**USE CASE STUDY REPORT**

**Group No**.: Group 11

**Student Names**: Kumar Sri Chandra Bhaskar Adabala and Abhinash Ambati

**Executive Summary:**

The goal of this study is to classify the genre of new song/music correctly using data mining techniques. This dataset for this study is available in Kaggle, which is created by the Marsyas (Music Analysis, Retrieval, and Synthesis for Audio Signals), an open-source framework for audio processing. This dataset contains 30 variables with 1000 records for ten genres containing 100 records for each genre. The features of the music data are extracted using the libROSA library. In Data Processing attributes that have a correlation coefficient more than a threshold is also removed since similar trends mean similar information is carried. We have done dimension reduction technique PCA which is a statistical method that reduces the numbers of attributes by lumping highly correlated attributes together, this data is used in different methods. We have standardized and normalizing data to use that data mining algorithms because they work better when features are on a relatively similar scale and close to normally distributed. We have also converted the multiclass response variable to binary class response variables to examine the results using binomials data mining techniques. The data mining techniques applied in this study are K nearest neighbors, Full tree classification, Pruned tree, Random Forest, Multinomial Logistic Regression, Neural Network, and Logistic regression. Out of all applied algorithms, Random forest performed better compared to other algorithms without any overfitting. Further, to improve the accuracy of the model, we would recommend collecting a dataset more observations with features containing low correlation to other predictors and apply data-driven models.

# I. Background and Introduction

Nowadays, all music lovers are interested in listening to personalized music playlists according to their interests. All online streaming platforms such as Spotify, Apple Music, Amazon Music, etc. are working on creating personalized playlists for their users. For this purpose, they have to use highly skilled professionals to identify and classify the music correctly to make playlists, so that it will reach the right audience who are interested in listening to particular types of songs/music such as rock, jazz, pop, blues, etc. However, this is a highly challenging task for them to hire and spend much money against people on this resolution because we have millions of songs out there, considering many languages. Many streaming platforms are releasing tune data to the public to find automated solutions to reduce costs and efforts by using effective Data Mining technologies.

## • The problem

Although it is easy for a human ear to listen and classify a genre of a song or music based on the instruments and the tempo of the tune, it is an essentially subjective task. On the other hand, computers cannot sense the same experience as humans. So, we use sound which can be represented in the form of an audio signal having parameters such as frequency, bandwidth, decibel, etc. A typical audio signal can be expressed as a function of Amplitude and Time. These sounds are available in many formats, for example, mp3, WMA (Windows Media Audio) format, etc., which makes it possible for the computer to read and analyze them. For analyzing audio signals and extracting features of a tune such as a tempo, beats, etc. we can use Marsyas (Music Analysis, Retrieval, and Synthesis for Audio Signals), an open-source framework for audio processing. So, computers need to be trained by feeding with this data containing different features like tempo, beats, etc. which can be further used to classify song/music into distinguishing different genres.

## • The goal of your study

The goal of this study is to classify new songs correctly according to their genres. For this, we need to understand the essential and exciting factors/features that are contributing to the genre of the song/music. So, we can classify the styles of new song/music. For this purpose, we would like to analyze the data using R programming by doing visualization/processing, using data mining techniques, and implementing different possible algorithms.

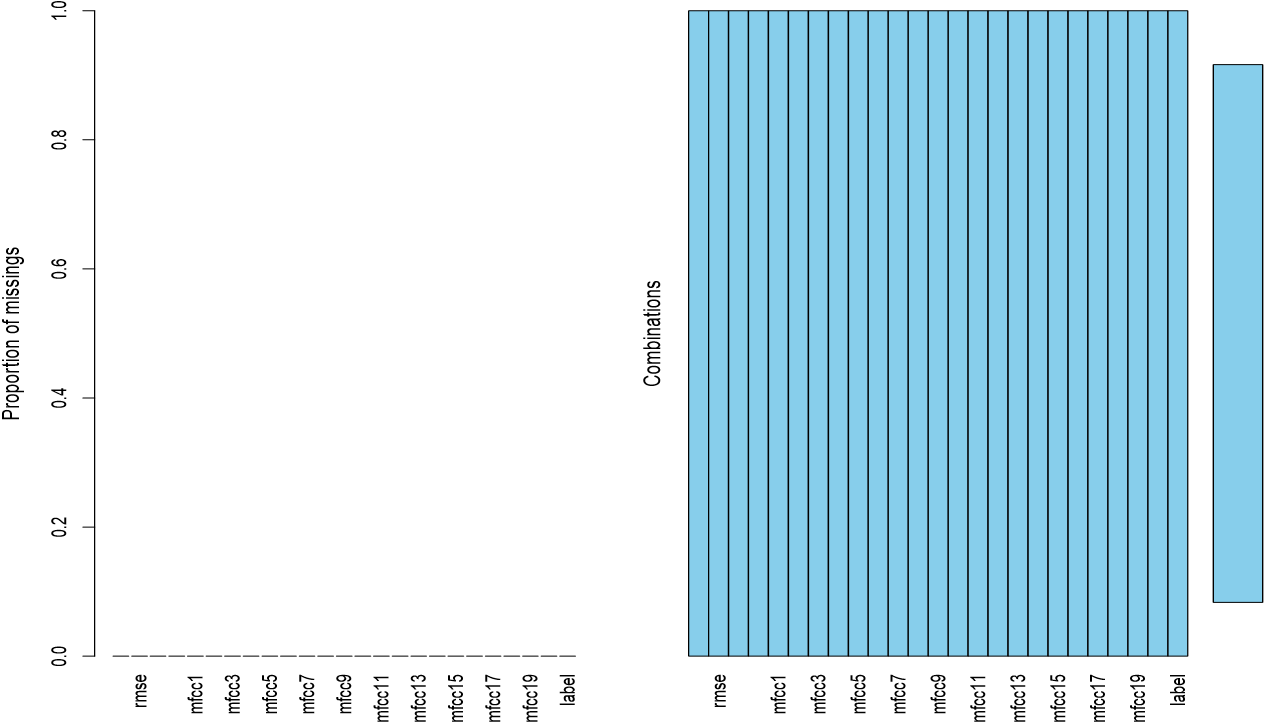
## • The possible solution

The correct approach for this problem would be to visualize the dataset to understand the distribution and correlation between all the variables. If we find any relationship between variables, we will use preprocessing techniques such as PCA to reduce the dimensions and use a few variables to train the data to achieve the best performance. Since we are not aware of which algorithms work well on this data, we will try to apply all possible algorithms and will pick the best model based on their performance scores and accuracy.

# II. Data Exploration and Visualization

## 1. Finding Missing Values

Using aggr function from VIM library we found that our dataset contains no missing values.



## 2. Visualizations

The dataset used contains equal number of observations for each genre of music, namely we are dealing with 10 different genres of music/songs (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae & rock).

0

25

50

75

100

125

b

lues

classical

countr

y

disco

hiphop

jazz

metal

pop

reggae

rock

Music Genre

N

o

o

f

s

o

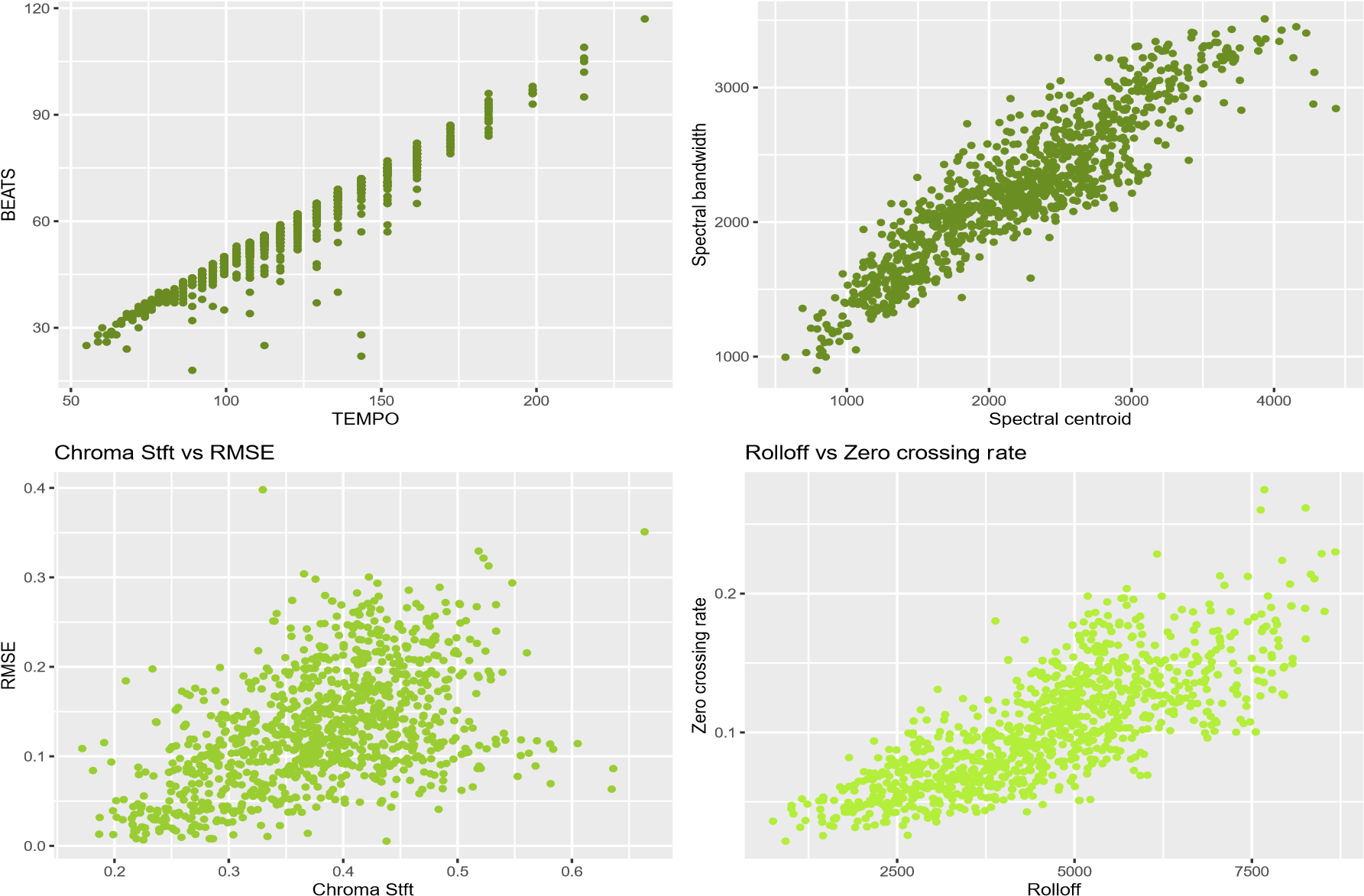
n

g

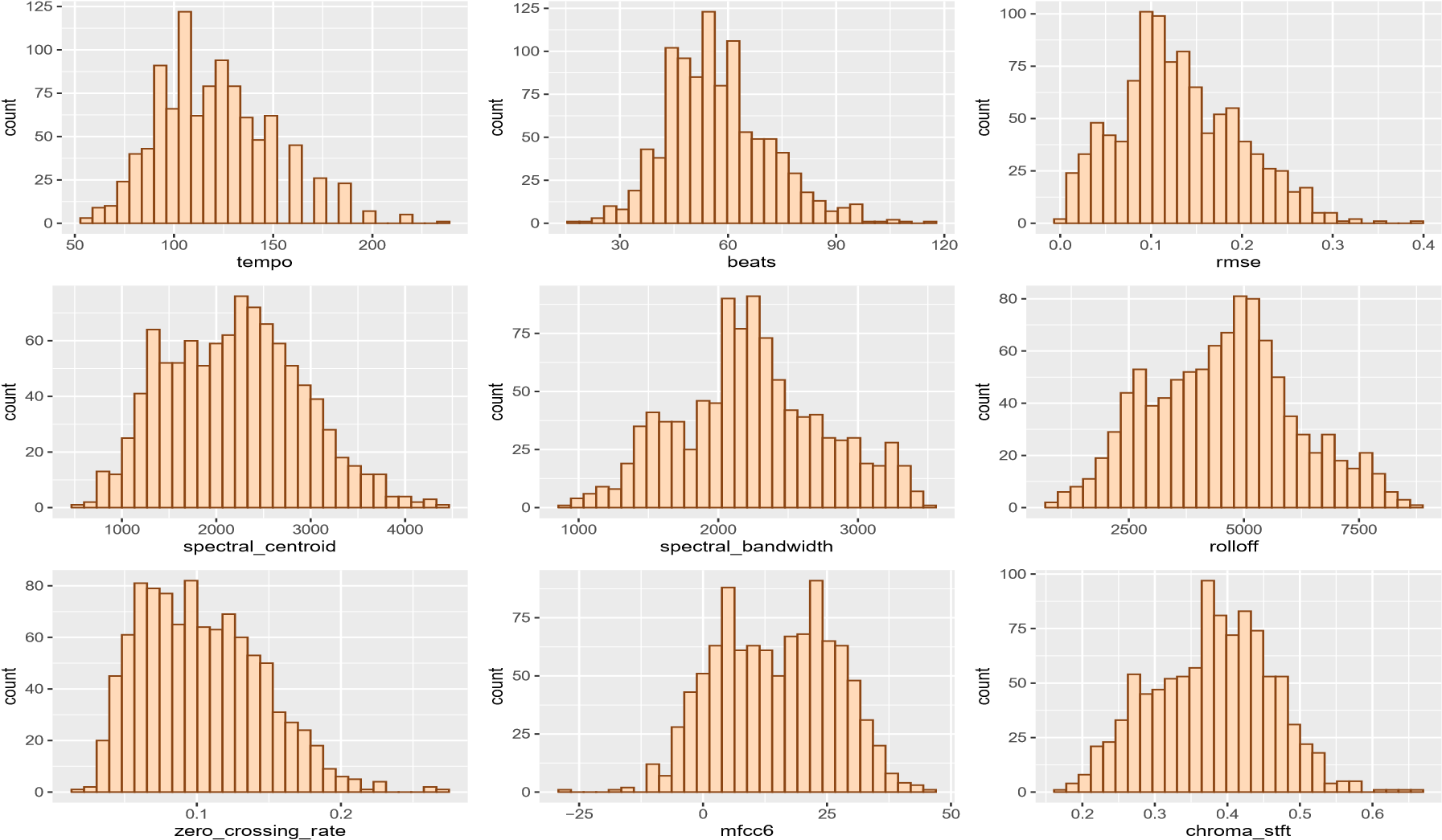
s

When we plotted a scatterplot for few important variables, we found a correlation between some variables such as Tempo vs Beats, Spectral centroid vs Spectral bandwidth.

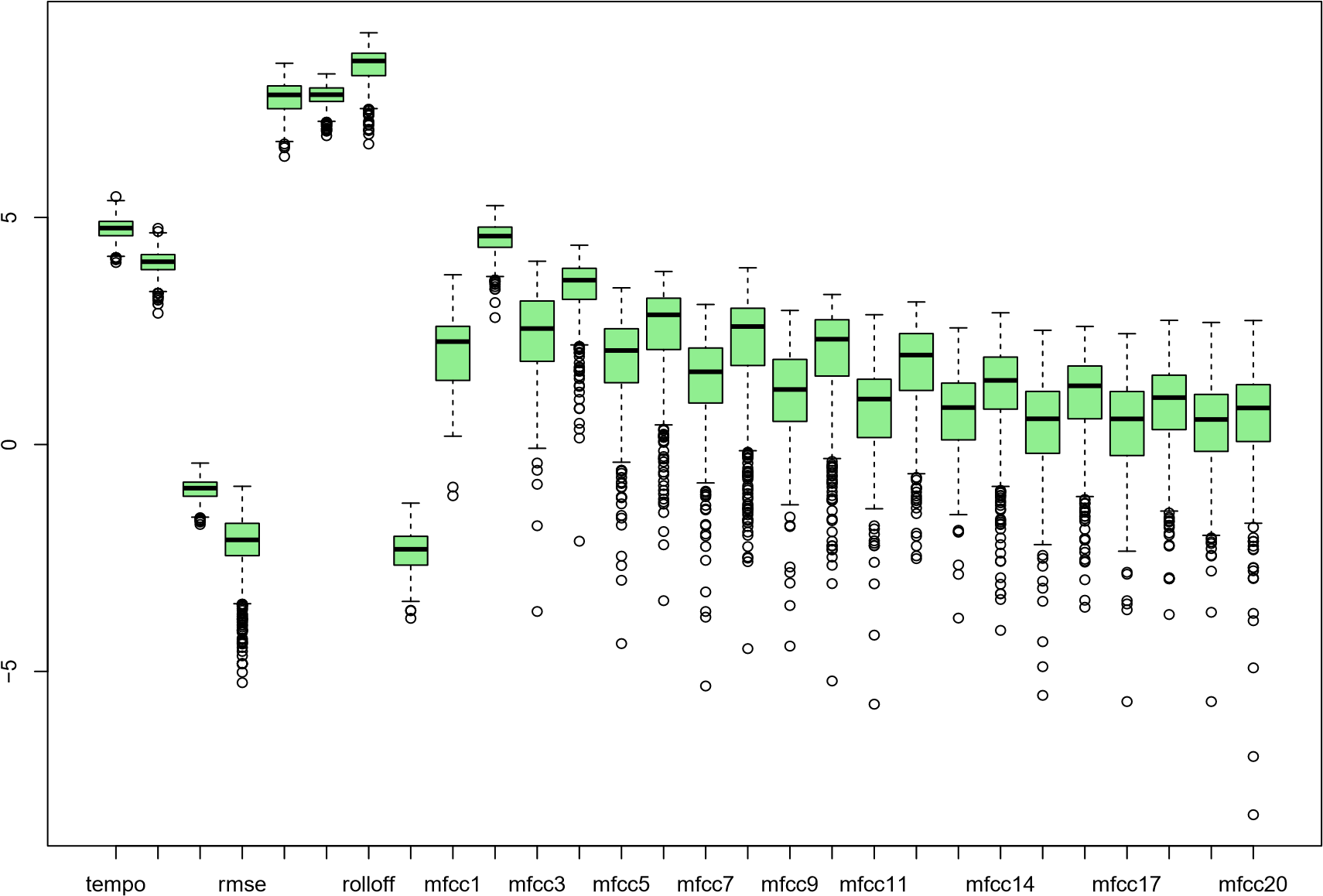
Tempo vs Beats Spectral centroid vs Spectral\_bandwidth



We can see almost all the variables in the dataset are normally distributed with slight skewness for few variables.



When we plotted a boxplot for all the independent variables by rescaling using log function, we found that distribution of rolloff, spectral\_centroid and spectral\_bandwidth is similar and same with tempo and beats.



# III. Data Preparation and Preprocessing

## 1. Data Summary

All the predictors in the dataset are numerical and the predictand(label) is categorical with 10 different classes. The complete summary of the whole dataset is shown in the figure below.

A close up of text on a white background

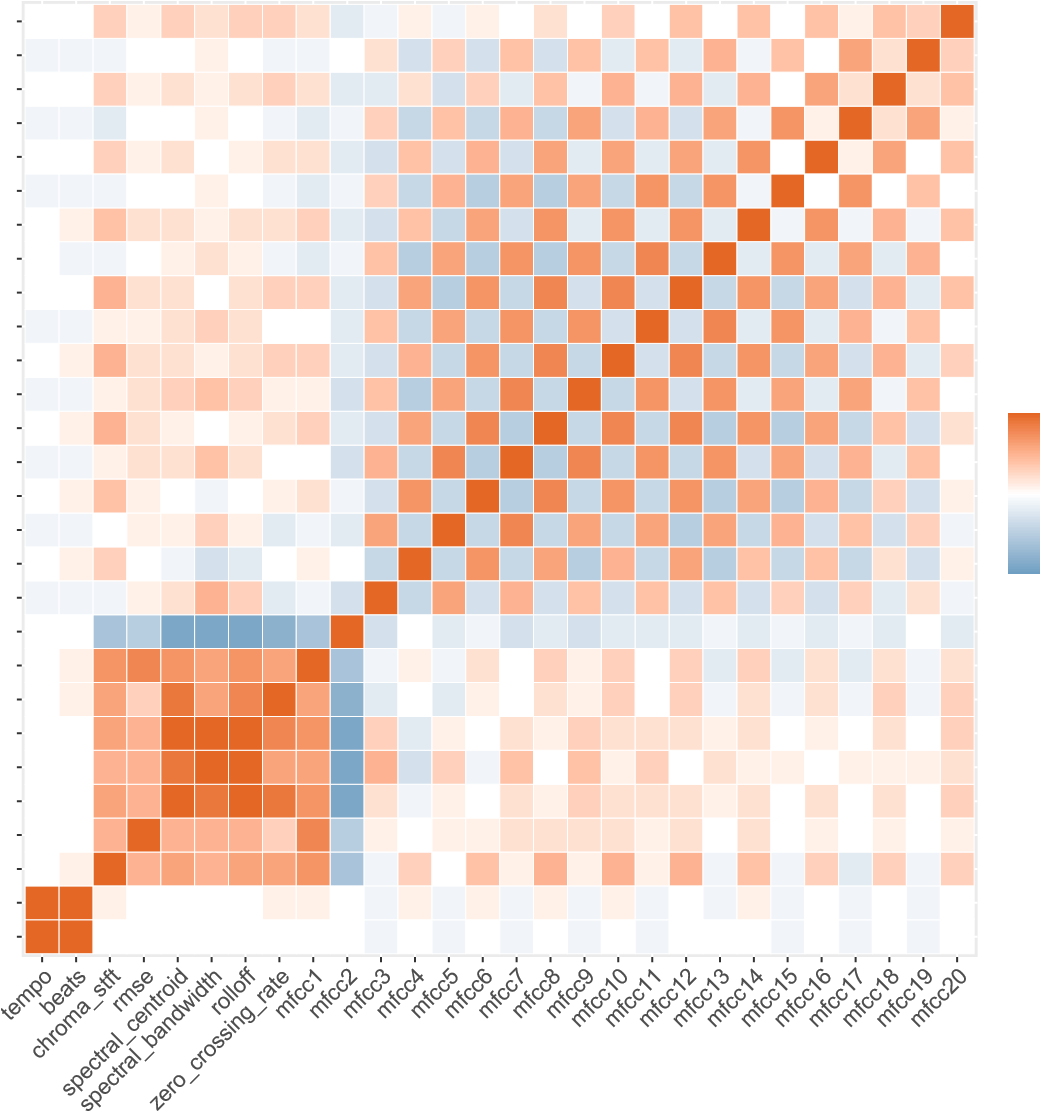
Description automatically generated

## 2. Variable Selection

From the correlation plot, we foundbeats and tempo are highly correlated, so we have only considered tempo. Rolloff, spectral\_centroid, and spectral\_bandwidth also has a high positive correlation**,** so, we removed spectral\_centroid and spectral\_bandwidth. This left us with 25 variables excluding the response variable.

mfcc20 0 0 0.3 0.1 0.3 0.2 0.3 0.3 0.2−0.2−0.10.1−0.10.1 0 0.2 0 0.3 0 0.4 0 0.4 0 0.4 0.1 0.4 0.3 1 mfcc19 −0.1−0.1−0.1 0 0 0.1 0 −0.1−0.1 0 0.2−0.30.3−0.30.4−0.30.4−0.20.4−0.20.5−0.10.4 0 0.6 0.2 1 0.3 mfcc18 0 0 0.3 0.1 0.2 0.1 0.2 0.3 0.2−0.2−0.20.2−0.30.3−0.20.4−0.10.5−0.10.5−0.20.5 0 0.6 0.2 1 0.2 0.4 mfcc17 −0.1−0.1−0.2 0 0 0.1 0 −0.1−0.2−0.10.3−0.40.4−0.40.5−0.40.6−0.30.5−0.30.6−0.10.7 0.1 1 0.2 0.6 0.1 mfcc16 0 0 0.3 0.1 0.2 0 0.1 0.2 0.2−0.2−0.30.4−0.30.5−0.30.6−0.20.6−0.20.6−0.20.7 0 1 0.1 0.6 0 0.4 mfcc15 −0.1−0.1−0.1 0 0 0.1 0 −0.1−0.2−0.10.3−0.40.5−0.50.6−0.50.6−0.40.7−0.40.7−0.1 1 0 0.7 0 0.4 0 mfcc14 0 0.1 0.4 0.2 0.2 0.1 0.2 0.2 0.3−0.2−0.30.4−0.40.6−0.30.7−0.20.7−0.20.7−0.2 1 −0.10.7−0.10.5−0.10.4 mfcc13 0 −0.1−0.1 0 0.1 0.2 0.1−0.1−0.2−0.10.4−0.50.6−0.50.7−0.50.7−0.40.8−0.4 1 −0.20.7−0.20.6−0.20.5 0 mfcc12 0 0 0.5 0.2 0.2 0 0.2 0.3 0.3−0.2−0.30.6−0.50.7−0.40.8−0.30.8−0.3 1 −0.40.7−0.40.6−0.30.5−0.20.4 mfcc11 −0.1−0.10.1 0.1 0.2 0.3 0.2 0 0 −0.20.4−0.40.6−0.40.7−0.40.7−0.3 1 −0.30.8−0.20.7−0.20.5−0.10.4 0 mfcc10 0 0.1 0.5 0.2 0.2 0.1 0.2 0.3 0.3−0.2−0.30.5−0.40.7−0.40.8−0.4 1 −0.30.8−0.40.7−0.40.6−0.30.5−0.20.3

mfcc9 −0.1−0.10.1 0.2 0.3 0.4 0.3 0.1 0.1−0.30.4−0.50.6−0.40.8−0.4 1 −0.40.7−0.30.7−0.20.6−0.20.6−0.10.4 0 Corr

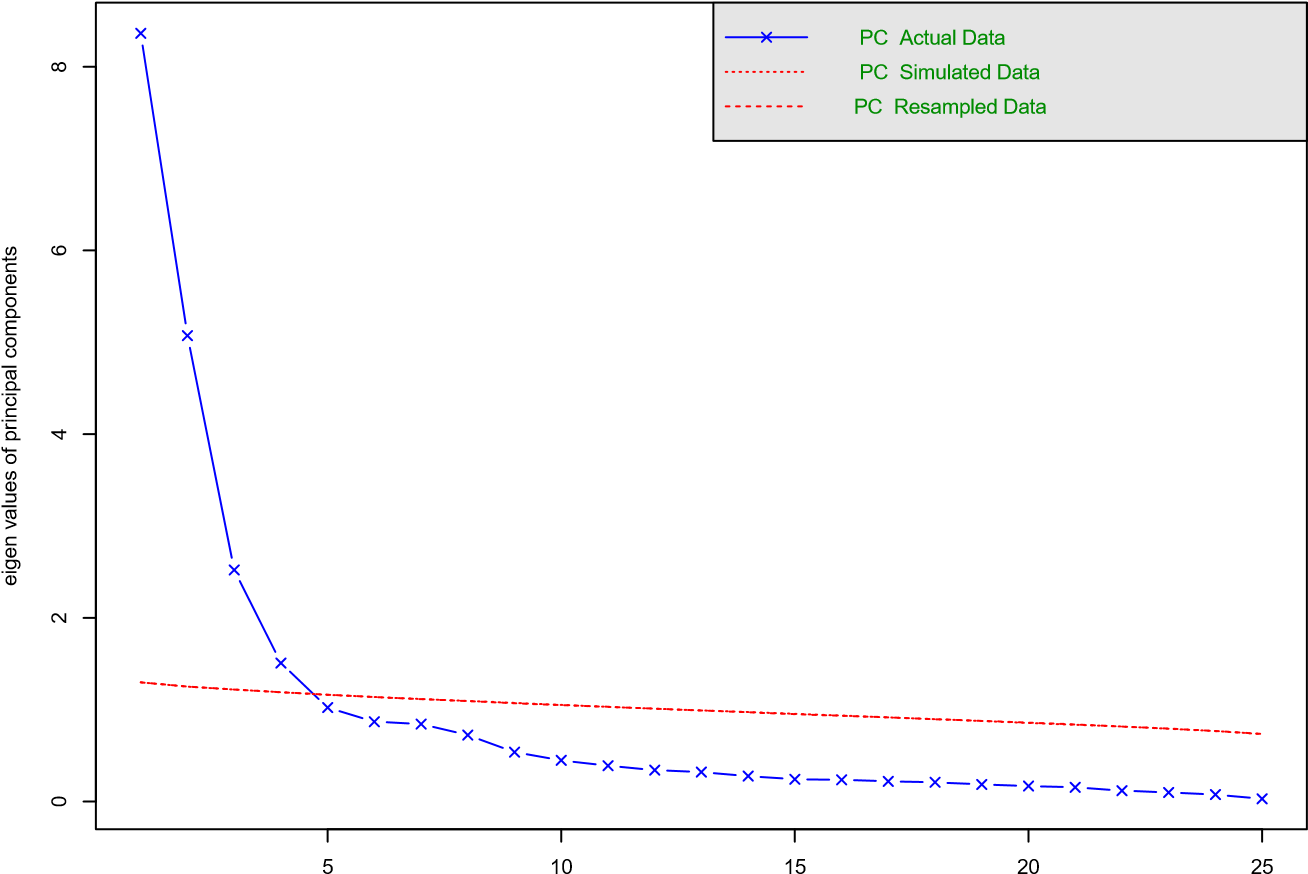
1.0 mfcc8 0 0.1 0.5 0.2 0.1 0 0.1 0.2 0.3−0.2−0.30.6−0.40.8−0.5 1 −0.40.8−0.40.8−0.50.7−0.50.6−0.40.4−0.30.2 mfcc7 −0.1−0.10.1 0.2 0.2 0.4 0.2 0 0 −0.30.5−0.40.8−0.5 1 −0.50.8−0.40.7−0.40.7−0.30.6−0.30.5−0.20.4 0 0.5 mfcc6 0 0.1 0.4 0.1 0 −0.1 0 0.1 0.2−0.1−0.30.7−0.4 1 −0.50.8−0.40.7−0.40.7−0.50.6−0.50.5−0.40.3−0.30.1 0.0 mfcc5 −0.1−0.1 0 0.1 0.1 0.3 0.1−0.2−0.1−0.20.6−0.4 1 −0.40.8−0.40.6−0.40.6−0.50.6−0.40.5−0.30.4−0.30.3−0.1 −0.5

mfcc4 0 0.1 0.3 0 −0.1−0.3−0.2 0 0.1 0 −0.4 1 −0.40.7−0.40.6−0.50.5−0.40.6−0.50.4−0.40.4−0.40.2−0.30.1 −1.0 mfcc3 −0.1−0.1−0.10.1 0.2 0.5 0.3−0.2−0.1−0.3 1 −0.40.6−0.30.5−0.30.4−0.30.4−0.30.4−0.30.3−0.30.3−0.20.2−0.1 mfcc2 0 0 −0.6−0.5−0.9−0.9−0.9−0.8−0.6 1 −0.3 0 −0.2−0.1−0.3−0.2−0.3−0.2−0.2−0.2−0.1−0.2−0.1−0.2−0.1−0.2 0 −0.2 mfcc1 0 0.1 0.7 0.8 0.7 0.6 0.7 0.6 1 −0.6−0.10.1−0.10.2 0 0.3 0.1 0.3 0 0.3−0.20.3−0.20.2−0.20.2−0.10.2 zero\_crossing\_rate 0 0.1 0.6 0.3 0.9 0.6 0.8 1 0.6−0.8−0.2 0 −0.20.1 0 0.2 0.1 0.3 0 0.3−0.10.2−0.10.2−0.10.3−0.10.3 rolloff 0 0 0.6 0.5 1 1 1 0.8 0.7−0.90.3−0.20.1 0 0.2 0.1 0.3 0.2 0.2 0.2 0.1 0.2 0 0.1 0 0.2 0 0.3 spectral\_bandwidth 0 0 0.5 0.5 0.9 1 1 0.6 0.6−0.90.5−0.30.3−0.10.4 0 0.4 0.1 0.3 0 0.2 0.1 0.1 0 0.1 0.1 0.1 0.2 spectral\_centroid 0 0 0.6 0.5 1 0.9 1 0.9 0.7−0.90.2−0.10.1 0 0.2 0.1 0.3 0.2 0.2 0.2 0.1 0.2 0 0.2 0 0.2 0 0.3 rmse 0 0 0.5 1 0.5 0.5 0.5 0.3 0.8−0.50.1 0 0.1 0.1 0.2 0.2 0.2 0.2 0.1 0.2 0 0.2 0 0.1 0 0.1 0 0.1 chroma\_stft 0 0.1 1 0.5 0.6 0.5 0.6 0.6 0.7−0.6−0.10.3 0 0.4 0.1 0.5 0.1 0.5 0.1 0.5−0.10.4−0.10.3−0.20.3−0.10.3 beats 1 1 0.1 0 0 0 0 0.1 0.1 0 −0.10.1−0.10.1−0.10.1−0.10.1−0.1 0 −0.10.1−0.1 0 −0.1 0 −0.1 0 tempo 1 1 0 0 0 0 0 0 0 0 −0.1 0 −0.1 0 −0.1 0 −0.1 0 −0.1 0 0 0 −0.1 0 −0.1 0 −0.1

## 3. PCA (Principal Component Analysis)

By plotting a parallel analysis scree plot we found 4 PCA components that will be ideal to perform further analysis. After implementing all data mining techniques with PCA components we observed this data is not efficient when data mining techniques

**Parallel Analysis Scree Plots**



Component Number

## 4. Variables Conversion

We standardized and Normalized all the input variables so that we can use the appropriate version of data for the algorithms.

To apply algorithms like Binary tree classification, Neural network classification, Logistic regression and to obtain high accuracy, we converted the response variable into two main classes called “Traditional” as “0” and “Modern” as “1”. The new binary response variable data is then normalized to be used for building models.

## 5. Data Splitting

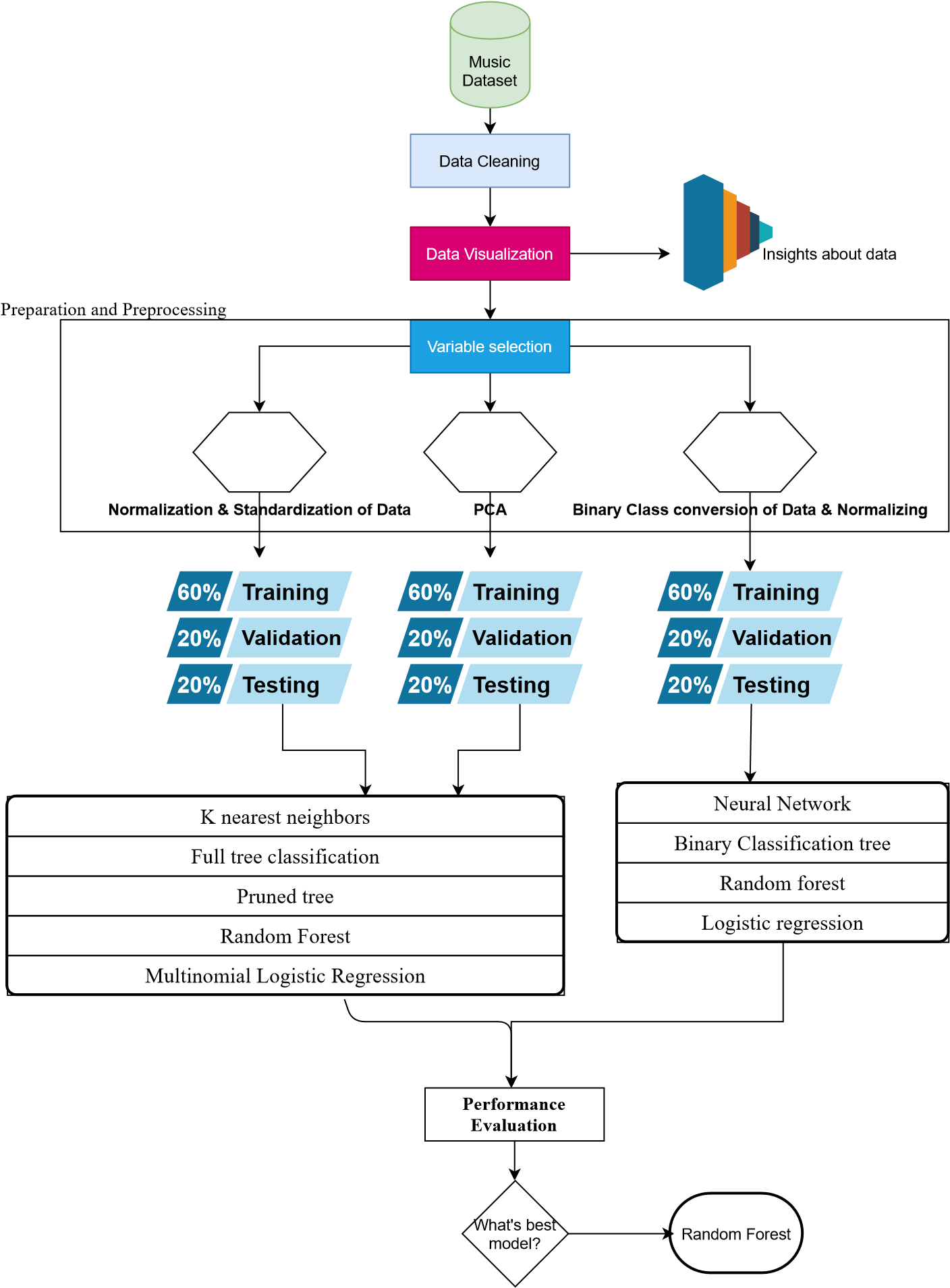
All versions of obtained data such as normalized, standardized, binary class is split into Training, Validation and Testing with 60%, 20%, 20% ratios.

# IV. Data Mining Techniques and Implementation

For this problem we aim to classify the genre of the music based on the set of predictors, we have 10 classes to classify so this is a multiclass classification problem. The supervised algorithms that can classify multiple classes based on predictors are used here. The algorithms that we have implemented here are K nearest neighbors, Full tree classification, Pruned tree, Random Forest, Multinomial Logistic Regression.

In addition to this, we have converted the multiple classes response variable into two main classes “Traditional” music and “Modern” music to make it into a binary class problem to obtain high accuracies. The algorithms that we have implemented for this data are Neural Network, Binary Classification tree, Binary classifying Random forest, Logistic regression.

## Flow Chart for implementation



# V. Performance Evaluation

# A. Evaluation of multiclass classification Algorithms

## 1. KNN

To apply knn we tried different “k” values and found k=3 is better with good performance and no overfitting. The performance, confusion matrix of knn algorithm with k = 3 on validation data is shown below.

A screenshot of a cell phone

Description automatically generated

## 2. Full Tree

The performance, confusion matrix of Classification tree algorithm on test data is shown below.

A screenshot of a cell phone

Description automatically generated

## 3. Pruned Tree

When the full tree is pruned and applied on test data, the following results are obtained.

A screenshot of a cell phone

Description automatically generated

## 4. Random Forest

When we applied random forest algorithm by varying number of trees, we found that the model performance is better with number of trees equals to 600. The performance, confusion matrix of Random forest algorithm on test data is shown below.

A screenshot of a cell phone

Description automatically generated

## 5. Multinomial Logistic Regression

The performance, confusion matrix of Multinomial Logistic regression on Validation data is shown below.

A screenshot of a cell phone

Description automatically generated

# B. Evaluation of binomial classification Algorithms

## 1. Neural Network

This algorithm is applied on the data where predictand is converted to binary class. After trying different thresholds, number of hidden layers, and number of hidden nodes, we found that default threshold, 2 hidden layers with 10 nodes each performed well. The Lift chart, performance, confusion matrix of Neural network algorithm on validation data is shown below.

**Neural Network Lift**

A screenshot of text

Description automatically generated

## 2. Binary Classification Tree

This algorithm is applied on the binary class converted data. The performance, confusion matrix of Classification tree algorithm on test data is shown below.

A screenshot of text

Description automatically generated

## 3. Random Forest for Binary

This algorithm is applied on the binary class converted data. When we applied random forest algorithm by trying different number of trees, we found that model performed better with number of trees is 700. The performance, confusion matrix of Random forest algorithm on test data is shown below.

A screenshot of text

Description automatically generated

## 4. Logistic Regression

## 

This algorithm is applied on the binary class converted data. The coefficients, performance, confusion matrix, lift chart of Logistic Regression algorithm on validation data is shown below.

A close up of text on a white background

Description automatically generated A screenshot of text

Description automatically generated

## Best Approach

The best approach for this problem is to use the Random Forest algorithm because the performance of the algorithm is very high when compared to all other algorithms considering both binary classes and multiple classes. This algorithm works well because it creates a different tree for different iteration and increases the accuracy of the model by performing multiple iterations. So, this model trains the data which makes overfitting low for this algorithm. For Binary class data we got an accuracy of 81% on test data which is higher than the accuracy achieved on validation dataset that means the model is not overfitting and performing well. For multiple classes, we got an accuracy of 68.5% accuracy.

## Accuracies Table

**Multiclass classification Algorithms**

|  |  |  |
| --- | --- | --- |
| **No.** | **Algorithms** | **Accuracy** |
| 1 | K nearest neighbors | 61% |
| 2 | Full tree classification | 46% |
| 3 | Pruned tree | 46.5% |
| 4 | Random Forest | 68.5% |
| 5 | Multinomial Logistic Regression | 61% |

**Binomial classification Algorithms**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Algorithms** | | | | | |  | **Accuracy** |
| 1 |  | Neural Network for binary tree | | |  | |  | 77.5% |
| 2 |  | Binary Classification tree | | | |  |  | 69.5% |
| 3 |  | Random forest for binary | data | | | |  | 81% |
| 4 |  | Logistic regression | |  | | |  | 74.5% |

# VI. Discussion and Recommendation

In this study, we have performed different data mining classification algorithms and evaluated the performances using validation and testing data. We have tried the classification in two ways, one by using all classes and classifying with multiclass classification supported algorithms and others by converting the 10 classes in traditional music and modern music. We got good accuracies with KNN, Multinomial regression and Random forest with 61%, 61% and 68.5% compared to other multiclass response variables. We got good accuracies with Random forest and Neural network for the binary class response variable.

Further, we would like to recommend making use of a dataset with more records and features for a greater number of songs so that the model with multiclass classification be trained better with all possible cases for each genre to improve the accuracy of the models. We have achieved more accuracies using Binomial classification Algorithms because of the number of response variables. When trying this approach, it is highly recommended to build data-driven models so that the higher accuracy without overfitting will be achieved.

# VII. Summary

An automatic music genre classification model project was aimed to tackle the problem of automatic music genre classification based on various features. The results after applying different supervised data mining classification algorithms look promising. These models can be used by digital music platforms to make playlists for their users based on their preference of music genre.

# Appendix: R Code for use case study

---

title: "Group 11 - Case Study" author:

* Kumar Sri Chandra Bhaskar Adabala, NUID - 001083381
* Abhinash Ambati, NUID - 001023924 output: pdf\_document

---

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

```

\footnotesize

```{r message=FALSE, warning=FALSE, include=FALSE, paged.print=TRUE}

#Including Libraries library(tidyverse) library(GGally) library(psych) library(rpart) library(rpart.plot) library(e1071) library(caret) library(class) library(forecast) library(ggcorrplot) library(VIM) library(neuralnet)

library(gains)

```

```{r include=FALSE} # Loading Datasets

music\_genre\_data <- read.csv("~/Downloads/Course Materials/IE 7275 - Data Mining in Engineering/Assignments/Case Study/Data/data.csv", stringsAsFactors = T)

```

# 2. Data Exploration and Visualization

## i. Cleaning Data

```{r}

#Checking for na values

colSums(sapply(music\_genre\_data, is.na)) aggr(music\_genre\_data)

#Remove Unessesary variables

music\_genre\_data <- music\_genre\_data[,-1]

#we found that the first column "filename" is useless for our classification and training the data. ```

\* We found that there are no missing values in the data and removed the Unessesary columns.

## ii. Visualizing Data

```{r}

ggplot(data = music\_genre\_data)+ geom\_bar(mapping = aes(x=label), color= 'saddlebrown',fill = 'peachpuff1')+scale\_x\_discrete("Music Genre")+ylim(0,120)+ labs( x="Music Genre", y="No of songs")

#we see all genres classes have equal number of records in the dataset.

#Boxplot for Tempo vs Beats

figs1 =ggplot(music\_genre\_data) + geom\_point(mapping = aes(x= tempo, y= beats), color = 'olivedrab4') + labs( x="TEMPO", y="BEATS")+

ggtitle("Tempo vs Beats")

figs2 =ggplot(music\_genre\_data) + geom\_point(mapping = aes(x= spectral\_centroid, y= spectral\_bandwidth), color = 'olivedrab') + labs( x="Spectral centroid", y="Spectral bandwidth")+ ggtitle("Spectral centroid vs Spectral\_bandwidth")

figs3 =ggplot(music\_genre\_data) + geom\_point(mapping = aes(x= chroma\_stft, y= rmse), color = 'olivedrab3') + labs( x="Chroma Stft", y="RMSE")+

ggtitle("Chroma Stft vs RMSE")

figs4 = ggplot(music\_genre\_data) + geom\_point(mapping = aes(x= rolloff, y= zero\_crossing\_rate), color = 'olivedrab2') + labs( x="Rolloff", y="Zero crossing rate")+

ggtitle("Rolloff vs Zero crossing rate")

gridExtra::grid.arrange(figs1, figs2, figs3, figs4, ncol=2)

#Histogram to show distribution of different observation data

figh1 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=tempo), color= 'saddlebrown',fill = 'peachpuff1')

figh2 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=beats), color= 'saddlebrown',fill = 'peachpuff1')

figh3 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=rmse), color= 'saddlebrown',fill = 'peachpuff1')

figh4 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=spectral\_centroid), color= 'saddlebrown',fill = 'peachpuff1')

figh5 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=spectral\_bandwidth), color= 'saddlebrown',fill = 'peachpuff1')

figh6 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=rolloff), color= 'saddlebrown',fill = 'peachpuff1')

figh7 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=zero\_crossing\_rate), color= 'saddlebrown',fill = 'peachpuff1')

figh8 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=mfcc6), color= 'saddlebrown',fill = 'peachpuff1')

figh9 = ggplot(data = music\_genre\_data)+

geom\_histogram(mapping = aes(x=chroma\_stft), color= 'saddlebrown',fill = 'peachpuff1')

gridExtra::grid.arrange(figh1, figh2, figh3, figh4,figh5,figh6,figh7,figh8,figh9, ncol=3)

#Box-plots for labels

fig1 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= tempo), fill = 'grey') +

ggtitle("tempo vs Label")

fig2 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= beats), fill = 'skyblue') +

ggtitle("beats vs Label")

fig3 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= chroma\_stft), fill = 'yellow') +

ggtitle("chroma\_stft vs Label")

fig4 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= rmse), fill

= 'navyblue') + ggtitle("rmse vs Label")

fig5 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= spectral\_centroid), fill = 'red') +

ggtitle("spectral\_centroid vs Label")

fig6 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= spectral\_bandwidth), fill = 'orange') +

ggtitle("spectral\_bandwidth vs Label")

fig7 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= rolloff), fill = '#E46726') +

ggtitle("rolloff vs Label")

fig8 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= zero\_crossing\_rate), fill = 'lightgreen') + ggtitle("zero\_crossing\_rate vs Label")

fig9 <- ggplot(music\_genre\_data) + geom\_boxplot(mapping = aes(x= label, y= mfcc1), fill = 'violet') +

ggtitle("mfcc1 vs Label")

gridExtra::grid.arrange(fig1, fig2, fig3, fig4,fig5,fig6,fig7,fig8,fig9, ncol=3)

#Plots for MFCC

fig11 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc1,), color = 'magenta')

fig12 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc3 ), color = 'magenta1')

fig13 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc5 ), color = 'magenta1')

fig14 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc11 ), color = 'magenta3')

fig15 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc2 ), color = 'magenta4') fig16 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc18), color = 'maroon')

fig17 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc16 ), color = 'maroon1') fig18 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc20 ), color = 'maroon3') fig19 <- ggplot(data = music\_genre\_data)+

geom\_line(mapping = aes(x =label ,y= mfcc13 ), color = 'maroon4')

gridExtra::grid.arrange(fig11, fig12, fig13, fig14,fig15,fig16,fig17,fig18,fig19, ncol=3)

boxplot(log(music\_genre\_data[,-29]), col = "lightgreen")

#Corelation-Graph

corr <- round(cor(music\_genre\_data[,-29]), 1) ggcorrplot(corr, lab = TRUE, outline.col = "white", ggtheme = ggplot2::theme\_gray,

colors = c("#6D9EC1", "white", "#E46726"))

```

# 3. Data Preparation and Preprocessing

```{r}

str(music\_genre\_data)

summary(music\_genre\_data)

# Our response variable is label which shows the genre classes.

# All predictor variables are numerical variables except for the reponse/dependent variable.

```

### i. Variable Selection

```{r}

#Corelation-Graph

corr <- round(cor(music\_genre\_data[,-29]), 1)

ggcorrplot(corr, lab = TRUE, outline.col = "white", ggtheme = ggplot2::theme\_gray,

colors = c("#6D9EC1", "white", "#E46726"))

#From Correlation we see that beats and tempo are highly correlated so we are removing beats

#From Correlation we see that rolloff and spectral\_centroid and spectral\_bandwidth are highly correlated so we are removing spectral\_centroid, spectral\_bandwidth. music\_genre\_data <- music\_genre\_data[,c(-2,-5,-6)]

```

### ii. PCA ```{r}

cor(music\_genre\_data[,-26])

fa.parallel(music\_genre\_data[,-26], fa="pc", n.iter=100,show.legend = T)

# we get to know that we have to use 5 components.

p <- principal(data.frame(music\_genre\_data[,-26]), nfactors = 4, rotate = "none") p$scores

pca\_genre <- cbind.data.frame( p$scores, Genre = music\_genre\_data$label) pca\_genre

# Splitting PCA data

set.seed(100)

train\_index <- sample(1:nrow(pca\_genre), 0.6 \* nrow(pca\_genre))

valid\_index <- sample(setdiff(1:nrow(pca\_genre), train\_index),0.2\*nrow(pca\_genre)) test\_index <- setdiff(1:nrow(pca\_genre), union(train\_index, valid\_index))

train\_df\_pca <- pca\_genre[train\_index, ] valid\_df\_pca <- pca\_genre[valid\_index, ]

test\_df\_pca <- pca\_genre[test\_index, ]

```

### iii. Standardizing Data

```{r}

Standard\_data <- music\_genre\_data

Standard\_data[,-26] <- scale(Standard\_data[,-26], center = T, scale = T)

#Splitting normal data

train\_index <- sample(1:nrow(Standard\_data), 0.6 \* nrow(Standard\_data)) valid\_index <- sample(setdiff(1:nrow(Standard\_data), train\_index),0.2\*nrow(Standard\_data))

test\_index <- setdiff(1:nrow(Standard\_data), union(train\_index, valid\_index))

train\_df\_stand <- Standard\_data[train\_index, ] valid\_df\_stand <- Standard\_data[valid\_index, ]

test\_df\_stand <- Standard\_data[test\_index, ]

```

### iv. Normalizing Data

```{r}

normal\_data <- music\_genre\_data

normalize <- function(x){

return ((x - min(x))/(max(x) - min(x)))

}

normal\_data[,-26] <- as.data.frame(lapply(normal\_data[,-26], normalize))

#Splitting normal data

train\_index <- sample(1:nrow(normal\_data), 0.6 \* nrow(normal\_data)) valid\_index <- sample(setdiff(1:nrow(normal\_data), train\_index),0.2\*nrow(normal\_data))

test\_index <- setdiff(1:nrow(normal\_data), union(train\_index, valid\_index))

train\_df\_norm <- normal\_data[train\_index, ] valid\_df\_norm <- normal\_data[valid\_index, ]

test\_df\_norm <- normal\_data[test\_index, ]

```

### v. Making binary classes

```{r}

#All traditional music is "0" and modern music is "1"

music\_new\_binary\_class <- music\_genre\_data

music\_new\_binary\_class$label <- as.character(music\_genre\_data$label) summary(music\_genre\_data$label) music\_new\_binary\_class$label[music\_new\_binary\_class$label == "blues"] <- 0 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "classical"] <- 0 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "rock"] <- 0 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "reggae"] <- 0 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "country"] <- 0 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "pop"] <- 1 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "jazz"] <- 1 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "metal"] <- 1 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "hiphop"] <- 1 music\_new\_binary\_class$label[music\_new\_binary\_class$label == "disco"] <- 1

music\_new\_binary\_class$label <- as.numeric(music\_new\_binary\_class$label)

normalize <- function(x){

return ((x - min(x))/(max(x) - min(x)))

}

normal\_data<- as.data.frame(lapply(music\_new\_binary\_class, normalize))

#Splitting normal data

train\_index <- sample(1:nrow(normal\_data), 0.6 \* nrow(normal\_data)) valid\_index <- sample(setdiff(1:nrow(normal\_data), train\_index),0.2\*nrow(normal\_data))

test\_index <- setdiff(1:nrow(normal\_data), union(train\_index, valid\_index))

train\_df\_norm\_binary <- normal\_data[train\_index, ] valid\_df\_norm\_binary <- normal\_data[valid\_index, ] test\_df\_norm\_binary <- normal\_data[test\_index, ]

```

### vi. Making binary classes

```{r}

music\_new\_binary\_class <- music\_genre\_data

music\_new\_binary\_class$label <- as.character(music\_genre\_data$label) summary(music\_genre\_data$label)

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "blues"] <-

"Traditional"

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "classical"] <-

"Traditional"

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "rock"] <-

"Traditional"

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "reggae"] <-

"Traditional"

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "country"] <- "Traditional"

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "pop"] <- "Modern" music\_new\_binary\_class$label[music\_new\_binary\_class$label == "jazz"] <- "Modern" music\_new\_binary\_class$label[music\_new\_binary\_class$label == "metal"] <- "Modern" music\_new\_binary\_class$label[music\_new\_binary\_class$label == "hiphop"] <-

"Modern"

music\_new\_binary\_class$label[music\_new\_binary\_class$label == "disco"] <- "Modern"

music\_new\_binary\_class$label <- as.numeric(music\_new\_binary\_class$label)

normalize <- function(x){

return ((x - min(x))/(max(x) - min(x)))

}

normal\_data<- as.data.frame(lapply(music\_new\_binary\_class, normalize))

#Splitting normal data

train\_index <- sample(1:nrow(music\_new\_binary\_class), 0.6 \* nrow(music\_new\_binary\_class))

valid\_index <- sample(setdiff(1:nrow(music\_new\_binary\_class), train\_index),0.2\*nrow(music\_new\_binary\_class))

test\_index <- setdiff(1:nrow(music\_new\_binary\_class), union(train\_index, valid\_index))

train\_df\_norm\_binary <- music\_new\_binary\_class[train\_index, ] valid\_df\_norm\_binary <- music\_new\_binary\_class[valid\_index, ] test\_df\_norm\_binary <- music\_new\_binary\_class[test\_index, ]

```

# 4. Data Mining Techniques and Implementation & 5. Performance Evaluation

## Algorithms

### i. KNN

```{r}

knn\_genre <- knn(train = train\_df\_stand[,-26, drop = T], test = valid\_df\_stand[,-26, drop = T], cl = train\_df\_stand[,26], k = 3)

knn\_genre\_test <- knn(train = train\_df\_stand[,-26, drop = T], test = test\_df\_stand[,-26, drop = T], cl = train\_df\_stand[,26], k = 3)

confusionMatrix(knn\_genre, valid\_df\_stand[,26])

confusionMatrix(knn\_genre\_test, test\_df\_stand[,26])

```

### ii. Full Tree

```{r} # Full tree

tree <- rpart(label ~ ., data = train\_df\_stand, method = "class") rpart.plot(tree)

tree$variable.importance

confusionMatrix(predict(tree, valid\_df\_stand[,-26], type = "class"),valid\_df\_stand[,26]) confusionMatrix(predict(tree, test\_df\_stand[,-26], type = "class"),test\_df\_stand[,26])

```

### iii. Pruned Tree

```{r}

#Pruned

pru\_tree <- rpart(label ~ ., data = train\_df\_stand, method = "class", cp = 0.00001, minsplit = 5, xval = 5)

pruned\_tree <- prune(pru\_tree, cp =

pru\_tree$cptable[which.min(pru\_tree$cptable[,"xerror"]),"CP"]) prp(pruned\_tree)

confusionMatrix(predict(pruned\_tree, valid\_df\_stand[,-26], type =

"class"),valid\_df\_stand[,26])

confusionMatrix(predict(pruned\_tree, test\_df\_stand[,-26], type =

"class"),test\_df\_stand[,26])

# We got same accuracy and performance for best pruned valudation and testing set.

```

### iv. Random Forest

```{r}

rf <- randomForest::randomForest(label ~ ., data = train\_df\_stand, ntree = 600, proximity

= TRUE)

confusionMatrix(predict(rf, valid\_df\_stand[,-26], type = "class"),valid\_df\_stand[,26])

confusionMatrix(predict(rf, test\_df\_stand[,-26], type = "class"),test\_df\_stand[,26])

```

### v. Multinominal Logistic Regression ```{r}

library(nnet)

#Applied on PCA data

mlr\_pca <- multinom(Genre~., train\_df\_pca)

confusionMatrix(predict(mlr\_pca,valid\_df\_pca[,-5]),valid\_df\_pca[,5]) confusionMatrix(predict(mlr\_pca,test\_df\_pca[,-5]),test\_df\_pca[,5])

#Applied on Standardized data mlr <- multinom(label~., train\_df\_stand)

confusionMatrix(predict(mlr,valid\_df\_stand[,-26]),valid\_df\_stand[,26]) confusionMatrix(predict(mlr,test\_df\_stand[,-26]),test\_df\_stand[,26])

```

### vi. Neural Network

```{r}

names\_of\_data <- colnames(train\_df\_norm\_binary)

Name<- as.formula(paste("label~", paste(names\_of\_data[!names\_of\_data %in% "label"], collapse = "+")))

nn\_model <- neuralnet(Name, data = train\_df\_norm\_binary,

linear.output = F, hidden = c(10,10), err.fct = "ce") plot(nn\_model)

nn\_pred <- compute(nn\_model,valid\_df\_norm\_binary[,-26]) nn\_pred\_out <- ifelse(nn\_pred$net.result>0.5, 1,0 )

confusionMatrix(as.factor(nn\_pred\_out),as.factor(valid\_df\_norm\_binary[,26]))

#test check nn\_pred\_t <- compute(nn\_model,test\_df\_norm\_binary[,-26]) nn\_pred\_out\_t <- ifelse(nn\_pred\_t$net.result>0.5, 1,0 )

confusionMatrix(as.factor(nn\_pred\_out\_t),as.factor(test\_df\_norm\_binary[,26]))

#Lift chart

gain <- gains(valid\_df\_norm\_binary$label, nn\_pred$net.result, groups=100) plot(c(0,gain$cume.pct.of.total\*sum(valid\_df\_norm\_binary$label ==

1))~c(0,gain$cume.obs), xlab="# cases", ylab="Cumulative", main="Neural Network Lift", type="l") lines(c(0,sum(valid\_df\_norm\_binary$label == 1))~c(0, dim(valid\_df\_norm\_binary)[1]), lty=2)

```

### vii. Classification Tree for Binary

```{r}

tree\_1 <- rpart(label ~ ., data = train\_df\_norm\_binary, method = "class") rpart.plot(tree\_1)

prp(tree\_1)

confusionMatrix(predict(tree\_1, valid\_df\_norm\_binary[,-26], type =

"class"),as.factor(valid\_df\_norm\_binary[,26]))

confusionMatrix(predict(tree\_1, test\_df\_norm\_binary[,-26], type =

"class"),as.factor(test\_df\_norm\_binary[,26]))

```

### viii. Random Forest for Binary

```{r}

rf\_1 <- randomForest::randomForest(as.factor(label) ~ ., data = train\_df\_norm\_binary, ntree = 700, proximity = TRUE)

confusionMatrix(predict(rf\_1, valid\_df\_norm\_binary[,-26], type =

"class"),as.factor(valid\_df\_norm\_binary[,26]))

confusionMatrix(predict(rf\_1, test\_df\_norm\_binary[,-26], type =

"class"),as.factor(test\_df\_norm\_binary[,26]))

```

### ix. Logistic Regression For Binary Data

```{r}

lr <- glm(label~., train\_df\_norm\_binary, family = "binomial") summary(lr) lr\_v<- predict(lr,valid\_df\_norm\_binary[,-26], type = "response") lr\_t <- predict(lr,test\_df\_norm\_binary[,-26], type = "response") confusionMatrix(as.factor(ifelse(lr\_v >= 0.5,1,0)),as.factor(valid\_df\_norm\_binary[,26])) confusionMatrix(as.factor(ifelse(lr\_t >= 0.5,1,0)),as.factor(test\_df\_norm\_binary[,26])) library(gains) gain <- gains(valid\_df\_norm\_binary$label, lr\_v, groups=100) plot(c(0,gain$cume.pct.of.total\*sum(valid\_df\_norm\_binary$label ==

1))~c(0,gain$cume.obs), xlab="# cases", ylab="Cumulative", main="Logistic Regression Lift", type="l") lines(c(0,sum(valid\_df\_norm\_binary$label == 1))~c(0, dim(valid\_df\_norm\_binary)[1]), lty=2)

```