USE CASE STUDY REPORT

Student name: Kumar Sri Chandra Bhaskar Adabala

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 binary classification algorithms. The data mining techniques applied in this study are K nearest neighbors, Full tree classification, Pruned tree, Random Forest, Multinomial Logistic Regression, Neural Network, Logistic regression, and Linear Discriminant Analysis. Out of all applied algorithms, Random forest performed better with both data 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 computers need to be trained by feeding data containing different features like tempo, beats, etc. of a song/music for 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.

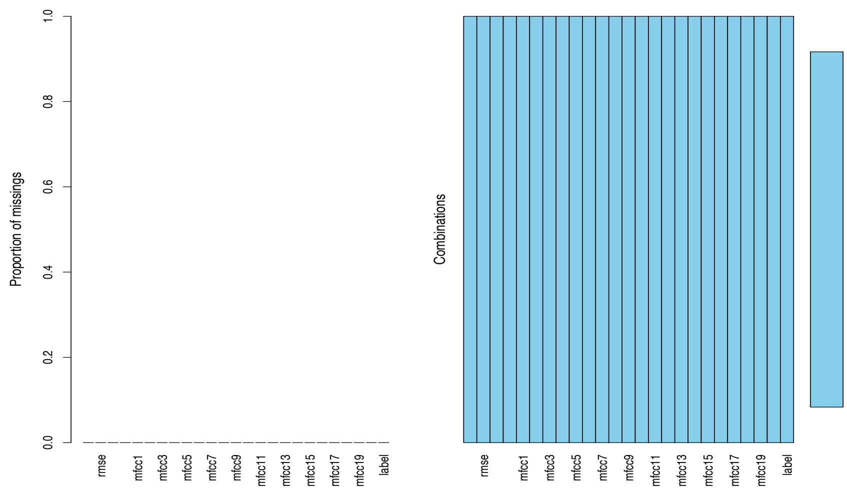
- **The possible solution**

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

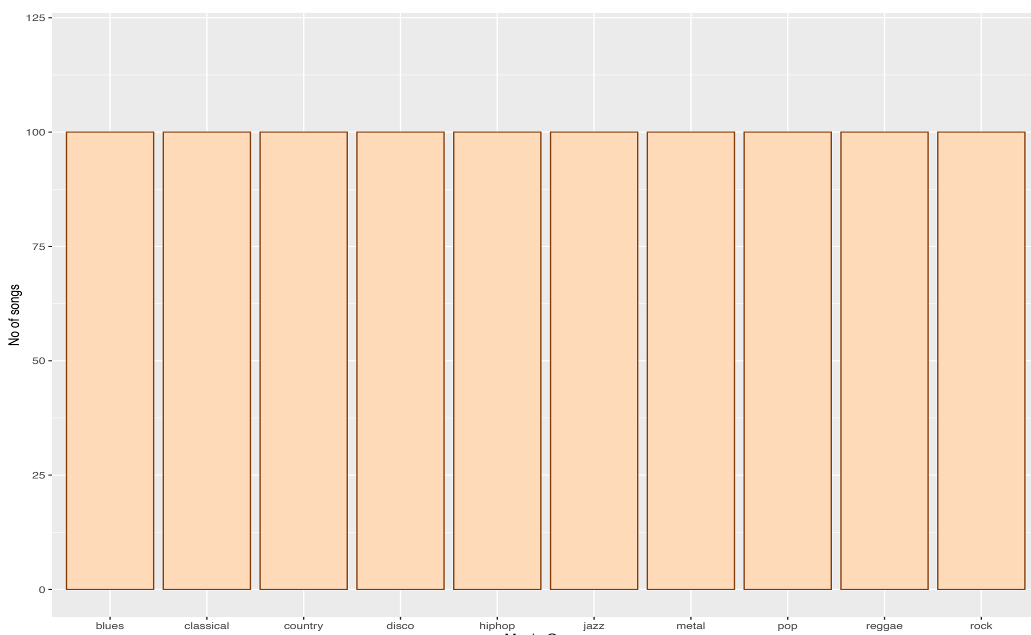
1. Finding Missing Values

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

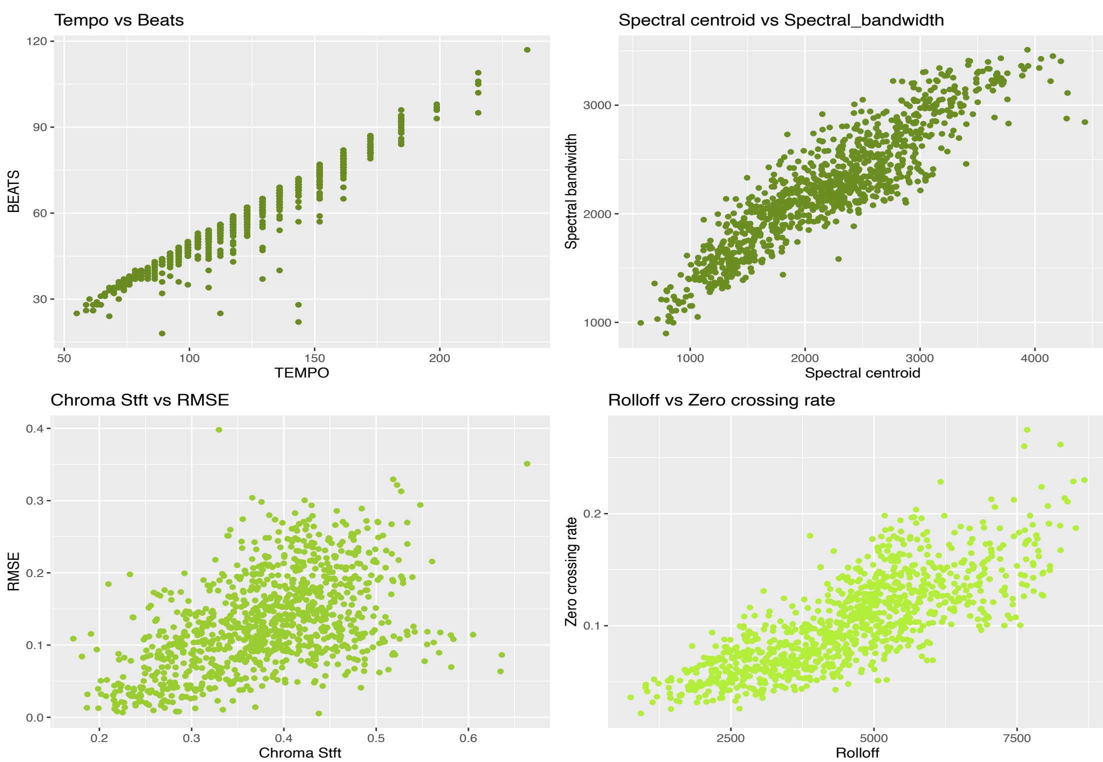


2. Visualizations

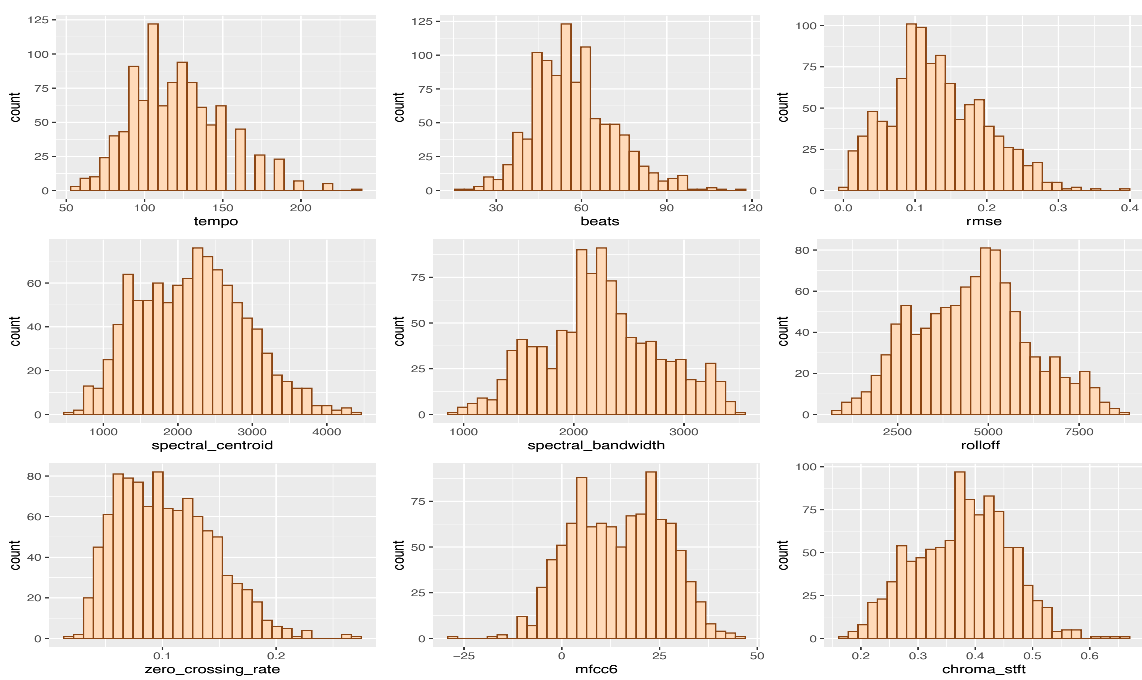
The dataset used contains equal number of observations for each genre of music, namely we are dealing with 10 different genres of music/songs (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae & rock).



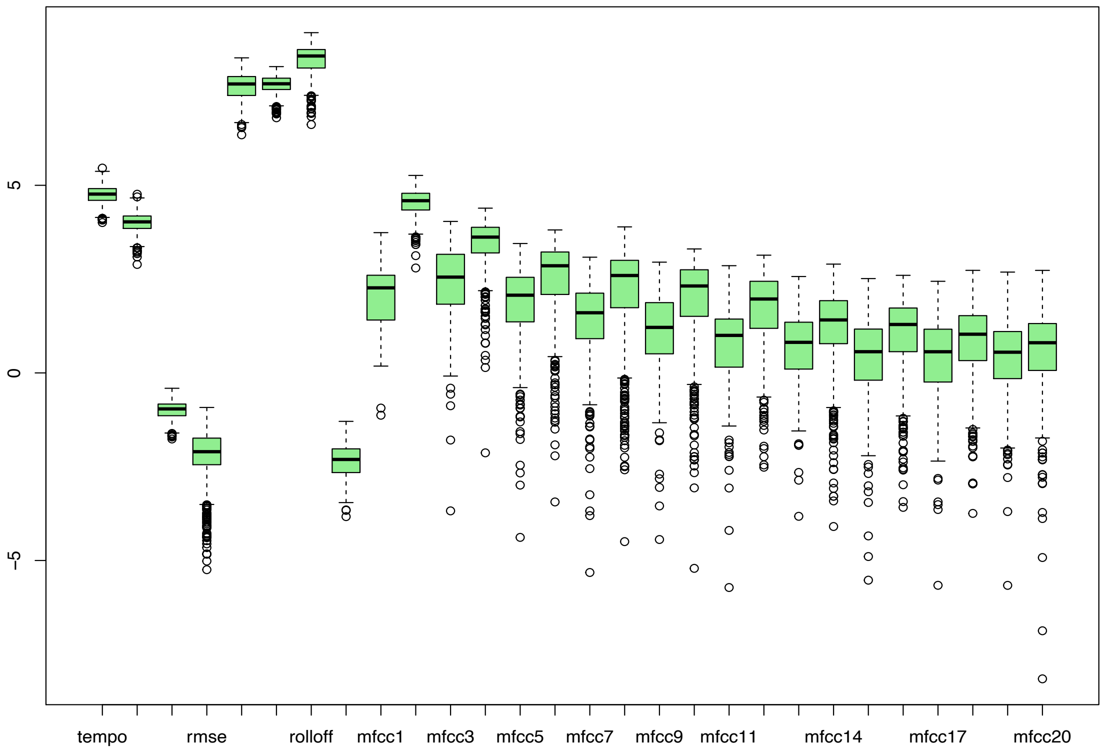
When we plotted a scatterplot for few important variables, we found a correlation between some variables such as Tempo vs Beats, Spectral centroid vs Spectral bandwidth.



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 data and the response variable(label) is categorical with 10 classes. The complete summary of the whole dataset is shown the figure below.

```
    tempo             beats          chroma_stft          rmse            spectral_centroid spectral_bandwidth
 Min.   : 54.98   Min.   : 18.00   Min.   :0.1718   Min.   :0.005276   Min.   : 569.9   Min.   : 898
 1st Qu.: 99.38   1st Qu.: 47.00   1st Qu.:0.3196   1st Qu.:0.086625   1st Qu.:1627.8   1st Qu.:1907
 Median :117.45   Median : 56.00   Median :0.3831   Median :0.122448   Median :2209.5   Median :2221
 Mean   :119.60   Mean   : 57.14   Mean   :0.3787   Mean   :0.130929   Mean   :2201.8   Mean   :2243
 3rd Qu.:136.00   3rd Qu.: 65.25   3rd Qu.:0.4360   3rd Qu.:0.175793   3rd Qu.:2692.0   3rd Qu.:2578
 Max.   :234.91   Max.   :117.00   Max.   :0.6636   Max.   :0.398012   Max.   :4434.4   Max.   :3510

   rolloff        zero_crossing_rate     mfcc1             mfcc2             mfcc3             mfcc4
 Min.   : 749.1   Min.   :0.02170    Min.   :-552.06   Min.   : -1.527   Min.   :-89.901   Min.   :-18.77
 1st Qu.:3381.0   1st Qu.:0.07028    1st Qu.:-200.70   1st Qu.: 76.811   1st Qu.:-24.224   1st Qu.: 24.11
 Median :4658.7   Median :0.09954    Median :-120.21   Median : 98.453   Median :-10.716   Median : 36.96
 Mean   :4571.7   Mean   :0.10364    Mean   :-144.48   Mean   : 99.552   Mean   : -8.922   Mean   : 36.29
 3rd Qu.:5534.2   3rd Qu.:0.13201    3rd Qu.: -73.89   3rd Qu.:119.894   3rd Qu.:  5.506   3rd Qu.: 48.21
 Max.   :8676.4   Max.   :0.27483    Max.   :  42.03   Max.   :193.097   Max.   : 56.666   Max.   : 80.69

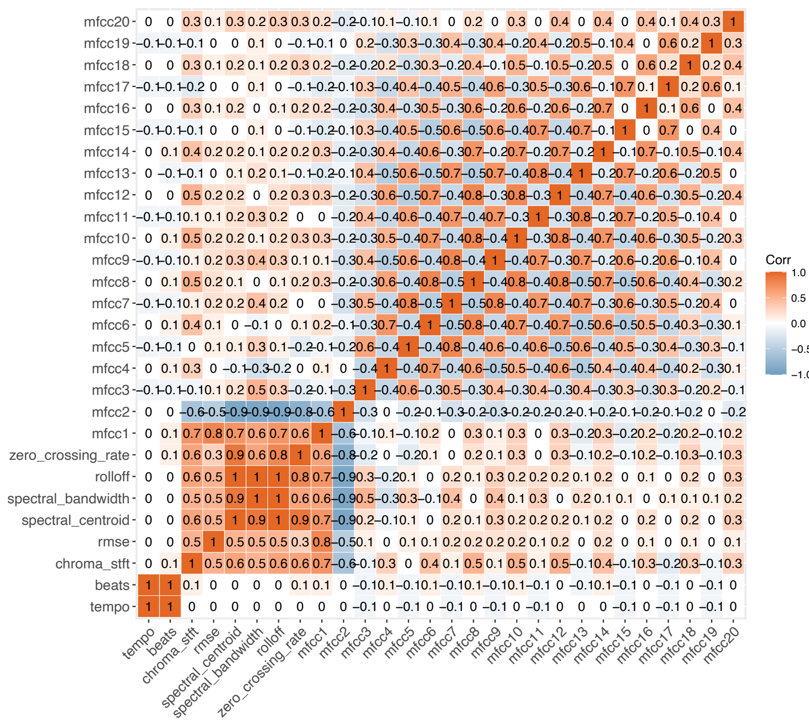
     mfcc5               mfcc6             mfcc7             mfcc8             mfcc9              mfcc10
 Min.   :-38.90345   Min.   :-28.425   Min.   :-32.934   Min.   :-24.948   Min.   :-31.6531   Min.   :-12.051
 1st Qu.: -9.97455   1st Qu.:  5.098   1st Qu.:-12.870   1st Qu.:  1.610   1st Qu.:-13.2499   1st Qu.:  1.622
 Median : -0.01524   Median : 15.008   Median : -5.717   Median :  9.664   Median : -7.5123   Median :  7.151
 Mean   : -1.14663   Mean   : 14.634   Mean   : -5.130   Mean   : 10.120   Mean   : -6.9958   Mean   :  7.730
 3rd Qu.:  7.92091   3rd Qu.: 23.858   3rd Qu.:  2.974   3rd Qu.: 18.709   3rd Qu.: -0.3834   3rd Qu.: 14.399
 Max.   : 31.46166   Max.   : 45.173   Max.   : 21.836   Max.   : 49.019   Max.   : 19.1292   Max.   : 27.217

     mfcc11            mfcc12             mfcc13            mfcc14            mfcc15             mfcc16
 Min.   :-28.052   Min.   :-15.8052   Min.   :-27.542   Min.   :-12.599   Min.   :-17.5455   Min.   :-15.694
 1st Qu.:-10.967   1st Qu.: -0.5516   1st Qu.: -9.363   1st Qu.: -1.640   1st Qu.: -7.1648   1st Qu.: -1.857
 Median : -5.920   Median :  3.8918   Median : -4.200   Median :  1.879   Median : -3.6145   Median :  1.212
 Mean   : -6.021   Mean   :  4.4716   Mean   : -4.797   Mean   :  1.782   Mean   : -3.8703   Mean   :  1.148
 3rd Qu.: -1.004   3rd Qu.:  9.7061   3rd Qu.: -0.161   3rd Qu.:  5.155   3rd Qu.: -0.3235   3rd Qu.:  4.351
 Max.   : 17.421   Max.   : 23.0376   Max.   : 13.054   Max.   : 18.162   Max.   : 12.3576   Max.   : 13.469

     mfcc17            mfcc18             mfcc19             mfcc20              label
 Min.   :-17.228   Min.   :-11.9757   Min.   :-18.5042   Min.   :-19.935   blues    :100
 1st Qu.: -7.194   1st Qu.: -2.0040   1st Qu.: -4.6703   1st Qu.: -3.368   classical:100
 Median : -4.059   Median :  0.6698   Median : -2.3913   Median : -1.155   country  :100
 Mean   : -3.967   Mean   :  0.5073   Mean   : -2.3288   Mean   : -1.095   disco    :100
 3rd Qu.: -0.843   3rd Qu.:  3.1125   3rd Qu.:  0.1491   3rd Qu.:  1.304   hiphop   :100
 Max.   : 11.490   Max.   : 15.3793   Max.   : 14.6869   Max.   : 15.369   jazz     :100
                                                                           (Other)  :400
```

2. Variable Selection

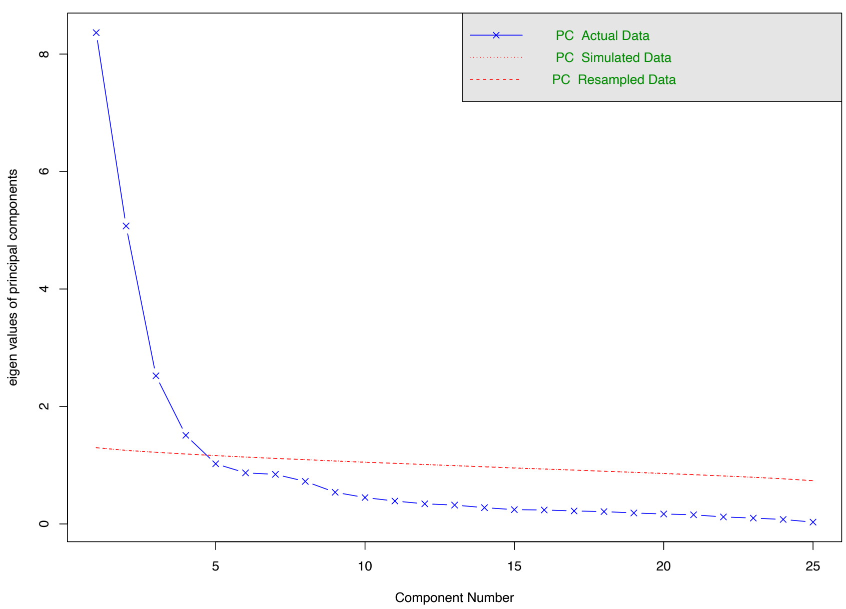
From the correlation plot, we found beats and tempo are highly correlated, so we have only considered tempo. Rolloff, spectral_centroid, and spectral_bandwidth also has a high positive correlation, so, we removed spectral_centroid and spectral_bandwidth. This left us with 25 variables excluding the response variable.



3. PCA (Principal Component Analysis)

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

Parallel Analysis Scree Plots



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

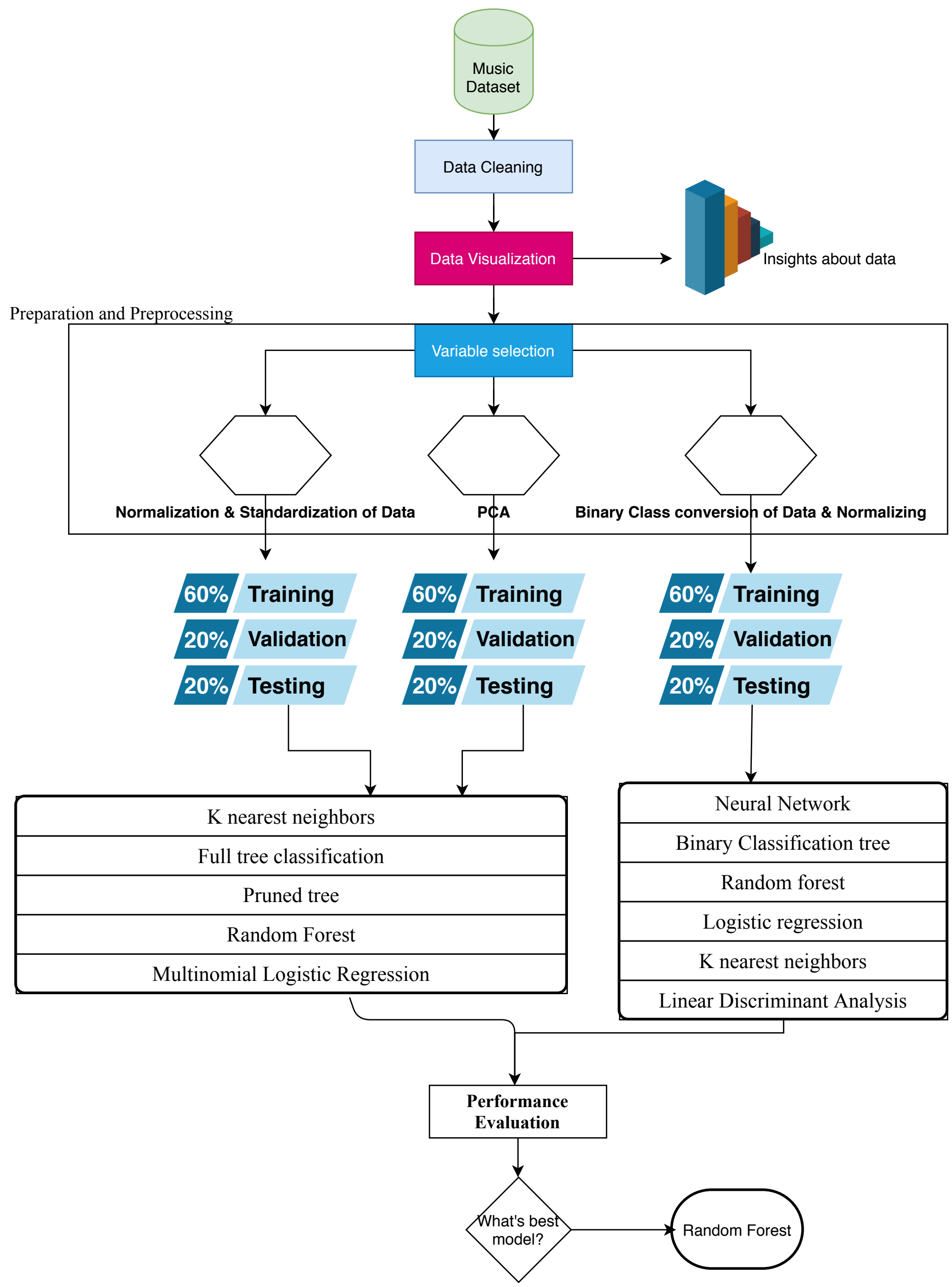
All versions of obtained data such as PCA, normalized, standardized, binary class is split into Training, Validation and Testing with 60%, 20%, 20% ratios.

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 based on the genre 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, K nearest neighbors, and Linear Discriminant Analysis.

Flow Chart for implementation



V. Performance Evaluation

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 not overfitting. The performance, confusion matrix of knn algorithm with k = 3 on validation data is shown below.

```
Confusion Matrix and Statistics

           Reference
Prediction  blues classical country disco hiphop jazz metal pop reggae rock
  blues        11         0       2     0      1    1     3   0      1    1
  classical     0        19       0     0      0    4     0   1      1    0
  country       1         0      16     2      0    1     0   0      0    2
  disco         0         1       2     4      3    0     3   2      1    3
  hiphop        0         0       0     2      9    0     1   2      0    1
  jazz          1         1       0     1      0   12     0   1      0    1
  metal         0         0       0     0      1    0    20   0      0    0
  pop           0         0       2     0      2    0     0  15      1    0
  reggae        1         0       1     3      3    1     0   1     12    1
  rock          1         0       3     6      0    1     1   0      2    4

Overall Statistics

               Accuracy : 0.61
                 95% CI : (0.5387, 0.678)
    No Information Rate : 0.14
    P-Value [Acc > NIR] : < 2.2e-16
```

2. Full Tree

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

```
Confusion Matrix and Statistics

           Reference
Prediction  blues classical country disco hiphop jazz metal pop reggae rock
  blues         9         0       2     0      0    2     0   0      0    0
  classical     0        15       1     0      0    2     1   0      0    0
  country       8         0      10     3      3    8     0   3      3    7
  disco         0         0       0     8      5    2     2   4      1    5
  hiphop        0         0       0     0      7    0     1   2      3    0
  jazz          2         0       0     0      0   11     0   0      0    1
  metal         1         0       0     0      5    0    15   0      2    0
  pop           0         0       1     0      2    1     0   8      1    1
  reggae        2         0       0     0      0    2     0   2      4    2
  rock          4         0       4     3      0    1     1   0      2    5

Overall Statistics

               Accuracy : 0.46
                 95% CI : (0.3895, 0.5317)
    No Information Rate : 0.145
    P-Value [Acc > NIR] : < 2.2e-16
```

3. Pruned Tree

When the full tree is pruned and applied on test data the obtained performance, confusion matrix is shown below.

```
Confusion Matrix and Statistics

           Reference
Prediction  blues classical country disco hiphop jazz metal pop reggae rock
  blues        11         0       3     0      0    2     1   0      0    3
  classical     0        13       1     0      0    2     0   0      0    1
  country       7         0       4     4      1    5     1   1      4    5
  disco         3         0       1     7      2    0     2   3      1    3
  hiphop        0         0       2     0     11    3     4   3      1    1
  jazz          3         2       2     0      0   15     1   0      0    2
  metal         1         0       0     0      2    0    11   0      1    0
  pop           0         0       2     0      3    1     0   9      1    1
  reggae        1         0       1     1      2    0     0   1      7    0
  rock          0         0       2     2      1    1     0   2      1    5

Overall Statistics

               Accuracy : 0.465
                 95% CI : (0.3944, 0.5367)
    No Information Rate : 0.145
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4046
```

4. Random Forest

When we applied random forest algorithm by trying different number of trees, we found that the performance is better when number of trees is 600. The performance, confusion matrix of Random forest algorithm on test data is shown below.

```
Confusion Matrix and Statistics

           Reference
Prediction  blues classical country disco hiphop jazz metal pop reggae rock
  blues        12         0       4     0      0    0     0   0      0    3
  classical     0        19       0     0      0    2     0   0      2    0
  country       2         1      12     0      0    0     0   0      1    1
  disco         0         0       0     7      4    1     1   0      0    1
  hiphop        0         0       0     2      8    0     0   1      2    0
  jazz          0         1       5     0      0   17     1   2      0    2
  metal         1         0       0     0      0    0    26   0      0    0
  pop           0         0       0     1      2    0     0  18      0    0
  reggae        0         0       2     1      5    0     0   0     13    1
  rock          0         0       3     7      0    0     0   1      0    5

Overall Statistics

               Accuracy : 0.685
                 95% CI : (0.6157, 0.7487)
    No Information Rate : 0.14
    P-Value [Acc > NIR] : < 2.2e-16
```

5. Multinomial Logistic Regression

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

```
Confusion Matrix and Statistics

           Reference
Prediction  blues classical country disco hiphop jazz metal pop reggae rock
  blues        10         1       1     0      0    2     0   0      1    4
  classical     0        19       0     0      0    4     0   0      1    0
  country       1         1      14     2      1    0     0   3      1    3
  disco         1         0       0     8      2    1     4   0      0    2
  hiphop        0         0       0     1     11    0     1   0      6    0
  jazz          1         0       2     1      0   11     0   0      1    1
  metal         1         0       0     0      0    0    22   0      0    0
  pop           0         0       0     0      2    0     0  17      1    0
  reggae        0         0       1     1      3    2     0   2      7    0
  rock          1         0       8     5      0    0     1   0      0    3

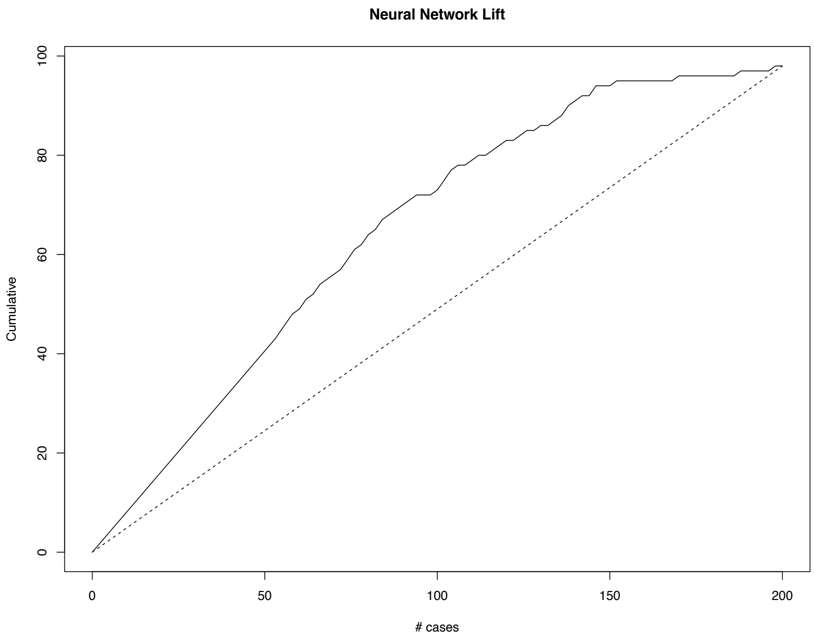
Overall Statistics

               Accuracy : 0.61
                 95% CI : (0.5387, 0.678)
    No Information Rate : 0.14
    P-Value [Acc > NIR] : < 2.2e-16
```

Evaluation of binary classification Algorithms

1. Neural Network

This algorithm is applied on the binary class converted data. We tried 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.




```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 81 24
         1 21 74

               Accuracy : 0.775
                 95% CI : (0.7108, 0.8309)
    No Information Rate : 0.51
    P-Value [Acc > NIR] : 1.057e-14

                  Kappa : 0.5495

 Mcnemar's Test P-Value : 0.7656

            Sensitivity : 0.7941
            Specificity : 0.7551
         Pos Pred Value : 0.7714
         Neg Pred Value : 0.7789
             Prevalence : 0.5100
         Detection Rate : 0.4050
   Detection Prevalence : 0.5250
      Balanced Accuracy : 0.7746

       'Positive' Class : 0
```

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.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 77 42
         1 19 62

               Accuracy : 0.695
                 95% CI : (0.6261, 0.758)
    No Information Rate : 0.52
    P-Value [Acc > NIR] : 3.619e-07

                  Kappa : 0.3946

 Mcnemar's Test P-Value : 0.00485

            Sensitivity : 0.8021
            Specificity : 0.5962
         Pos Pred Value : 0.6471
         Neg Pred Value : 0.7654
             Prevalence : 0.4800
         Detection Rate : 0.3850
   Detection Prevalence : 0.5950
      Balanced Accuracy : 0.6991

       'Positive' Class : 0
```

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 the performance is better when number of trees is 700. The performance, confusion matrix of Random forest algorithm on test data is shown below.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 80 22
         1 16 82

               Accuracy : 0.81
                 95% CI : (0.7487, 0.8619)
    No Information Rate : 0.52
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6203

 Mcnemar's Test P-Value : 0.4173

            Sensitivity : 0.8333
            Specificity : 0.7885
         Pos Pred Value : 0.7843
         Neg Pred Value : 0.8367
             Prevalence : 0.4800
         Detection Rate : 0.4000
   Detection Prevalence : 0.5100
      Balanced Accuracy : 0.8109

       'Positive' Class : 0
```

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.

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -3.6718     3.1553  -1.164 0.244541
tempo              -0.7356     0.6731  -1.093 0.274475
chroma_stft         4.6631     1.1526   4.046 5.22e-05 ***
rmse                1.1188     1.4531   0.770 0.441336
rolloff             5.7579     2.6120   2.204 0.027496 *
zero_crossing_rate -1.5894     1.7691  -0.898 0.368971
mfcc1              -0.7007     2.1390  -0.328 0.743223
mfcc2               0.9872     2.4295   0.406 0.684481
mfcc3              -0.1459     1.4682  -0.099 0.920854
mfcc4               0.6951     1.0529   0.660 0.509146
mfcc5              -3.7152     1.1718  -3.170 0.001522 **
mfcc6              -3.1060     1.2718  -2.442 0.014597 *
mfcc7               3.3592     1.1750   2.859 0.004251 **
mfcc8              -2.1364     1.7571  -1.216 0.224052
mfcc9               0.2580     1.2370   0.209 0.834761
mfcc10              1.0037     1.0962   0.916 0.359900
mfcc11             -0.7310     1.2455  -0.587 0.557276
mfcc12              0.3894     1.3063   0.298 0.765648
mfcc13             -2.8308     1.3962  -2.028 0.042602 *
mfcc14              3.7613     1.1106   3.387 0.000708 ***
mfcc15             -2.2144     1.1811  -1.875 0.060804 .
mfcc16             -0.6089     1.0604  -0.574 0.565786
mfcc17              4.1391     1.0824   3.824 0.000131 ***
mfcc18             -2.7678     1.1716  -2.362 0.018158 *
mfcc19              2.0576     1.3005   1.582 0.113632
mfcc20              2.0990     1.2500   1.679 0.093119 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 831.75  on 599  degrees of freedom
Residual deviance: 590.11  on 574  degrees of freedom
AIC: 642.11
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 81 30
         1 21 68

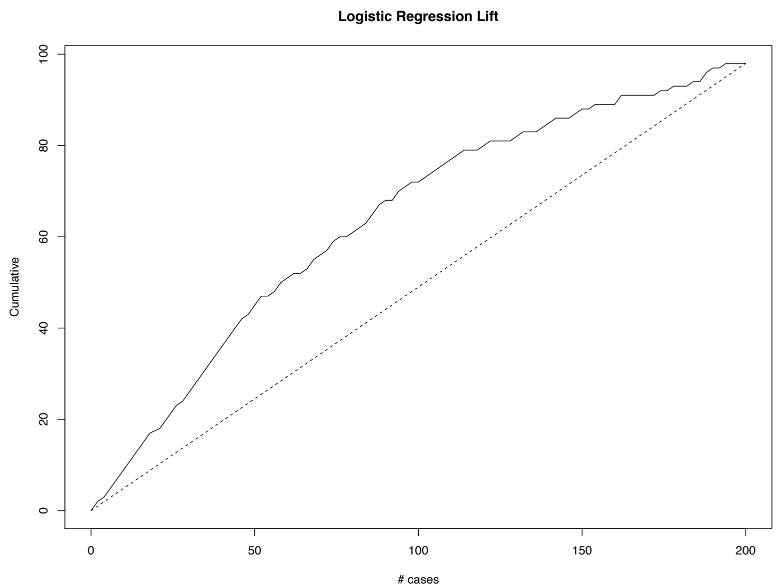
               Accuracy : 0.745
                 95% CI : (0.6787, 0.8039)
    No Information Rate : 0.51
    P-Value [Acc > NIR] : 8.707e-12

                  Kappa : 0.4889

 Mcnemar's Test P-Value : 0.2626

            Sensitivity : 0.7941
            Specificity : 0.6939
         Pos Pred Value : 0.7297
         Neg Pred Value : 0.7640
             Prevalence : 0.5100
         Detection Rate : 0.4050
   Detection Prevalence : 0.5550
      Balanced Accuracy : 0.7440

       'Positive' Class : 0
```



5. K Nearest Neighbors

This algorithm is applied on the binary class converted data. When we applied KNN by trying different k values, we found that the performance is better when k is 3 without overfitting. The performance, confusion matrix of KNN algorithm on validation data is shown below.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 91 30
         1 11 68

               Accuracy : 0.795
                 95% CI : (0.7323, 0.8487)
    No Information Rate : 0.51
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5883

 Mcnemar's Test P-Value : 0.004937

            Sensitivity : 0.8922
            Specificity : 0.6939
         Pos Pred Value : 0.7521
         Neg Pred Value : 0.8608
             Prevalence : 0.5100
         Detection Rate : 0.4550
   Detection Prevalence : 0.6050
      Balanced Accuracy : 0.7930

       'Positive' Class : 0
```

6. Linear Discriminant Analysis

This algorithm is applied on the binary class converted data. The probabilities, performance, confusion matrix of LDA algorithm on validation data is shown below.

```
            0           1
184 0.91092683 0.089073170
723 0.13816427 0.861835728
63  0.98594273 0.014057271
370 0.58888823 0.411111771
596 0.86017256 0.139827439
628 0.46223932 0.537760683
65  0.99072309 0.009276912
333 0.34756547 0.652434529
282 0.88178975 0.118210245
797 0.15552942 0.844470579
616 0.26582144 0.734178560
900 0.90622475 0.093775247
989 0.80311081 0.196889193
285 0.63586534 0.364134657
38  0.52904498 0.470955024
958 0.74076282 0.259237175
122 0.90425679 0.095743211
368 0.33168277 0.668317232
570 0.88918356 0.110816436
185 0.96560718 0.034392816
465 0.20456020 0.795439801
66  0.98139596 0.018604036
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 81 29
         1 21 69

               Accuracy : 0.75
                 95% CI : (0.684, 0.8084)
    No Information Rate : 0.51
    P-Value [Acc > NIR] : 3.039e-12

                  Kappa : 0.499

 Mcnemar's Test P-Value : 0.3222

            Sensitivity : 0.7941
            Specificity : 0.7041
         Pos Pred Value : 0.7364
         Neg Pred Value : 0.7667
             Prevalence : 0.5100
         Detection Rate : 0.4050
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.7491

       'Positive' Class : 0
```

Best Approach

The best approach for this problem is to use Random Forest algorithm because the performance of this model is very high when compared to all other algorithms with 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, the chance of overfitting is also low. 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%.

Accuracies Tables

1. **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%

2. **Binary classification Algorithms**

No.	Algorithms	Accuracy
1	Neural Network	77.5%
2	Binary classification tree	69.5%
3	Random Forest	81%
4	Logistic Regression	74.5%
5	K Nearest Neighbors	79.5%
6	Linear Discriminant Analysis	75%

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 by classifying with multiclass classification supported algorithms and other by converting the 10 classes in traditional music and modern music. We got good accuracies for KNN, Multinomial logistic regression and Random forest models with 61%, 61% and 68.5% compared to others for multiclass response variable. We got good accuracies for Random forest, KNN and Neural network models with 81%, 79.5% and 77.5% for 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 Binary 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 based on the music features is performed in this use case. The experimenting results obtained by applying many data mining supervised classification algorithms show look promising, these models can be used by digital music platforms to make playlists for their users based on their preference of music genre. These models save a lot of time to identify the genre of music effortlessly.

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)
library(DiscriMiner)
```

```{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])


k_binary <- knn(train = train_df_norm_binary[,-26], test = valid_df_norm_binary[,-26],cl = train_df_norm_binary$label, k =3)
confusionMatrix(k_binary,as.factor(valid_df_norm_binary$label))
```

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)

```

X. Linear discriminant analysis

```{r}
library(MASS)
````

```
lda_model <- MASS::lda(label~., train_df_norm_binary)

lda_pred <- predict(lda_model, valid_df_norm_binary[,-26])

confusionMatrix(lda_pred$class, as.factor(valid_df_norm_binary[,26]))
```