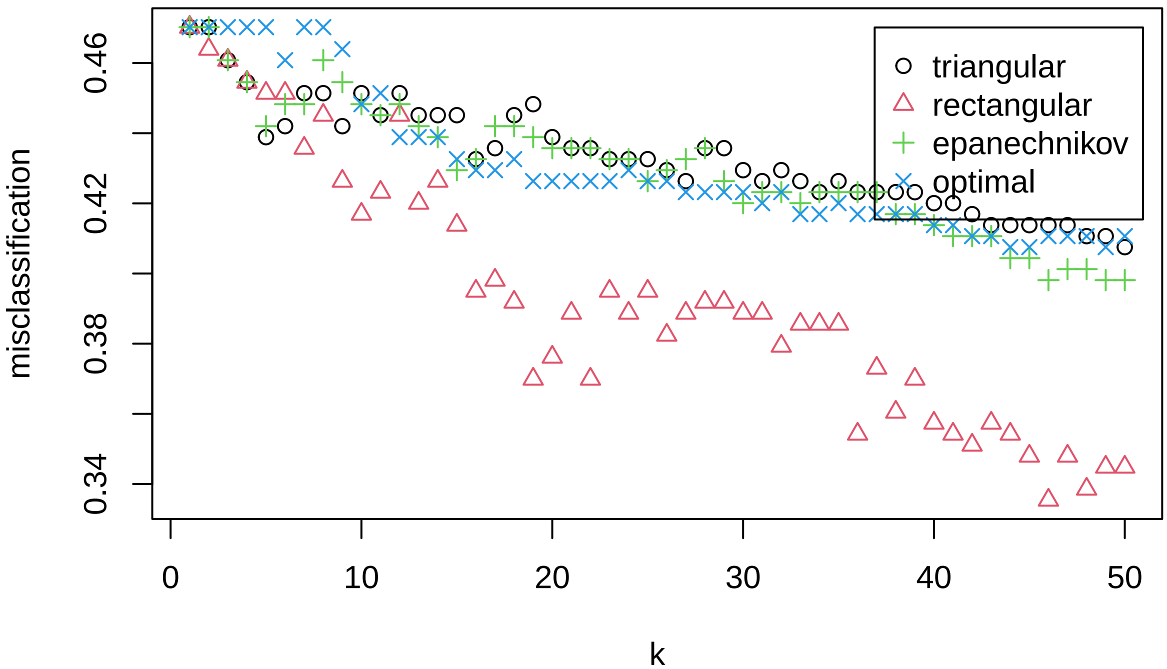
KNN

```
kknn.train <- train.kknn(Class ~ ., train.data.log, kmax = 50,
                          kernel = c("triangular", "rectangular",
                                     "epanechnikov", "optimal"),
                          scale = TRUE)

summary(kknn.train)
```

```
##
## Call:
## train.kknn(formula = Class ~ ., data = train.data.log, kmax = 50,     kernel = c("triangular", "recta
##
## Type of response variable: nominal
## Minimal misclassification: 0.3354232
## Best kernel: rectangular
## Best k: 46
```

KNN: K and Kernel Visualization

```
plot(kknn.train)
```




```
cat("\n The lowest missclassification error is achieved with a",
    kknn.train$best.parameters[[1]],
    "kernel and a number of nearest neighbors (k) of",
    kknn.train$best.parameters[[2]])
```

```
##
##  The lowest missclassification error is achieved with a rectangular kernel and a number of nearest ne
```

KNN: Prediction

```
pred.knn <- predict(kknn.train, test.data.log[, -ncol(test.data.log)])

head(pred.knn)
```

```
## [1] 0 0 0 1 1 0
## Levels: 0 1
```

KNN: Evaluation

```
pred_metrics("KNN Model", test.data.log[, ncol(test.data.log)][1:57], pred.knn)
```

```
## KNN Model
```

```
##    Accuracy Sensitivity Specificity
##   0.7894737   0.7500000   0.8275862
```