Music Mood Classification

Introduction To Machine Learning

Naveen Kumar Mishra

Contents

FIGURE TABLE

1. INTRODUCTION

In this report, first of all, information about the data set to be used is given and the pre-processing stages of this data are explained. The Clustering method and algorithms used to make the data set suitable for modeling are explained by visualization. Then, the models used to classify the data set are explained. In addition, analyzes were made on these models. As a result, the most appropriate classification method for the data set used was selected by looking at the metrics.

1.1 Getting Familiar with the Dataset

The dataset consists of 15 numerical attributes and 4 categorical attributes making up to 19 total columns. The raw dataset contains *170,654* rows which are representing unique songs. The dataset does not come with a target feature hence, as the main purpose of the project, after clustering the songs and labeling them into 6 different moods, the target feature column shall be acquired. Further explanation of the features is below:

Name	Description
valence	The positiveness of the track
year	The release year of the track (1921 - 2020)
acousticness	The relative metric of the track being acoustic
artists	The list of artists credited for the production of the track
danceability	The relative measurement of the track being danceable
duration_ms	The length of the track in milliseconds (ms)
energy	The energy of the track
explicit	The binary value whether the track contains explicit content or not
id	The primary identifier for the track, generated by Spotify
instrumentalness	The relative ratio of the track being instrumental
key	The primary key of the track encoded as integers between 0 and 11
liveness	The relative duration of the track sounding as a live performance
loudness	Relative loudness of the track in the typical range [-60, 0] in decibel (dB)
mode	The binary value representing whether the track starts with a major (1) chord progression or not (0)
name	The title of the track
popularity	The popularity of the song lately, default country = the US
release_date	The date of release of the track in yyyy-mm-dd, yyyy-mm
speechiness	The relative length of the track containing any kind of human voice
tempo	The tempo of the track in Beat Per Minute (BPM)

Figure 1.1.1 Description of the Dataset

The key features while labeling song moods are valence, acousticness, danceability, energy, instrumentalness, loudness, speechiness, and tempo. The values are already normalized for these features in the raw dataset within the range 0 to 1. (Except for tempo and loudness)

The year feature ranges from 1921 to 2020 thus the dataset consists of songs released between these two boundary years.

1.2 Exploratory Data Analysis

First of all, the dataset needs to be explored in order to gain more insight into the data itself. Missing values are checked if there exists any.

```
valence             year     acousticness          artists     danceability      duration_ms
      0                0                0                0                0                0
 energy         explicit               id instrumentalness              key         liveness
      0                0                0                0                0                0
loudness             mode             name       popularity     release_date      speechiness
      0                0                0                0                0                0
   tempo
      0
```

Figure 1.2.1 Checking Missing Values

As can be seen from the output, there are no missing values in any of the columns in the dataset. Duplicate values should be checked as well since if there exists any, the extra rows should be removed. However, there are no duplicate rows in the dataset. The dataset is checked for outliers such as years earlier than 1921 or values greater than 1 for features where they are normalized. As a result, none of these issues are found thus the raw dataset is ready to be used.

1.3 Data Visualization

Histograms for the descriptive features are shown below. It can be seen from the plots as well that there are no outliers and most of them have normalized values between 0 and 1.

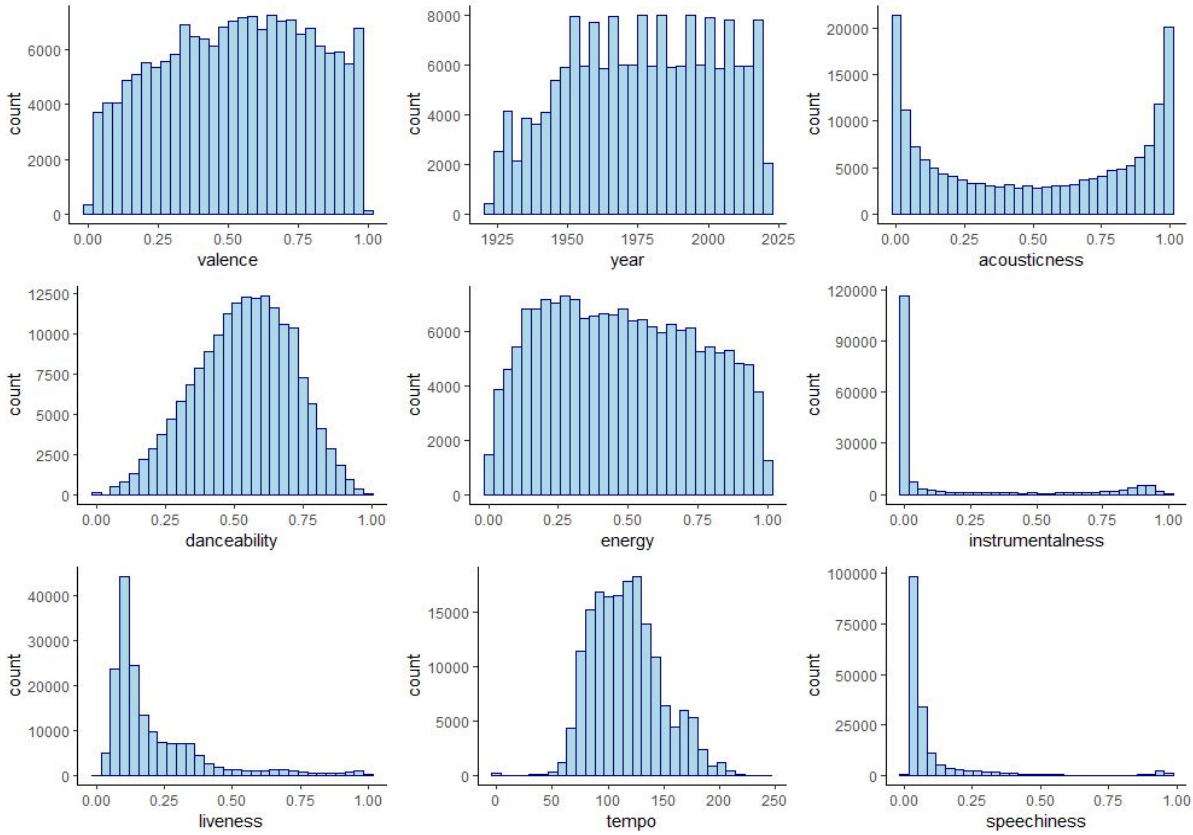

Figure 1.3.1 Histogram for Descriptive Features

Since the clustering is going to be applied to the raw data to label the song moods, the correlation matrix is crucial while deciding the descriptive features to be included or not. The correlation heat map is shown below:

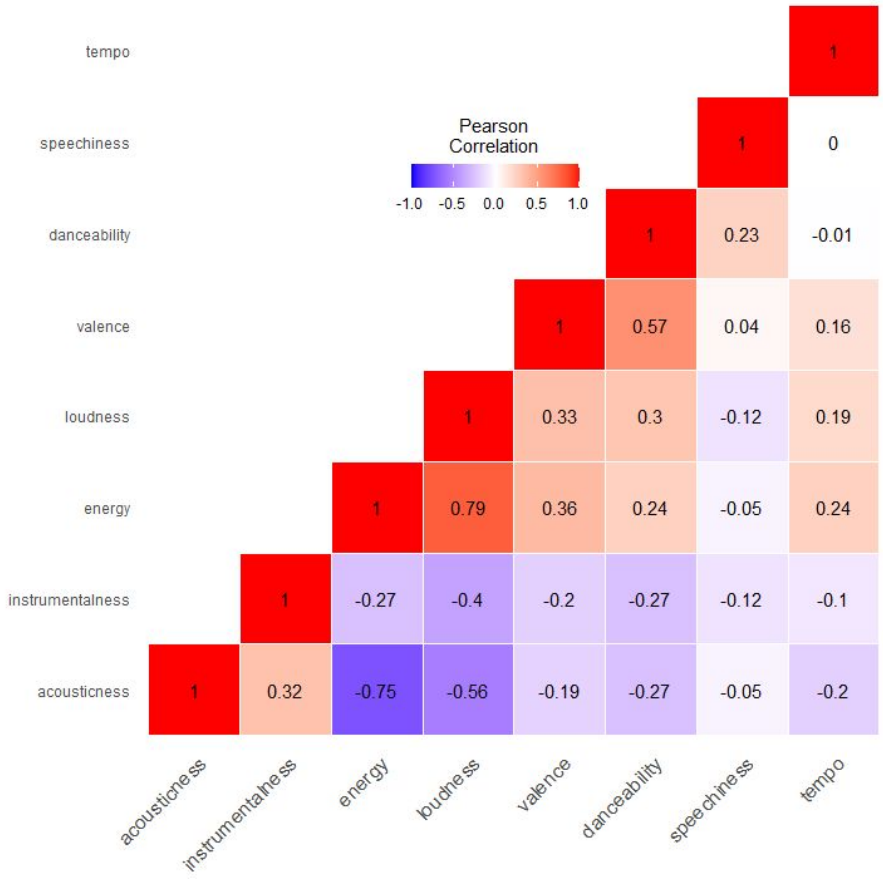

Figure 1.3.2 Heat Map

There seems to be a strong correlation between energy and loudness so any of them could be dropped to get better results in clustering. However, our works have shown that using both features grants more information towards clustering so all of the descriptive features are kept.

2. DATA PREPARATION

2.1 Data Cleaning

2.1.1 Remove Missing Values

There exist no missing values in the dataset thus if there exist any duplicate rows or outliers, they should be removed.

```
colSums(is.na(music_data))
```

yields the result:

```
 valence             year     acousticness          artists     danceability      duration_ms
       0                0                0                0                0                0
  energy         explicit               id instrumentalness              key         liveness
       0                0                0                0                0                0
loudness             mode             name       popularity     release_date      speechiness
       0                0                0                0                0                0
   tempo
       0
```

Figure *2.1.1.1 Checking Missing Values*

2.1.2 Remove Duplicate Values

There exist no duplicate rows in the dataset either. The raw data can be used safely for clustering purposes.

```
sum(duplicated(music_data[]))
```

yields the result:

```
[1] 0
```

As the plots show above, there are no outliers in the dataset. Most of the columns are already normalized and the other columns contain safe values. So the entire dataset is used for clustering purposes.

3. CLUSTERING DATA

As mentioned before, the raw dataset was clean in terms of missing values, outliers, and duplicate values. However, the dataset still lacks a target feature. As the project's first main goal, the data is clustered using two different clustering techniques K-means and Hierarchical clustering (hclust) to create the "mood" target feature column.

3.1 K-Means Clustering Algorithm

K-means clustering groups similar kinds of data points in the form of clusters. It finds the similarity between the data points and groups them into clusters. K-means clustering works in three steps:

1- Select the k values.
2- Initialize the centroids.
3- Select the group and find the average.

The above steps could be explained more in detail with the help of this figure:

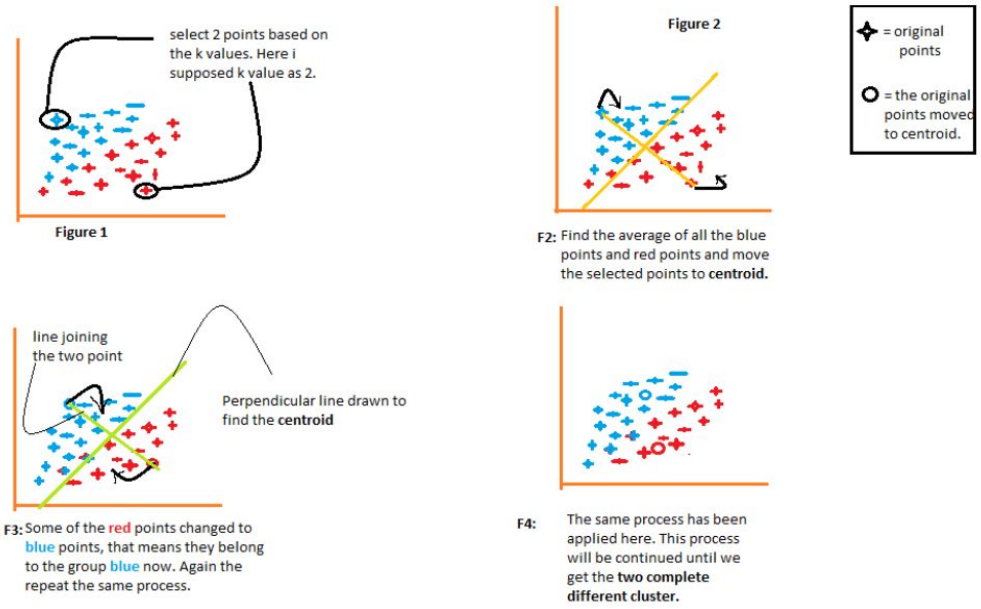

Figure 3.1.1 K-Means Steps [1]

k-means is very simple to implement and scalable to huge datasets. However, it is sensitive to outliers, and managing the selection of the right k value is a tough job. K-means has pros that are valuable for our purposes so the algorithm is going to be applied to the dataset.

3.2 Hierarchical Clustering Algorithm

Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis in data mining and statistics that seeks to construct a cluster hierarchy. In general, strategies for hierarchical clustering fall into two types:

- Agglomerative: This is a "bottom-up" approach: each observation begins in its cluster, and as one moves up the hierarchy, pairs of clusters are combined.
- Divisive: This is a "top-down" approach: all observations begin in one cluster, and as one moves down the hierarchy, splits are carried out recursively.

In general, the merges and splits are determined greedily. The results of hierarchical clustering are usually presented in a dendrogram.

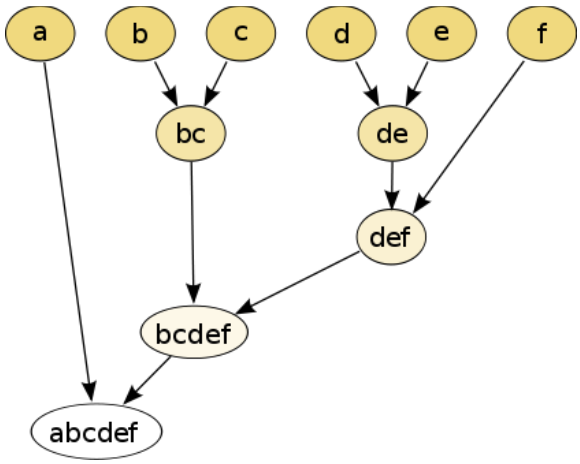

Figure 3.2.1 Hierarchical Clustering Dendrogram 1 [2]

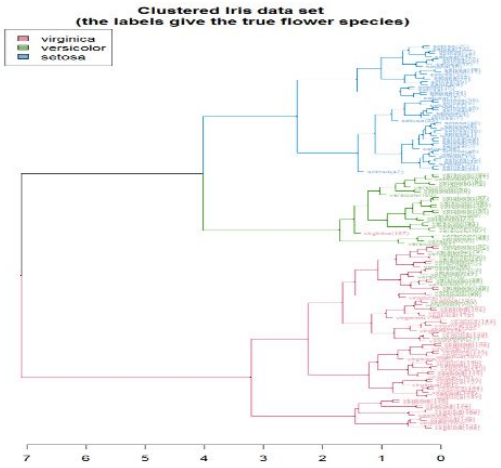

Figure 3.2.2 Hierarchical Clustering Dendrogram 2 [2]

For clustering purposes, a package called "*eclust*" is used which contains both k-means and hclust. The library itself is not very fast however gives the most accurate clusters.

The first step is to subsample our 170,000 rows of data into 18,000 samples. Clustering 170,000 data is not possible since it requires greater computation power hence 100 different subsamples with the size of 18,000 from the dataset are taken and a subsample list is created.

For every subsample of 18,000 data, k-means and hclust algorithms are applied to create the clusters. It is also important to mention that k = 6 is used for our purposes since it's planned to have 6 different moods for songs.

The goodness of the clusters had to be measured as well. The Silhouette Scores of each clustering result are calculated and compared. Before showing the plots the Silhouette Score is explained briefly.

3.3 Silhouette Score

Silhouette Coefficient or Score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

- 1 means that clusters are well apart from each other and distinguished.
- 0 Means that clusters are indifferent or the distance between the clusters is not significant
- -1 Means that clusters are assigned to the wrong clusters.

Silhouette Score = (b-a)/max(a,b)

- a= average intra-cluster distance i.e the average distance between each point within a cluster.
- b= average inter-cluster distance i.e the average distance between all clusters.

So it can be deduced that the higher the Silhouette Score is the better the clusters.

The silhouette scores of 100 samples for each clustering results can be seen in the plots below:

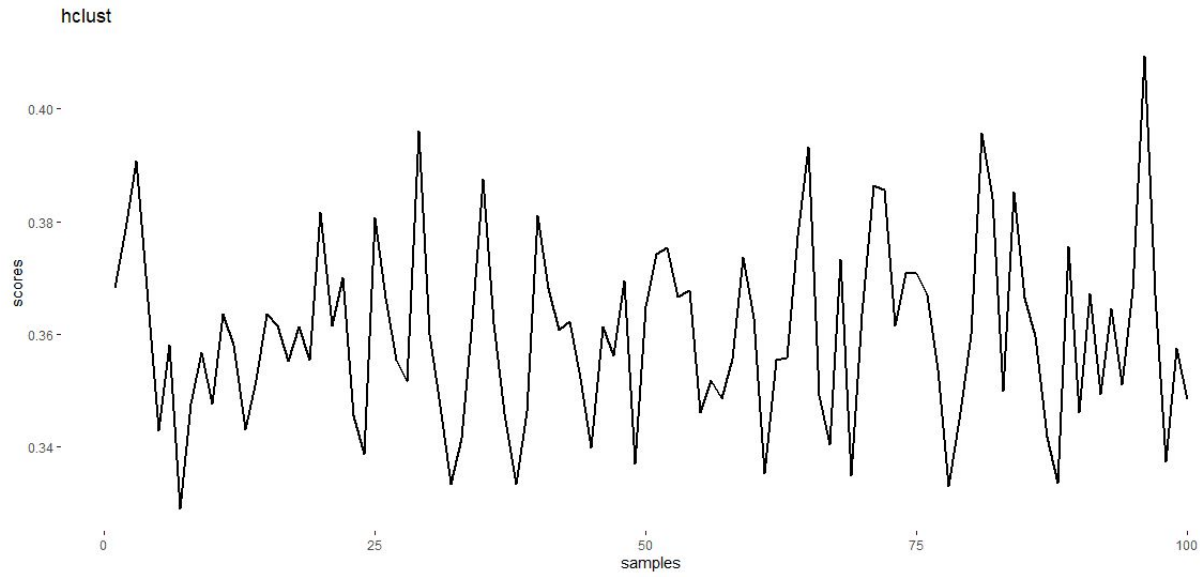

Figure *3.3.1 Silhouette Scores of 100 samples for hclust*

The highest silhouette score for hclust is the 96th sample with 0.409287.

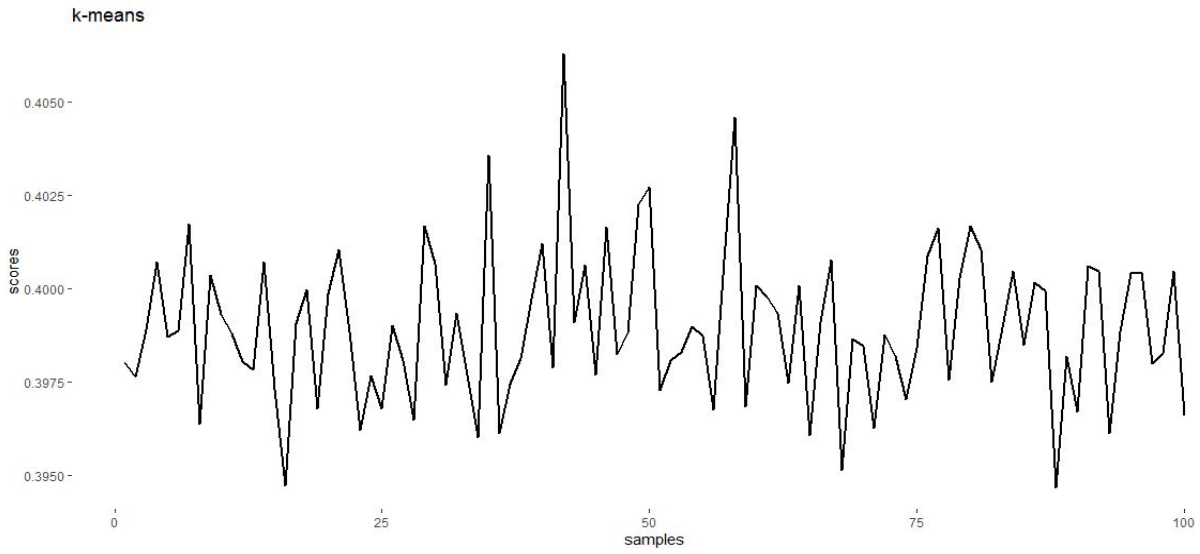

Figure 3.3.1 Silhouette Scores of 100 samples for k-means

The highest silhouette score for k-means is the 42nd sample with 0.4062914.

The scores of the clustering algorithms are fairly close, however, the overall average of the scores is higher for hclust so the 96th sample is chosen and clustered with hclust. For further purposes, hierarchical clustering with "complete" linkage is used.

3.4 Mood Creation

The clusters with their average values exist but the moods have to be assigned to them so the target feature can be created. Thus by giving proper mood names to the clusters based on their key feature attributes, the target feature column is acquired. Analyzing the 6 clusters we have:

HAPPY	
CLUSTER 1	VALUES
acousticness	0.1610
energy	0.6869
danceability	0.5963
valence	0.6007
instrumentalness	0.0247
speechiness	0.0841
tempo	0.4998
loudness	0.8139

GLEEFUL	
CLUSTER 2	VALUES
acousticness	0.8083
energy	0.3737
danceability	0.5721
valence	0.6509
instrumentalness	0.0388
speechiness	0.0813
tempo	0.4854
loudness	0.7541

SAD	
CLUSTER 3	VALUES
acousticness	0.8092
energy	0.2245
danceability	0.3827
valence	0.2369
instrumentalness	0.0753
speechiness	0.0448
tempo	0.4394
loudness	0.7046

CALM	
CLUSTER 4	VALUES
acousticness	0.9193
energy	0.2373
danceability	0.4279
valence	0.4033
instrumentalness	0.8472
speechiness	0.0591
tempo	0.4433
loudness	0.6617

ROMANTIC	
CLUSTER 5	VALUES
acousticness	0.5151
energy	0.2474
danceability	0.6755
valence	0.5204
instrumentalness	0.0022
speechiness	0.9151
tempo	0.4492
loudness	0.6451

SENSUAL	
CLUSTER 6	VALUES
acousticness	0.1200
energy	0.6880
danceability	0.5554
valence	0.5125
instrumentalness	0.7780
speechiness	0.0592
tempo	0.5025
loudness	0.7797

Figure 3.4.1 Six Clusters

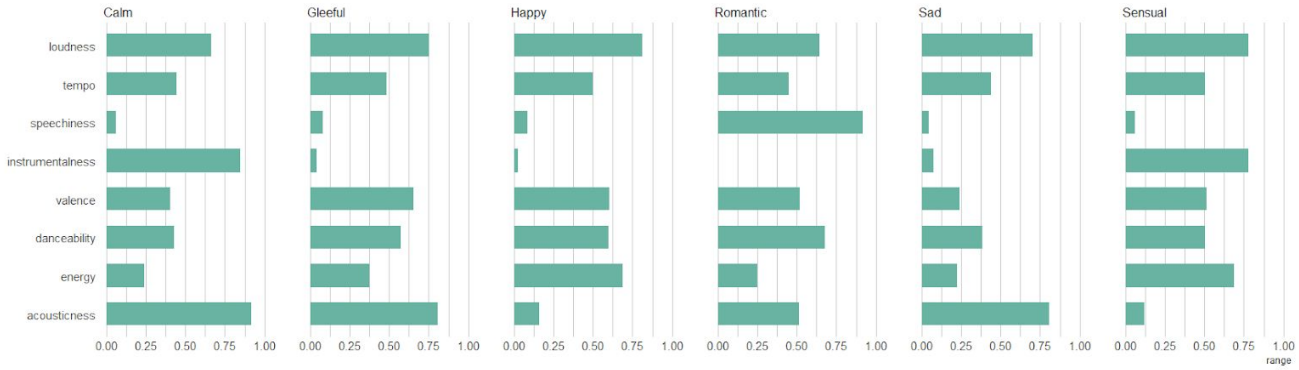

Figure *3.4.2 Mood Based Descriptive Features*

The differences can be seen between each mood based on the descriptive features.

As a result, the data is labeled with 18,000 rows and 9 columns. Before moving onto the classification chapter the count of the moods should be checked in the dataset.

There are exactly:

- 8027 happy songs
- 3989 gleeful songs
- 2495 sad songs
- 2362 calm songs
- 484 romantic songs
- 643 sensual songs

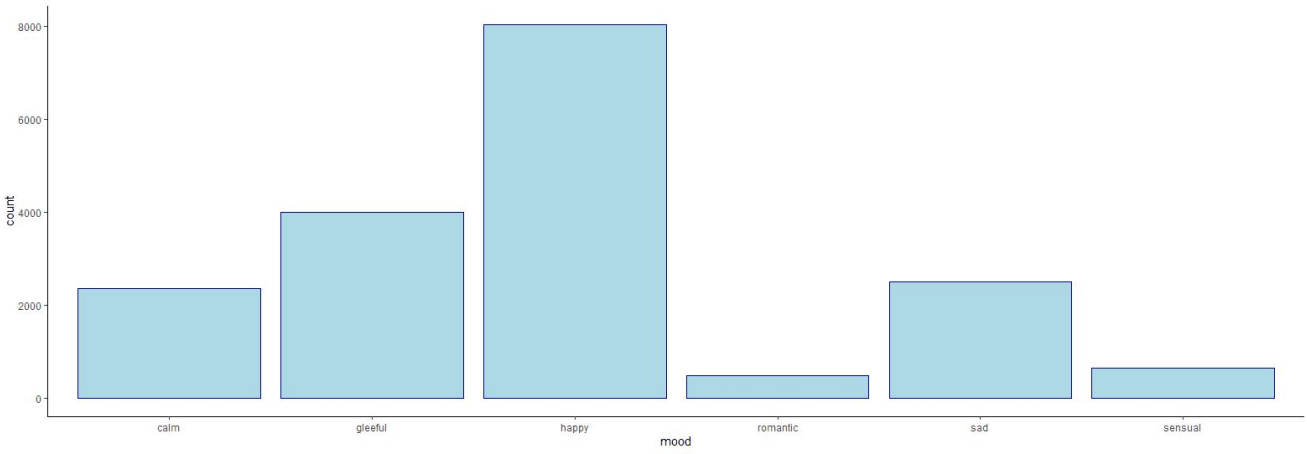

Figure 3.4.8 Count of Moods

And 18,000 in total songs in the dataset. The labels seem to be imbalanced at the first glance however, our research showed that leaving the dataset as is yields higher accuracy while applying classification techniques so under-sampling is not needed for our dataset.

4. SUPERVISED MODELS

Caret library in r was used for all models in the study. The Caret library is used for teaching regression and classification models. This library has other technical functions besides classification. The train function in the library performs the cross-validation process by using the "trControl" function. In the study, the cross-validation number was determined as five, so data is trained five times at a rate of 0.8. Modeling specific methods are used in the Train library, and these methods contain some parameters. Train function evaluates the effect of parameters on performance and chooses the most suitable among these parameters.

4.1 Decision Tree Model

Decision trees are used for both classification and regression problems, the decision tree is used for classification in this project. The decision tree is a tree in which each node represents a property (attribute), each link (branch) a decision (rule), and each leaf represents a result (categorical or continuous value). The whole idea is to create a tree-like this for all data and process a single result on each leaf (or minimize the error on each leaf)[3].

In the study, the "rpart2" method of train function was used while creating the decision tree model. There is "max tree depth" as a parameter in this method. Selects the deepest tree with the best accuracy for the dataset used by tuning through the max tree depth parameter in the modeling train function.

MaxDepth	Accuracy	Kappa
1	0.6383334	0.4575126
2	0.7626110	0.6500562
3	0.8942781	0.8512914

Figure 4.1.1 Accuracies of MaxDepths

The cross-validation method was applied to the Decision Tree to find the optimal tree depth. With this method, each tree depth is calculated for 5 times the value and the accuracy value of the bend with the highest accuracy is selected. The table above shows 3 tree depths with the highest accuracy. When tree depth 1 was selected, an accuracy of 0.6383334 was obtained, when tree depth 2 was selected, a value of 0.7626110 was obtained, and finally, when tree depth 3 was selected, 0.8942781 was obtained. As can be seen in the table below, the tree depth with the highest accuracy is 3. MaxDepth = 3 with the highest precision was chosen to find the optimal tree depth.

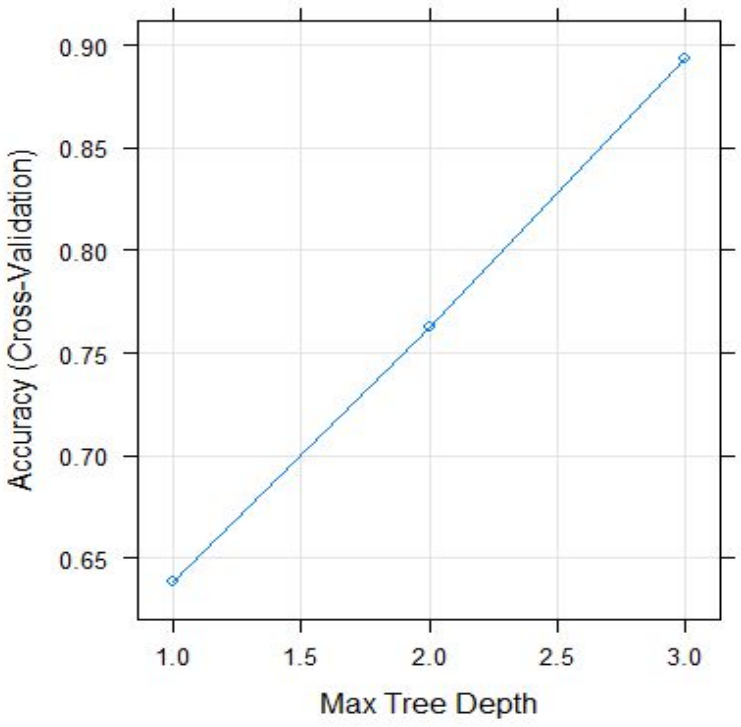

Figure 4.1.2 Tuning Plot of Decision Tree Model

Accuracy	0.8942781
Error-rate	0.1057219
Precision	0.825741
Recall	0.8954792
F1-score	0.8469063
Run Time	2.281689 secs

Figure 4.1.3 Metrics of Decision Tree

4.2 Random Forest Model

Random Forest generates hundreds of decision trees randomly. In each decision tree, predictions are made, and the best one is chosen from among those found. Unlike the decision tree model, overfit is blocked in the Random Forest model [4].

While doing Random Forest modeling in the study, the "rf" method within the train function was used. This method includes the "number of randomly selected predictors (mtyr)" as parameters. Train function performs tune operation according to the "mtyr" parameter in the Random Forest model and chooses the model with the best "mtyr" parameter. In this modeling, cross-validation was not used because many trees were created randomly.

mtry	Accuracy	Kappa
2	0.9502779	0.9301746
5	0.9554446	0.9374933
8	0.9543893	0.9360233

Figure 4.2.1. Accuracy Rates for mtry Value

The table above contains the tuning results of the parameter "number of randomly selected predictors (mtyr)". The tuning process in modeling is done separately for five-folds and the highest fold accuracy value is accepted as accuracy. Looking at the table, the mtry value with the best accuracy value was found to be five as a result of the tuning process. The accuracy rate obtained with this parameter is 0.9554446 and the kappa ratio is 0.9374933. In the plot below, there is a visualization of the mtyr values used.

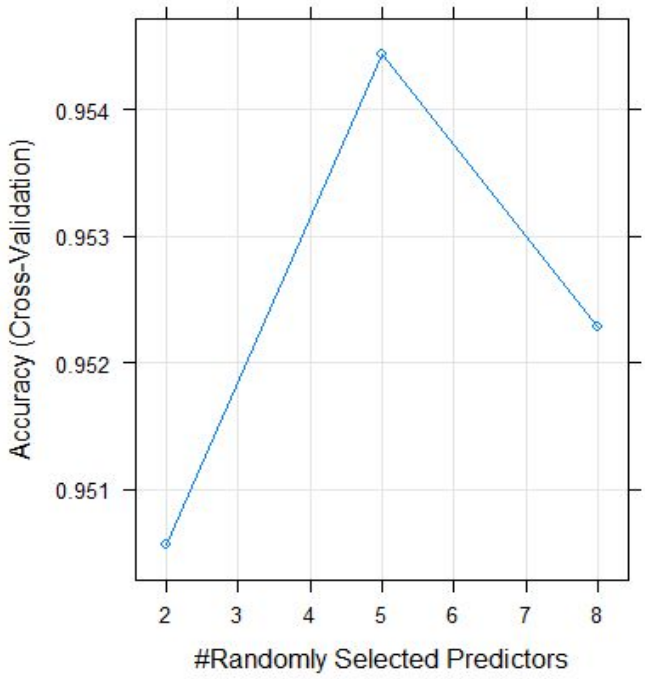

Figure 4.2.2 Tuning Plot of Random Forest Model

Accuracy	0.9554446
Error-rate	0.0445554
Precision	0.9489857
Recall	0.951534
F1-score	0.9502304
Run Time	2.731373 mins

Figure 4.2.3 Metric of Random Forest Model

4.3 K-Nearest Neighbors Model

The K-nearest neighbors (KNN) algorithm is a simple, easy to implement, supervised machine learning algorithm that can be used to solve both classification and regression problems. In the study, the KNN algorithm was used for classification. A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on tagged input data to learn a function that produces an appropriate output when new untagged data is given[5].

In K-Nearest Neighbors modeling, the "kknn" method was used in the train function. The use case of "k value", "distance value" and "kernel" to be used in this method is used as parameters. Train function finds the best k value, distance value, which is the most important parameter for K-Nearest Neighbors modeling, by applying tune on the model. Thus, values with the highest accuracy are obtained.

kmax	Accuracy	Kappa
5	0.9620554	0.946788
7	0.9632775	0.948506
9	0.9641109	0.949661

Figure 4.3.1 Accuracy Rate for kmax Value

The table above contains the values used to find the best k value for K-Nearest Neighbors modeling. This process was done on cross-validation and individual accuracy values over five folds were calculated. Looking at the table, the final values for the data set used as a result of the tuning process are k max = 9, distance = 2, and kernel = optimal. With these values, the accuracy rate is 0.9641109 and the kappa ratio is 0.949661. The plot below includes varying accuracy rates while finding the optimum k value.

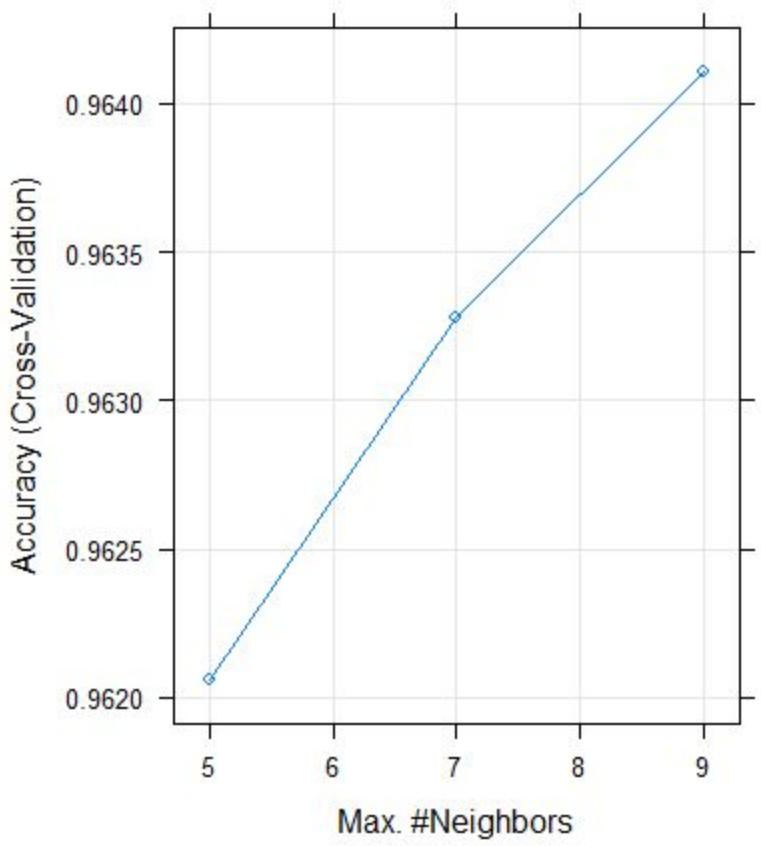

Figure 4.3.2 Tuning Plot of K-Nearest Neighbors Model

Accuracy	0.9641109
Error-rate	0.0358891
Precision	0.9608045
Recall	0.961765
F1-score	0.9612576
Run Time	2.404891 mins

Figure 4.3.3 Metrics for K-Nearest Neighbors Model

4.4 Support Vector Machine Model

Support Vector Machine is a classification algorithm similar to Logistic Regression. Both try to find the best line that separates the two classes. It is a classifier that takes no parameters (nonparametric). SVM can also classify linear and nonlinear data but generally tries to classify the data linearly[6].

In the Support Vector Machine modeling, "lssvmRadial" is used as a method in the train function. This method includes "sigma" and "tau" values as parameters. The most suitable sigma and tau values are selected for the data set used to find the highest accuracy.

Sigma	Tau	Accuracy	Kappa
0.03059712	0.0625	0.9305557	0.9021530
0.03059712	0.1250	0.9307224	0.9024013
0.03059712	0.2500	0.9302221	0.9017087
0.13871277	0.0625	0.9282223	0.8989964
0.13871277	0.1250	0.9283334	0.8991533
0.13871277	0.2500	0.9283334	0.8991524
0.24682843	0.0625	0.9125560	0.8765613
0.24682843	0.1250	0.9126115	0.8766414
0.24682843	0.2500	0.9128893	0.8770387

Figure 4.4.1 Accuracy Rate for Sigma and Tau Values

In order to find the highest accuracy sigma and tau values, the support vector machine has been cross-validated. K-fold cross-validation was applied for each sigma and tau value and the highest accuracy value was found. Sigma and tau values with the highest accuracy are shown in the table above. The table below has been visualized in order to show the highest accuracy among these values. As can be seen from both tables, the highest accuracy is 0.9307224. For this reason, the value of sigma 0.03059712 tau 0.125 was chosen.

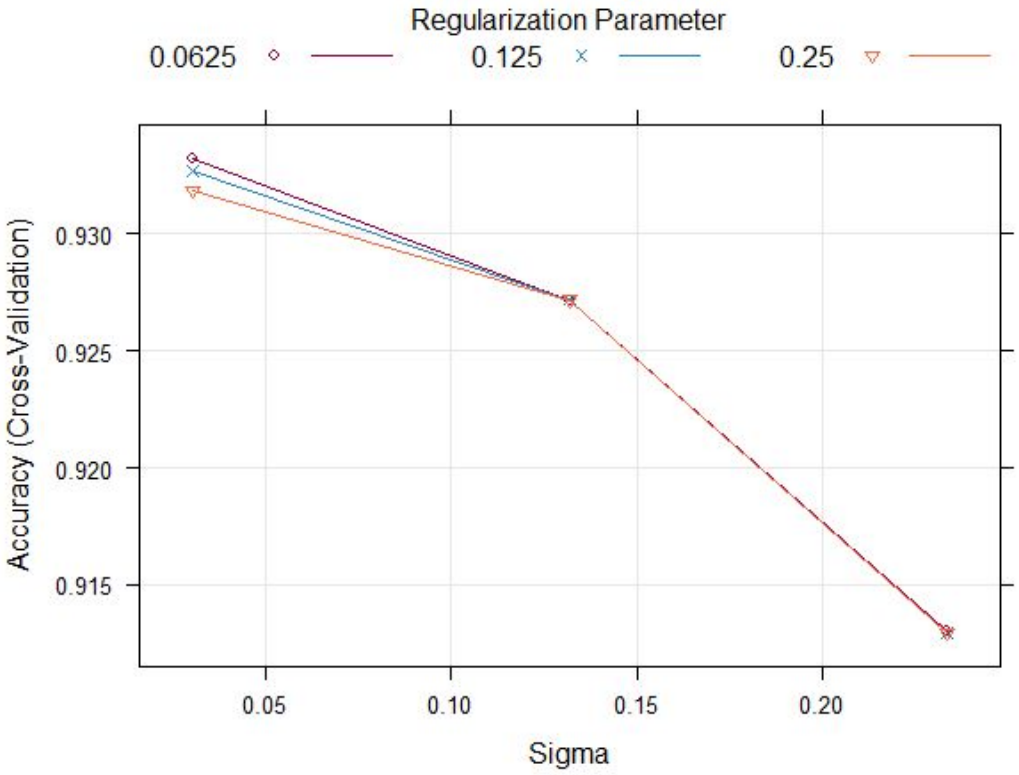

Figure 4.4.2 Tuning Plot of Support Vector Machine Model

Accuracy	0.9307224
Error-rate	0,0692776
Precision	0.9211058
Recall	0.9290497
F1-score	0.9246035
Run Time	5.75493 mins

Figure 4.4.3 Metrics for Support Vector Machine Model

4.5 Logistic Regression Model

Logistic Regression model, prediction algorithms (Prediction Algorithms) are used to predict numerical data in machine learning. Classification is used for estimation of non-numerical, ie categorical data. Logistic Regression is a regression method for classification[7].

"LogitBoost" was used as a method in Logistic Regression modeling. This method contains the parameter "number of boosting iterations (nIter)". Based on this parameter, accuracy values are calculated and the parameter with the best accuracy value is selected.

nlter	Accuracy	Kappa
11	0.9394192	0.9136649
21	0.9420167	0.9180249
31	0.9401334	0.9152465

Figure 4.5.1 Accuracy Rate for nlter Value

The k-fold cross-validation method was applied to the logistic regression model to find the nlter value with the highest accuracy. For each “nlter” value, the optimal accuracy value was found by the cross-validation method. These optimal accuracy values are shown in the table above. For nlter 11, accuracy is 0.9394192 for nlter 21, accuracy is 0.9420167 for nlter 31, is 0.9401334. In order to select the “nlter” value with the highest accuracy from these values, the accuracy and nlter values are visualized with the following plot. As can be seen from this plot, the break occurred at nlter 21 value. Therefore, the nlter 21 value with the highest accuracy was chosen.

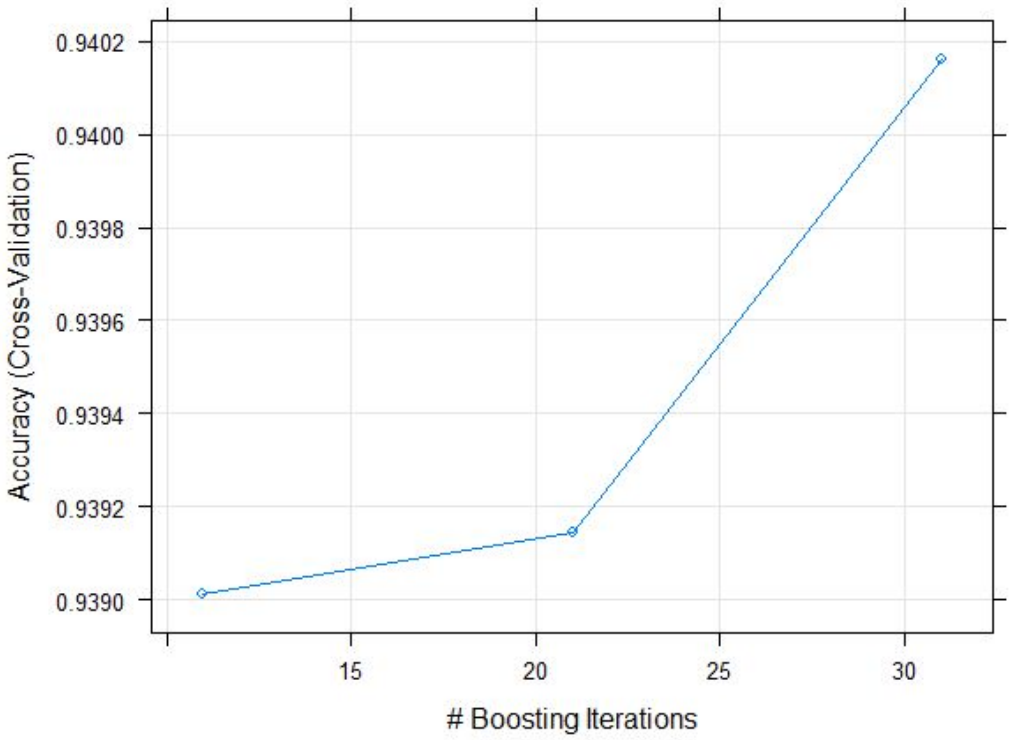

Figure 4.5.2 Tuning Plot of Logistic Regression Model

Accuracy	0.9420167
Error-rate	0,0579833
Precision	0.9347919
Recall	0.9375328
F1-score	0.9360159
Run Time	13.79412 secs

Figure 4.5.3 Metrics for Logistic Regression Model

4.6 Naive Bayes Model

The Naive Bayes classifier is based on Bayes' theorem. It is a lazy learning algorithm, it can also work on unstable data sets. The way the algorithm works calculates the probability of each state for an element and classifies it according to the one with the highest probability value[8].

The "nb" method was used in Naive Bayes modeling. In this method, there are "Laplace correction", "distribution type", "bandwidth adjustment" values as parameters. In the tuning process, the best accuracy value is found by using these parameters and those parameters are selected for modeling.

Use Kernel	Accuracy	Kappa
False	0.9062779	0.8698713
True	0.8995006	0.8576910

Figure 4.6.1 Accuracy Rate for Use Kernel

The table above shows the change in accuracy when using a kernel for Naive Bayes modeling. The use of kernel was evaluated using cross-validation and the highest accuracy value was obtained over five folds. The last values used for the model were fL = 0, usekernel = False and adjust = 1. When kernel = false, the accuracy rate is 0.9062779 and the kappa ratio is 0.8698713. In the plot below, there are accuracy values that vary according to using Gaussian and not using parameters for the Naive Bayes model.

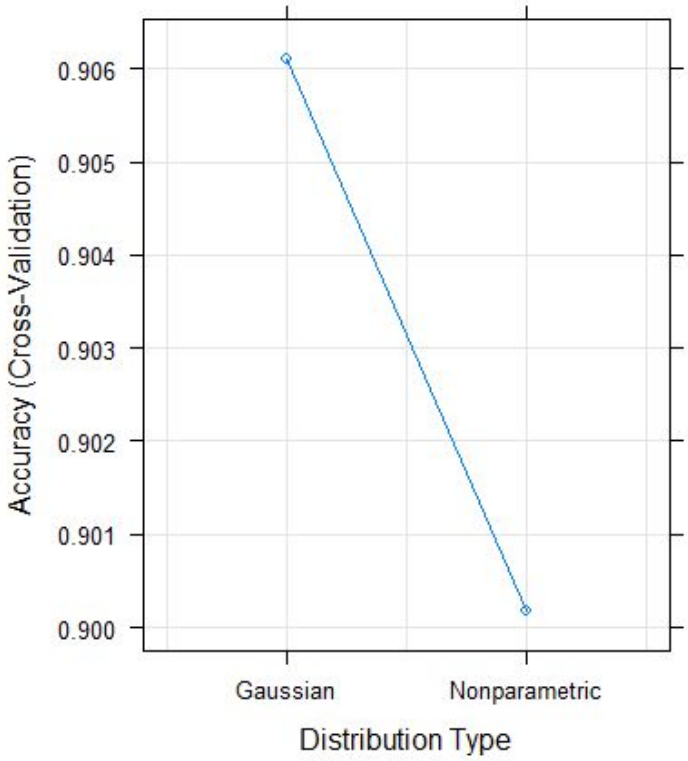

Figure 4.6.2 Tuning Plot of Naive Bayes Model

Accuracy	0.9062779
Error-rate	0.0937221
Precision	0.9255925
Recall	0.8773011
F1-score	0.8975248
Run Time	44.48191 secs

Figure 4.6.3 Metrics for Naive Bayes Model

5. CONCLUSION

	Accuracy	Error-rate	Precision	Recall	F1-score	Run Time
Decision Tree	0.8942781	0.1057219	0.825741	0.8954792	0.8469063	2.281689 secs
Random Forest	0.9554446	0.0445554	0.9489857	0.951534	0.9502304	2.731373 mins
K-Nearest Neighbors	0.9641109	0.0358891	0.9608045	0.961765	0.9612576	2.404891 mins
Support Vector Machine	0.9307224	0,0692776	0.9211058	0.9290497	0.9246035	5.75493 mins
Logistic Regression	0.9420167	0,0579833	0.9347919	0.9375328	0.9360159	13.79412 secs
Naive Bayes	0.9062779	0.0937221	0.9255925	0.8773011	0.8975248	44.48191 secs

Figure 5.1 Metric Values of All Models

Evaluation metrics are available for all models above. Modeling with the highest accuracy rate, precision value, recall value, and F1-score value for the data set used K-Nearest Neighbors. Looking at the runtime values, the fastest working model is the Decision Tree. However, Decision Tree modeling has the lowest accuracy rate. In addition, the slowest modeling is the Support Vector Machine. According to the results in the table, when the runtime evaluation is made for the data set used in the study, the Decision Tree model with the average accuracy rate can be selected, and the K-Nearest Neighbors model with the average runtime can be selected when evaluating for all other metrics.

6. REFERENCES

[1] Clustering, A. (2021). A Simple Explanation of K-Means Clustering and its Advantages. Retrieved 18 January 2021, from https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/#:~:text=How%20Does%20the%20K%2Dmeans%20clustering%20algorithm%20work%3F,groups%20them%20into%20the%20clusters

[2] Hierarchical clustering. (2021). Retrieved 18 January 2021, from https://en.wikipedia.org/wiki/Hierarchical_clustering

[3] Chapter 4: Decision Trees Algorithms. (2021). Retrieved 18 January 2021, from https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1

[4] Makine Öğrenmesi Dersleri 5a: Random Forest (Sınıflandırma). (2021). Retrieved 18 January 2021, from https://medium.com/data-science-tr/makine-%C3%B6%C4%9Frenmesi-dersleri-5-bagging-ve-random-forest-2f803cf21e07

[5] Machine Learning Basics with the K-Nearest Neighbors Algorithm. (2021). Retrieved 18 January 2021, from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[6] Machine Learning —Classification — Support Vector Machine— Kernel Trick— Part 10. (2021). Retrieved 18 January 2021, from https://medium.com/@ekrem.hatipoglu/machine-learning-classification-support-vector-machine-kernel-trick-part-10-7ab928333158

[7] Machine Learning — Classification — Logistic Regression — Part 8. (2021). Retrieved 18 January 2021, from https://medium.com/@ekrem.hatipoglu/machine-learning-classification-logistic-regression-part-8-b77d2a61aae1

[8] Machine Learning - Classification - Naive Bayes - Part 11. (2021). Retrieved 18 January 2021, from https://medium.com/@ekrem.hatipoglu/machine-learning-classification-naive-bayes-part-11-4a10cd3452b4

APPENDIX

CLUSTER CODES

```
library("scales")
library("purrr")
library("tidyverse")  # data manipulation
library("cluster")    # clustering algorithms
library("factoextra")
library("NbClust")
library("ggplot2")
library("dplyr")
library("mclust")
library("fpc")
library("plyr")
library("meltt")
library("class")
library("reshape")
library("reshape2")
library("ISLR")
library("caret")
library("ClusterR")
library("colormap")
library("hrbrthemes")
library("ggpubr")
library("wesanderson")



memory.limit(size = 512000)
music_data <- read.csv("C:/Users/berki/Desktop/data.csv", sep=",", encoding = "UTF-8")
cluster_data <- read.csv("C:/Users/berki/Desktop/final.csv", sep=",", encoding = "UTF-8")

# check for missing values
colSums(is.na(music_data))

# check for duplicate values
sum(duplicated(music_data))

# Change line color and fill color
p1 <- ggplot(music_data, aes(x=valence))+
  geom_histogram(color="darkblue", fill="lightblue")+
  theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
```

```
# Change line color and fill color
p2 <- ggplot(music_data, aes(x=year))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p3 <- ggplot(music_data, aes(x=acousticness))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p4 <- ggplot(music_data, aes(x=danceability))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p5 <- ggplot(music_data, aes(x=energy))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p6 <- ggplot(music_data, aes(x=instrumentalness))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p7 <- ggplot(music_data, aes(x=liveness))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p8 <- ggplot(music_data, aes(x=tempo))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
# Change line color and fill color
p9 <- ggplot(music_data, aes(x=speechiness))+
 geom_histogram(color="darkblue", fill="lightblue")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))


ggarrange(p1, p2, p3, p4, p5, p6, p7, p8, p9 ,
          ncol = 3, nrow = 3)
```

```
list_kmeans = list()
list_hclust = list()
max_kmeans=0
max_hclust=0
sample_K=data.frame()
sample_H=data.frame()
list_subset = list()
############################### find the best clustered sample
for (i in 1:100){
  sub<-sample_n(music_data[,c("acousticness" , "energy" ,
                        "danceability" , "valence",
                        "instrumentalness", "speechiness" ,
                        "tempo" , "loudness")], 18000)
  list_subset[[i]]=sub
}

print("Kmeans")
for (i in 1:100){
  k2 <- eclust(list_subset[[i]], "kmeans", k = 6,
               nstart = 25,graph = FALSE)
  sil <- silhouette(k2$cluster, dist(list_subset[[i]]))
  silinfo <- k2$silinfo
  dd <- dist(list_subset[[i]] ,method ="euclidean")
  # Statistics for k-means clustering
  km_stats <- cluster.stats(dd,  k2$cluster)
  list_kmeans[i] <- silinfo$avg.width
  if(silinfo$avg.width>max_kmeans){
    max_kmeans=silinfo$avg.width
    sample_K=list_subset[[i]]
  }
  cat("Iteration",i,'average:',silinfo$avg.width,"\n")
  cat('Size:',k2$size,"\n")
}
print(max_kmeans)
print("Hclust")
for (i in 1:100){
  k2 <- eclust(list_subset[[i]], "hclust", k = 6,
               method = "complete", graph = FALSE)
  sil <- silhouette(k2$cluster, dist(list_subset[[i]]))
  silinfo <- k2$silinfo
  dd <- dist(list_subset[[i]], method ="euclidean")
  # Statistics for k-means clustering
  km_stats <- cluster.stats(dd, k2$cluster)
```

```
 list_hclust[i] <- silinfo$avg.width
 if(silinfo$avg.width>max_hclust){
  max_hclust=silinfo$avg.width
  sample_H=list_subset[[i]]
 }
 cat("Iteration",i,'average:',silinfo$avg.width,"\n")
 cat('Size:',k2$size,"\n")
}
print( max_hclust)
##############################
df <- list_hclust %>% tibble::enframe() %>% tidyr::unnest()
colnames(df) <- c("samples", "scores")
df <- list_kmeans %>% tibble::enframe() %>% tidyr::unnest()
colnames(df) <- c("samples", "scores")
# Plot
ggplot(df, aes(x=samples, y=scores)) +ggtitle("k-means") +
 geom_line(lwd = 0.8) +
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(), axis.line = element_line(colour = "white"))

# corelation matrix
cormat <- round(cor(cluster_data[,1:8]),2)
melted_cormat <- melt(cormat)
# Get lower triangle of the correlation matrix
get_lower_tri<-function(cormat){
 cormat[upper.tri(cormat)] <- NA
 return(cormat)
}
# Get upper triangle of the correlation matrix
get_upper_tri <- function(cormat){
 cormat[lower.tri(cormat)]<- NA
 return(cormat)
}
upper_tri <- get_upper_tri(cormat)
upper_tri

# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Heatmap
ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                      midpoint = 0, limit = c(-1,1), space = "Lab",
                      name="Pearson\nCorrelation") +
```

```
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
reorder_cormat <- function(cormat){
  # Use correlation between variables as distance
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
}
# Reorder the correlation matrix
cormat <- reorder_cormat(cormat)
upper_tri <- get_upper_tri(cormat)
# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
# Print the heatmap
print(ggheatmap)
ggheatmap +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.6, 0.7),
    legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))
#############################
# we should consider dropping loudness column with the value 0.78 (between 0.7 - 0.9)
# kmeans++ #################
```

```
k2 <- eclust(cluster_data, "hclust", k = 6,
          method = "complete", graph = FALSE)

k2$size
# sizes are not really equal this might be problematic
############################
# Change line color and fill color
p1 <- ggplot(cluster_data, aes(x=mood))+
 geom_histogram(color="darkblue", fill="lightblue",stat="count")+
 theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
                panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
### Cluster 1
cat("############################")
df <- cluster_data[k2$cluster==1,]
cat("acousticness",mean(df$acousticness))
cat("energy",mean(df$energy))
cat("danceability",mean(df$danceability))
cat("valence",mean(df$valence))
cat("instrumentalness",mean(df$instrumentalness))
cat("speechiness",mean(df$speechiness))
cat("tempo",mean(df$tempo))
cat("loudness",mean(df$loudness))
c1 <- c(mean(df$acousticness),mean(df$energy),mean(df$danceability),mean(df$valence),
      mean(df$instrumentalness),mean(df$speechiness),mean(df$tempo),mean(df$loudness))
cat("############################")
### Cluster 2
cat("############################")
df <- cluster_data[k2$cluster==2,]
cat("acousticness",mean(df$acousticness))
cat("energy",mean(df$energy))
cat("danceability",mean(df$danceability))
cat("valence",mean(df$valence))
cat("instrumentalness",mean(df$instrumentalness))
cat("speechiness",mean(df$speechiness))
cat("tempo",mean(df$tempo))
cat("loudness",mean(df$loudness))
c2 <- c(mean(df$acousticness),mean(df$energy),mean(df$danceability),mean(df$valence),
      mean(df$instrumentalness),mean(df$speechiness),mean(df$tempo),mean(df$loudness))
cat("############################")
### Cluster 3
cat("############################")
df <- cluster_data[k2$cluster==3,]
cat("acousticness",mean(df$acousticness))
cat("energy",mean(df$energy))
```

```
cat("danceability",mean(df$danceability))
cat("valence",mean(df$valence))
cat("instrumentalness",mean(df$instrumentalness))
cat("speechiness",mean(df$speechiness))
cat("tempo",mean(df$tempo))
cat("loudness",mean(df$loudness))
c3 <- c(mean(df$acousticness),mean(df$energy),mean(df$danceability),mean(df$valence),
      mean(df$instrumentalness),mean(df$speechiness),mean(df$tempo),mean(df$loudness))
cat("############################")
### Cluster 4
cat("############################")
df <- cluster_data[k2$cluster==4,]
cat("acousticness",mean(df$acousticness))
cat("energy",mean(df$energy))
cat("danceability",mean(df$danceability))
cat("valence",mean(df$valence))
cat("instrumentalness",mean(df$instrumentalness))
cat("speechiness",mean(df$speechiness))
cat("tempo",mean(df$tempo))
cat("loudness",mean(df$loudness))
c4 <- c(mean(df$acousticness),mean(df$energy),mean(df$danceability),mean(df$valence),
      mean(df$instrumentalness),mean(df$speechiness),mean(df$tempo),mean(df$loudness))
cat("############################")
### Cluster 5
cat("############################")
df <- cluster_data[k2$cluster==5,]
cat("acousticness",mean(df$acousticness))
cat("energy",mean(df$energy))
cat("danceability",mean(df$danceability))
cat("valence",mean(df$valence))
cat("instrumentalness",mean(df$instrumentalness))
cat("speechiness",mean(df$speechiness))
cat("tempo",mean(df$tempo))
cat("loudness",mean(df$loudness))
c5 <- c(mean(df$acousticness),mean(df$energy),mean(df$danceability),mean(df$valence),
      mean(df$instrumentalness),mean(df$speechiness),mean(df$tempo),mean(df$loudness))
cat("############################")
### Cluster 6
cat("############################")
df <- cluster_data[k2$cluster==6,]
cat("acousticness",mean(df$acousticness))
cat("energy",mean(df$energy))
cat("danceability",mean(df$danceability))
cat("valence",mean(df$valence))
```

```
cat("instrumentalness",mean(df$instrumentalness))
cat("speechiness",mean(df$speechiness))
cat("tempo",mean(df$tempo))
cat("loudness",mean(df$loudness))
c6 <- c(mean(df$acousticness),mean(df$energy),mean(df$danceability),mean(df$valence),
      mean(df$instrumentalness),mean(df$speechiness),mean(df$tempo),mean(df$loudness))
cat("############################")

cluster_data[k2$cluster==1,"mood"]<-"happy"
cluster_data[k2$cluster==2,"mood"]<-"gleeful"
cluster_data[k2$cluster==3,"mood"]<-"sad"
cluster_data[k2$cluster==4,"mood"]<-"calm"
cluster_data[k2$cluster==5,"mood"]<-"romantic"
cluster_data[k2$cluster==6,"mood"]<-"sensual"

cluster_data$mood <- as.factor(cluster_data$mood)
################################
# Plotting the clusters
# Create data
set.seed(1)
data <- t(data.frame(c1, c2, c3, c4, c5, c6))


colnames(data) <- c("acousticness" , "energy" ,
                    "danceability" , "valence",
                    "instrumentalness", "speechiness" ,
                    "tempo" , "loudness")


rownames(data) <- c("Happy", "Gleeful", "Sad", "Calm",
                    "Romantic","Sensual")
data <- as.data.frame(data)
# Barplot
data <- data %>% slice(c(1:8)) %>%
 t() %>%
 as.data.frame() %>%
 add_rownames() %>%
 mutate(rowname=factor(rowname, rowname)) %>%
 gather(key=name, value=mark, -1)

#Recode
data$name <- recode(data$name, V1 = "Cluster1", V2 = "Cluster2",
                    V3 = "Cluster3", V4 = "Cluster4", V5 = "Cluster5",
                    V6 = "Cluster6")
```

```
# Plot
data %>% ggplot( aes(x=rowname, y=mark)) +
  geom_bar(stat="identity", fill="#69b3a2", width=0.6) +
  coord_flip() +
  theme_ipsum() +
  theme(
    panel.grid.minor.y = element_blank(),
    panel.grid.major.y = element_blank(),
    axis.text = element_text( size=48 )
  ) +
  ylim(0,1) +
  ylab("range") +
  xlab("") +
  facet_wrap(~name, ncol=6)
################################

# create final dataset with 1000 1000 1000 1000 484 643 distribution
sub_happy<-sample_n(cluster_data[cluster_data$mood=="happy",], 2000)
sub_gleeful<-sample_n(cluster_data[cluster_data$mood=="gleeful",], 2000)
sub_sad<-sample_n(cluster_data[cluster_data$mood=="sad",], 2000)
sub_calm<-sample_n(cluster_data[cluster_data$mood=="calm",], 2000)
romantic <- cluster_data[cluster_data$mood=="romantic",]
sensual <- cluster_data[cluster_data$mood=="sensual",]

final <- rbind(sub_happy, sub_gleeful,sub_sad,sub_calm,romantic,sensual)


write.csv(cluster_data,"C:/Users/berki/Desktop/final.csv", row.names = FALSE)
library("caret")

control <- trainControl(method='repeatedcv',
                        number=10,
                        repeats=3)
x <- cluster_data[,1:8]
mtry <- sqrt(ncol(x))
tunegrid <- expand.grid(.mtry=mtry)
rf_default <- train(mood~.,
                    data=cluster_data,
                    method='rf',
                    metric='Accuracy',
                    tuneGrid=tunegrid,
                    trControl=control)
print(rf_default)
```

CLASSIFICATION CODES

```
trellis.par.set(caretTheme())


#*****COMPUTE METRİCS*****

compute_metrics <- function(cm) {
  precision = diag(cm$table) / colSums(cm$table)
  recall = diag(cm$table) / rowSums(cm$table)
  f1 = 2 * precision * recall / (precision + recall)
  data.frame(precision, recall, f1)
  macroPrecision = mean(precision)
  macroRecall = mean(recall)
  macroF1 = mean(f1)
  data.frame(macroPrecision, macroRecall, macroF1)
}

#*******NAIVE BAYES**********

start_time <- Sys.time()
trControl <- trainControl(method="cv", number=5)
naiveBayes_model <- train(mood~., data=final_data, method = "nb",trControl=trControl)
end_time <- Sys.time()
end_time - start_time
print(naiveBayes_model)
cm<-confusionMatrix(naiveBayes_model)
compute_metrics(cm)
plot(naiveBayes_model)

#*******RANDOM FOREST*********
start_time <- Sys.time()
randomForest_model <- train(mood ~ .,data = final_data,  method="rf",trControl=trControl)
end_time <- Sys.time()
end_time - start_time
print(randomForest_model)
cm<-confusionMatrix(randomForest_model)
compute_metrics(cm)
plot(randomForest_model)
#******DECISION TREE******
start_time <- Sys.time()
trControl <- trainControl(method  = "cv",number  = 5)
decisionTree_model = train(mood ~ ., data=final_data, method="rpart2", trControl = trControl)
end_time <- Sys.time()
end_time - start_time
print(decisionTree_model)
```

```
cm<-confusionMatrix(decisionTree_model)
compute_metrics(cm)
plot(decisionTree_model)

#*****KNN ********
start_time <- Sys.time()
trControl <- trainControl(method  = "cv",number  = 5)
set.seed(3333)
knn_model <- train(mood ~ .,data= final_data,method = "kknn",trControl  = trControl)
end_time <- Sys.time()
end_time - start_time
print(knn_model)
cm<-confusionMatrix(knn_model)
compute_metrics(cm)
plot(knn_model)

#****** SVM ********
start_time <- Sys.time()
trControl <- trainControl(method="cv", number=5)
svm_model <-train (mood~., data=final_data, method= "lssvmRadial",trControl  = trControl)
end_time <- Sys.time()
end_time - start_time
print(svm_model)
cm<-confusionMatrix(svm_model)
compute_metrics(cm)
plot(svm_model)
summary(svm_model)

#*******LOGISTIC REGRESSION******
start_time <- Sys.time()
trControl <- trainControl(method="cv", number=5)
logisticReg_model <-train (mood~., data=final_data, method= "LogitBoost",trControl  = trControl)
end_time <- Sys.time()
end_time - start_time
print(logisticReg_model)
cm<-confusionMatrix(logisticReg_model)
compute_metrics(cm)
plot(logisticReg_model)
```