Genre Classification on Music Dataset Using Data Mining Techniques

A Project Report

Submitted in fulfilment

In

Data Mining – ITE2006

By-

Registration Number	Name
16BIT0184	Astha Baranwal
16BIT0204	Bhagyashree Bagwe

Under the Guidance of

PROF: Lakshmi Priya G G

October, 2018

Table of Contents

1. Abstract

Music genre classification means segregating or classifying songs in to types like, Pop, Country, Rock, Jazz, R&B, Classical, etc. Genre classification is a very subjective topic. Based on the approach, songs might get classified differently by different people. It is fairly simple for a human being to identify the genre of a song. One thinks about how fast the beat of the song is, the mood of the song and the ambiance it creates. All these help create a mental picture of the song and thus the genres associated with it are determined.

That makes music genre classification a very controversial topic. Hence, creating an automated system based on a scientific approach is beneficial to the ever-growing music industry.

Genre classification is crucial to the music industry as it plays a vital role in song recommendation systems. Song recommendation systems work in a way that help listeners find similar music.

2. Problem Statement

Music genre classification is a trending topic in the music industry. The objective of this project is to create an automated system to classify songs into specific mainstream genres based on their features. Classification technique (K Nearest Neighbor and Random Forest) will be used to classify songs into genres.

3. Motivation

Music is one thing that connects people from all around the world. Irrespective of language, it soothes the souls of people. Music being likable or not likable is subjective and varies from person to person.

There are hundreds of music genres available all over the world. Some genres include, pop, rock, jazz, bass, country, etc. Different genres are likable to different people.

We have always been music enthusiasts and have wanted to do know why people like the music they like. Is it the beats? Is it notes? Or is it the keys? Does the genre of a song depend on these factors?

The motivation behind this project is to find songs that belong to a particular genre based on attributes like, beats, keys, bars, tatums, loudness etc. The goal here is to build a system that determines the genre of the song based on these attributes.

Another task at hand includes, checking for the accuracy of the predicted values of the built system against the actual genres of the songs.

The dataset used in this project is a subset of the million song dataset. The million song dataset contains a million songs from all over the world. These songs come from various countries of various continents. A smaller version of this dataset is used in the project due to computational limitations.

The reason behind working on a music genre classification system is that there isn't much research available in this area. The existing systems are either limited to certain genres or limited features.

Thus, this report focuses on building a system that will include maximum possible genres and attributes.

4. Introduction

Music Genre classification means classifying songs into a specific genre. A genre can be termed as a method to group similar songs or simply a tag name.

In the proposed model, we classify songs based on attributes like, beats, keys, bars, tatums, loudness etc. There are a total of 13 Genres in the selected dataset.

Classification technique is used to classify genres. The algorithms used in the proposed system are KNN (k Nearest Neighbor) and Random Forest.

The KNN uses the concept of analogy. The basic idea is that the instance is that what it resembles. So if a particular instance is to be classified, at its K-closest (most similar) neighbors are considered. The point is classified as the majority class in those neighbors.

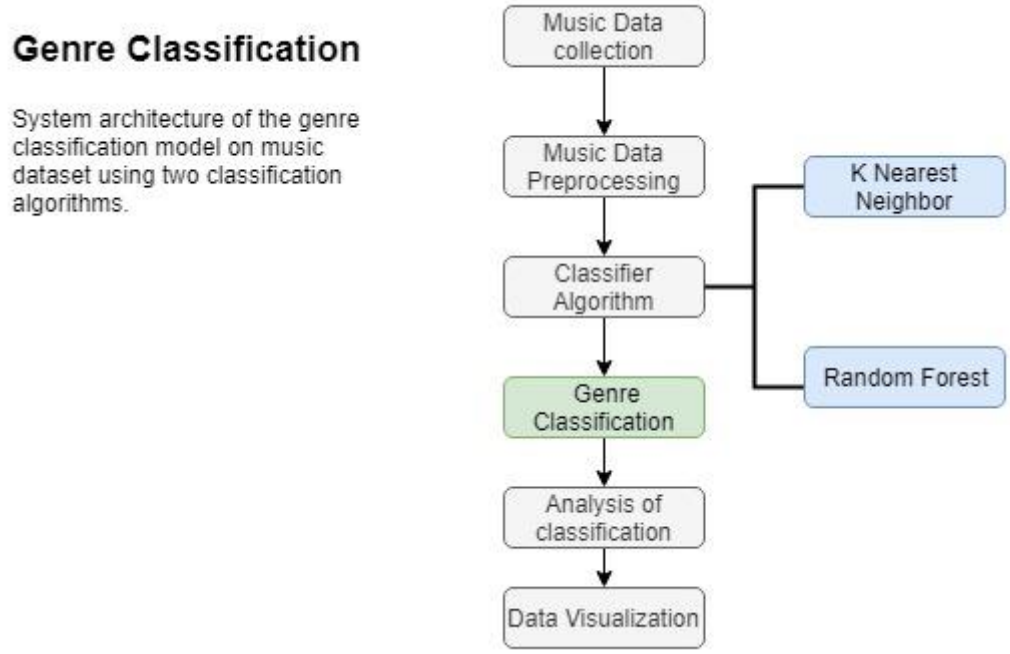
The KNN algorithm depends on two things, metric used to compute the distance between two points and the value of parameter 'k' which is the number of neighbors to consider. When 'k' is a very small number KNN may overfit i.e., it may classify the instance just based on the closest neighbors instead of learning a good separating frontier between classes. But if 'k' is a very big number KNN may underfit, i.e. the model may predict that every instance belongs to the class that has more samples.

As the name suggests, random forest is a forest of random decision tress. The algorithm randomly selects k features from total m features. Among these k features, the node d is calculated which is the best split point. The best split concept is used to split the node into daughter nodes. These steps are repeated a n number of times to get n trees.

Basically, Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. The random algorithm used in wide varieties applications namely, banking, medicine, e – commerce, stock market etc.

5. Model description

a. Architecture



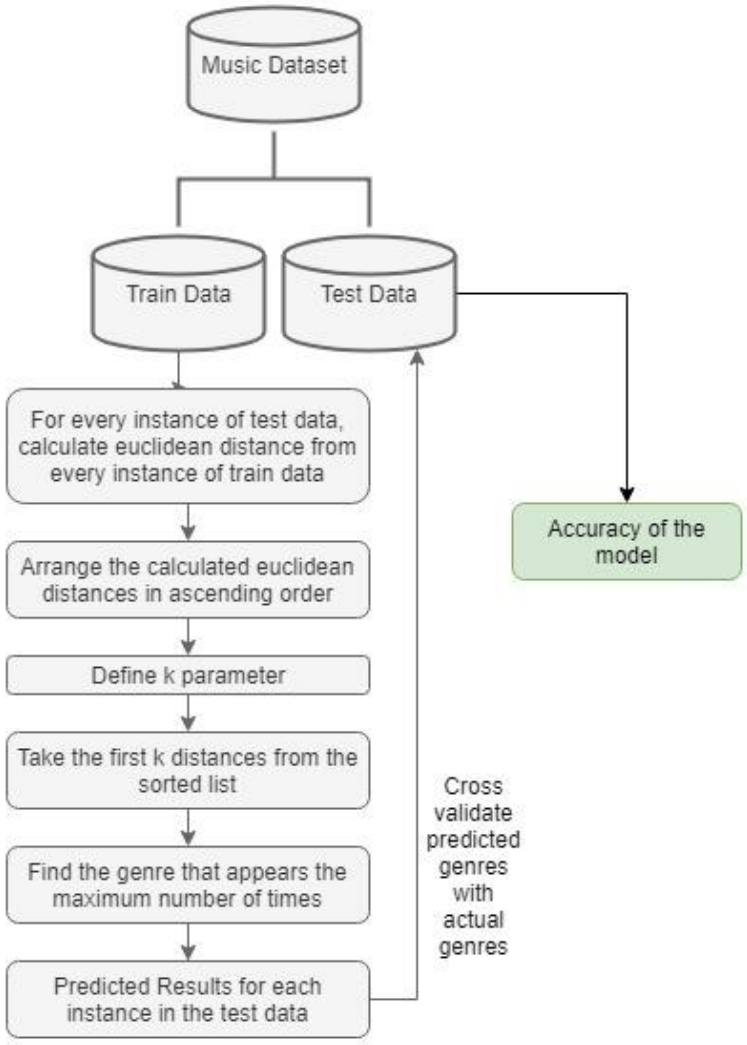
The above flowchart explains the basic general architecture of the system proposed in the system.

The steps include:

i. Music Data Collection
ii. Music Data Preprocessing
iii. Classifier Algorithm
iv. Genre Classification
    a. K Nearest Neighbor
    b. Random Forest
v. Analysis of classification
vi. Data Visualization

K Nearest Neighbor

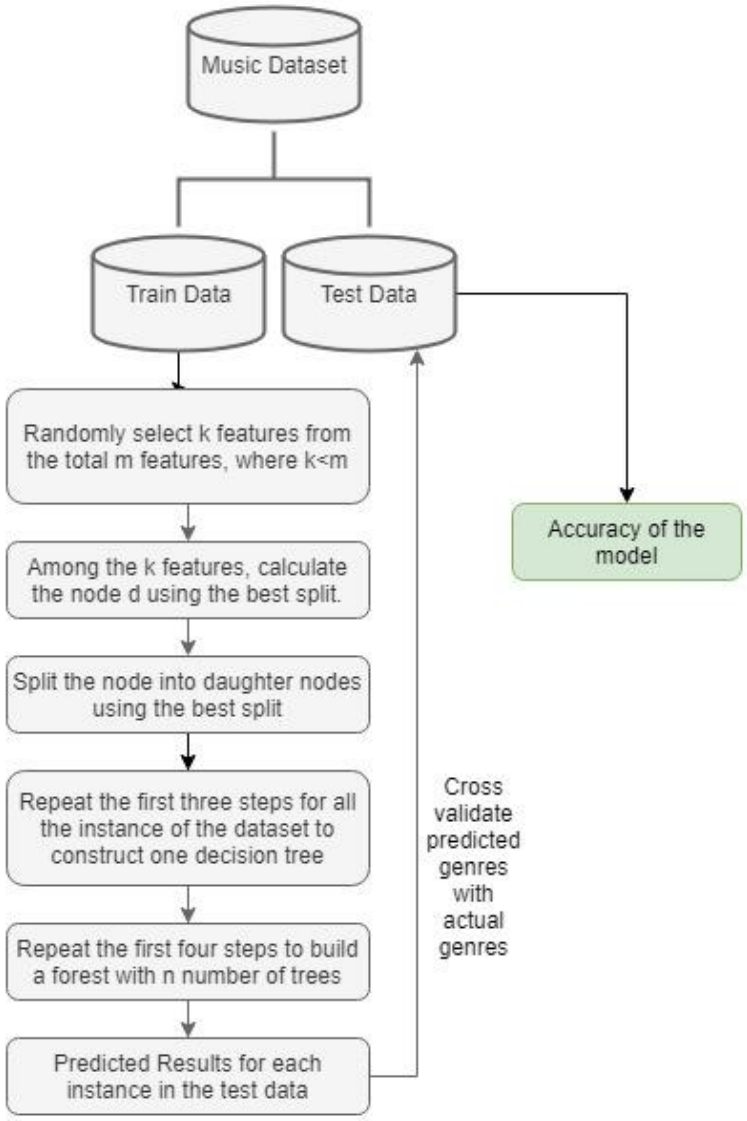
Architecture of KNN classifier on music dataset for genre classification.



i. The above flowchart explains the architecture of the KNN classifier model
ii. The steps include:
iii. Divide the dataset in to train and test data
iv. For every instance of test data, calculate Euclidean distance from every instance of train data
v. Arrange the calculated Euclidean distances in ascending order
vi. Define k parameter
vii. Take the first k distances from the sorted list
viii. Find the genre that appears the maximum number of times
ix. Predicted Results for each instance in the test data
x. Cross validate predicted genres with actual genres
xi. Obtain the accuracy of the model

Random Forest

Architecture of Random Forest classifier on music dataset for genre classification.



The above flowchart explains the architecture of the Random classifier model

The steps include:

xii. Divide the dataset in to train and test data

xiii. Randomly select k features from the total m features, where k<m

xiv. Among the k features, calculate the node d using the best split

xv. Split the node into daughter nodes using the best split

xvi. Repeat the steps 2 to 4 for all the instance of the dataset to construct one decision tree

xvii. Repeat the steps 2 to 5 to build a forest with n number of trees

xviii. Obtain the Predicted Results for each instance in the test data

xix. Cross validate predicted genres with actual genres

xx. Obtain the accuracy of the model

b. Parameter settings

K Nearest Neighbor

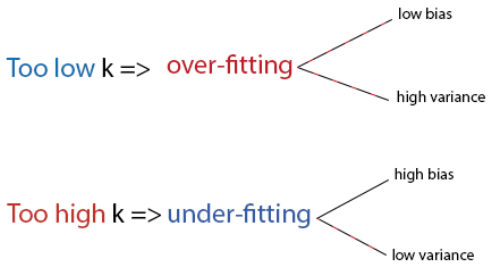
The Parameters in KNN are:

i. The Distance Measure

- The distance measure can be calculated in terms of Euclidean distance, Chebychev distance, Mahalanobis distance, Hamming distance and Cosine similarity.
- In the proposed system, we select Euclidean distance as the distance measure.

ii. The value of K

- **Optimal value:** Generally speaking, we get and optimal value of k at the square root of the number of instances.
- K = sqrt (No of instances)
- **Over fitting:** Over fitting occurs if the k value is too small. It may classify the instance just based on the closest neighbors instead of learning a good separating frontier between classes.
- **Under fitting:** Under fitting occurs if the k value is too large. The model may predict that every instance belongs to the class that has more sample.



Bias-Variance tradeoff for k-Nearest Neighbor

Bias are the simplifying assumptions made by a model to make the target function easier to learn. Variance is the amount that the estimate of the target function will change if different training data was used.

Random Forest

i. Number of tress (n_trees)

- ✓ This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower. You should choose as high value as your processor can handle because this makes your predictions stronger and more stable.

ii. Number of features (n_features)

- ✓ These are the maximum number of features Random Forest is allowed to try in an individual tree. There are multiple options available in Python to assign maximum features. Here are a few of them :
  - i. Auto/None : This will simply take all the features which make sense in every tree. Here we simply do not put any restrictions on the individual tree.
  - ii. sqrt : This option will take square root of the total number of features in individual run. For instance, if the total number of variables are 100, we can only take 10 of them. Generally speaking, this is the optimal.
  - iii. 0.2 : This option allows the random forest to take 20% of variables in individual run. We can assign and value in a format "0.x" where we want x% of features to be considered.

iii. Number of folds (k_folds)

- ✓ This indicates the number of folds in the dataset.
- ✓ The dataset is divided in to k folds and the accuracy is calculated for each fold.
- ✓ The average accuracy of all the folds is taken in to consideration to take the final accuracy.

6. Dataset description

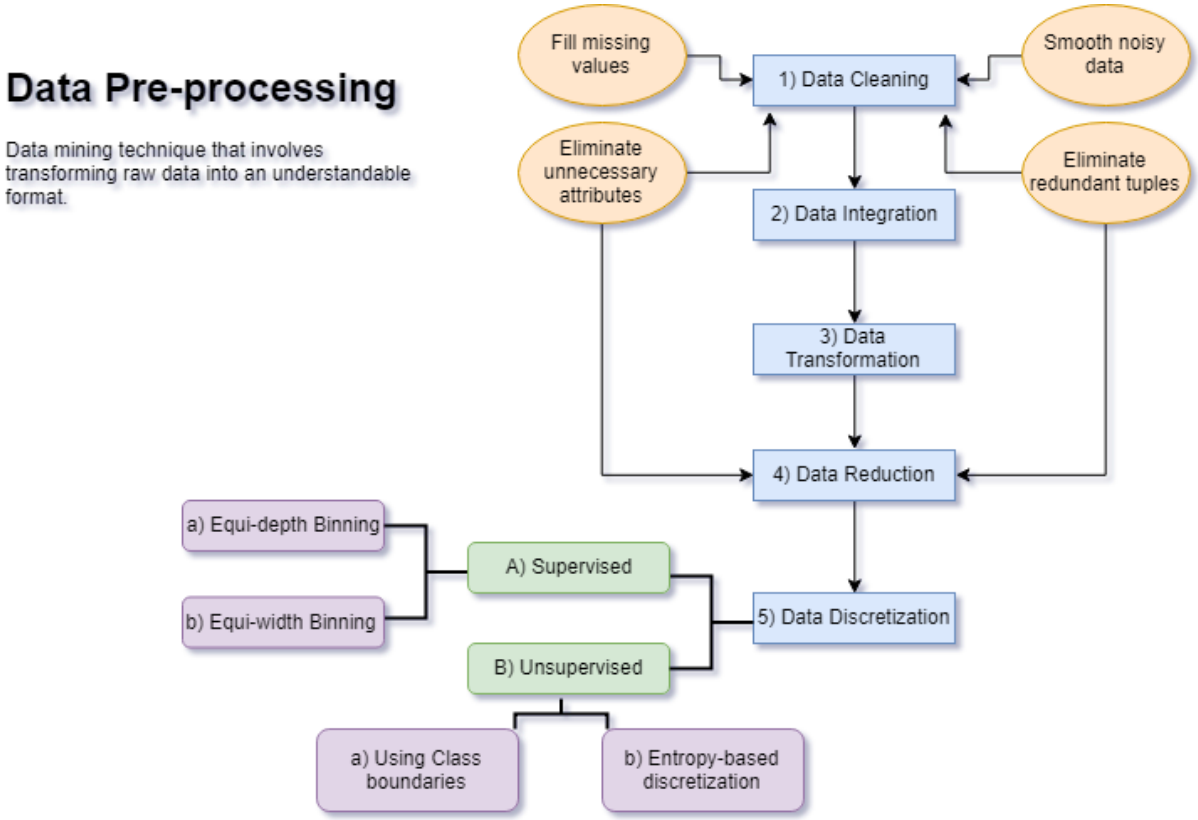
- ✓ Subset (10k instances – 3.2 MB) of million song dataset from the CORGIS dataset project.
- ✓ It has 10 k instances and 32 attributes.
- ✓ Link: https://think.cs.vt.edu/corgis/csv/music/music.html
- ✓ It contains instances from the Million Song Dataset, which is a collaboration between the EchoNest and LabROSA about one million popular contemporary songs.

Sr. No.	Attribute	Description	Datatype
1	Song_id	Unique ID of the song	string
2	Song_name	Name of the song	string
3	Album_id	Album ID of the song	int
4	Album_name	Name of the Album the song was released in	string
5	Artist_id	Unique ID of the artist	string
6	Artist_name	Name of the artist	string
7	Song_popularity	Popularity of song	float
8	Artist_popularity	This corresponds to how much buzz the artist is getting right now. This is derived from many sources, including mentions on the web, mentions in music blogs, music reviews, play counts, etc.	float
9	Familiarity	This corresponds to how well known the artist is. You can look at familiarity as the likelihood that any person selected at random will have heard of the artist.	float
10	Duration	Duration of the song in seconds	float
11	Bars_confidence	Confidence value (between 0 and 1) associated with each bar by The Echo Nest	array float
12	Bars_start	Start time of each bar according to The Echo Nest. It denotes the number of bars in the song	array float

13	Beats_confidence	Confidence value (between 0 and 1) associated with each beat by The Echo Nest	array float
14	Beats_start	Start time of each beat according to The Echo Nest. It denotes the number of beats in the song	array float
15	End_of_fade_in	Time of the end of the fade in, in seconds , at the beginning of the song	float
16	Key	Estimation of the key the song is in	int
17	Key_confidence	Confidence of the key estimation	float
18	Location	Location of the artist	string
19	Latitude	Location latitude of artist	float
20	Longitude	Location longitude of artist	float
21	Loudness	General loudness of the track in dB	float
22	Mode	Major or minor mode	int
23	Mode_confidence	Confidence measure of mode	float
24	Start_of_fade_out	Start time of the fade out, in seconds, at the end of the song	float
25	Tatums_confidence	Confidence value (between 0 and 1) associated with each tatum	array float
26	Tatums_start	start time of each tatum (smallest rhythmic element)	array float
27	Tempo	Estimated tempo in BPM (Beats Per Minute)	float
28	Terms_freq	Frequency of tags of artists in the Echo Nest API	array float
29	Time_signature	Estimate of number of beats per bar	int
30	Time_signature_confidence	Confidence of the time signature estimation	float
31	Year	Year of release	int
32	Genre	The genre of the song	string

7. Implementation

i. Data Preprocessing



a) Data Cleaning -

a. Selection of attributes: Selecting the most suitable and necessary attributes based on semantics. Eliminating the unnecessary or redundant attributes.
b. Arranging attributes according to importance or general protocol.
c. Fill in missing values wherever possible – Using concepts of mean, median, mode, etc. or by researching about the song.
    i. song_name: One value missing – Deleted the tuple | Noisy data eliminated using excel
    ii. Loudness: Cannot be positive. So tuples with positive loudness eliminated.
    iii. For the attribues with 0 values, we first detect the 0 values and turn them into null values. Next, we replace them with the mean values.

d. Detect and remove outliers.
e. Eliminate tuples with too many missing values which cannot be filled with a central tendency value.

b) Data Integration - NA

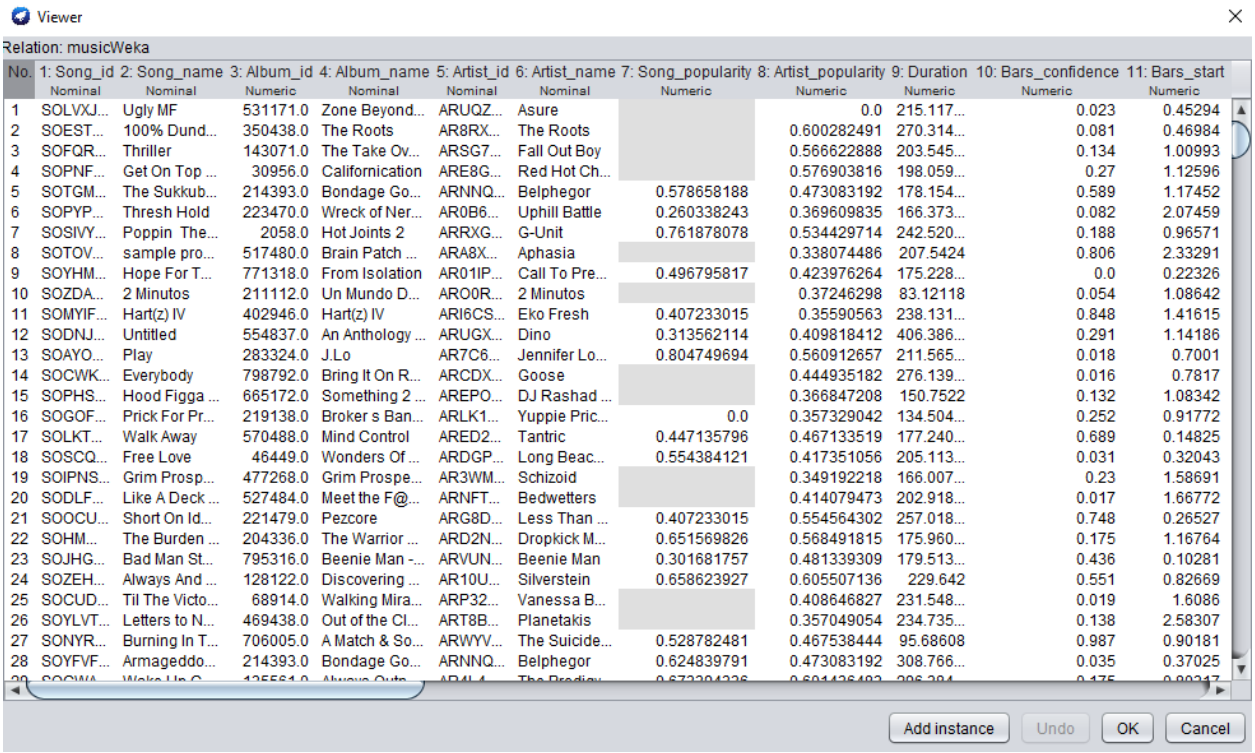
c) Data Transformation - NA

d) Data Reduction - Delete unnecessary or redundant tuples. Also, delete tuples with too many missing values.

e) Data discretization -

a. KNN & RF: Doesn't require discrete values. So, no need to discretized dataset to apply KNN or RF.

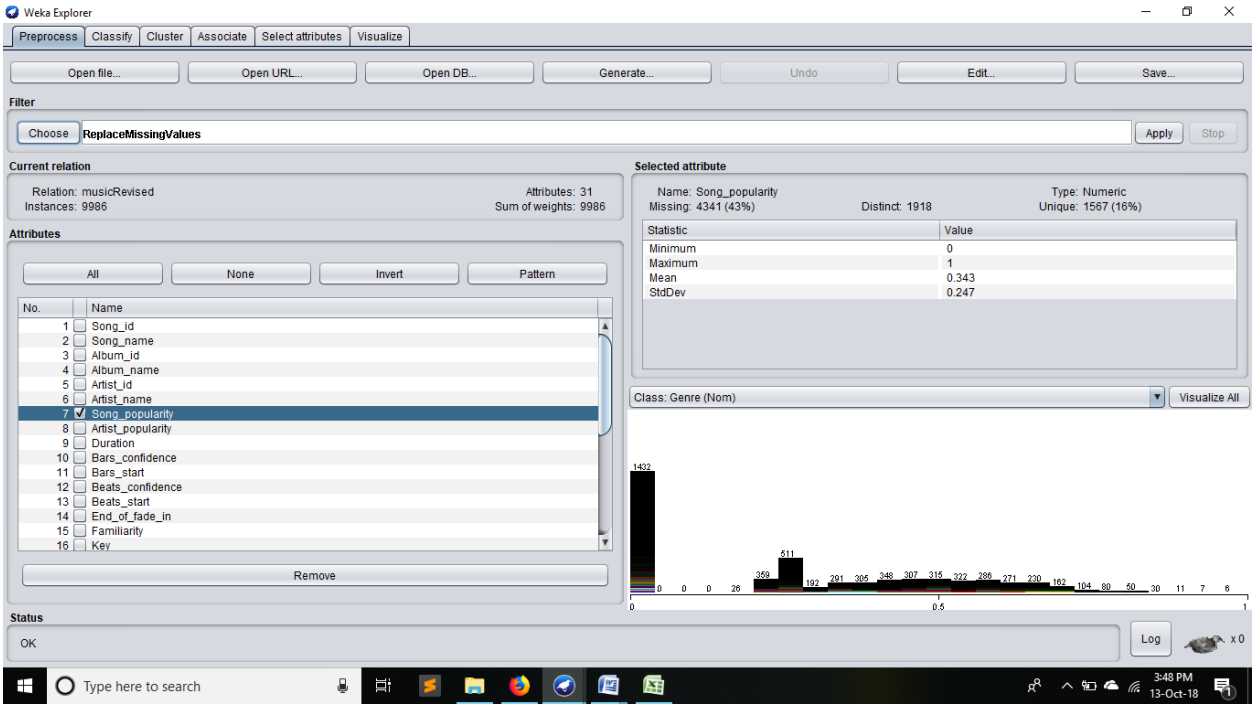
Dataset Before Cleaning



No.	1: Song_id	2: Song_name	3: Album_id	4: Album_name	5: Artist_id	6: Artist_name	7: Song_popularity	8: Artist_popularity	9: Duration	10: Bars_confidence	11: Bars_start
	Nominal	Nominal	Numeric	Nominal	Nominal	Nominal	Numeric	Numeric	Numeric	Numeric	Numeric
1	SOLVXJ...	Ugly MF	531171.0	Zone Beyond...	ARUQZ...	Asure		0.0	215.117...	0.023	0.45294
2	SOEST...	100% Dund...	350438.0	The Roots	AR8RX...	The Roots		0.600282491	270.314...	0.081	0.46984
3	SOFQR...	Thriller	143071.0	The Take Ov...	ARSG7...	Fall Out Boy		0.566622888	203.545...	0.134	1.00993
4	SOPNF...	Get On Top ...	30956.0	Californication	ARE8G...	Red Hot Ch...		0.576903816	198.059...	0.27	1.12596
5	SOTGM...	The Sukkub...	214393.0	Bondage Go...	ARNNQ...	Belphegor	0.578658188	0.473083192	178.154...	0.589	1.17452
6	SOPYP...	Thresh Hold	223470.0	Wreck of Ner...	AR0B6...	Uphill Battle	0.260338243	0.369609835	166.373...	0.082	2.07459
7	SOSIVY...	Poppin The...	2058.0	Hot Joints 2	ARRXG...	G-Unit	0.761878078	0.534429714	242.520...	0.188	0.96571
8	SOTOV...	sample pro...	517480.0	Brain Patch ...	ARA8X...	Aphasia		0.338074486	207.5424	0.806	2.33291
9	SOYHM...	Hope For T...	771318.0	From Isolation	AR01IP...	Call To Pre...	0.496795817	0.423976264	175.228...	0.0	0.22326
10	SOZDA...	2 Minutos	211112.0	Un Mundo D...	ARO0R...	2 Minutos		0.37246298	83.12118	0.054	1.08642
11	SOMYIF...	Hart(z) IV	402946.0	Hart(z) IV	ARI6CS...	Eko Fresh	0.407233015	0.35590563	238.131...	0.848	1.41615
12	SODNJ...	Untitled	554837.0	An Anthology ...	ARUGX...	Dino	0.313562114	0.409818412	406.386...	0.291	1.14186
13	SOAYO...	Play	283324.0	J.Lo	AR7C6...	Jennifer Lo...	0.804749694	0.560912657	211.565...	0.018	0.7001
14	SOCWK...	Everybody	798792.0	Bring It On R...	ARCDX...	Goose		0.444935182	276.139...	0.016	0.7817
15	SOPHS...	Hood Figga ...	665172.0	Something 2 ...	AREPO...	DJ Rashad ...		0.366847208	150.7522	0.132	1.08342
16	SOGOF...	Prick For Pr...	219138.0	Broker s Ban...	ARLK1...	Yuppie Pric...	0.0	0.357329042	134.504...	0.252	0.91772
17	SOLKT...	Walk Away	570488.0	Mind Control	ARED2...	Tantric	0.447135796	0.467133519	177.240...	0.689	0.14825
18	SOSCQ...	Free Love	46449.0	Wonders Of ...	ARDGP...	Long Beac...	0.554384121	0.417351056	205.113...	0.031	0.32043
19	SOIPNS...	Grim Prosp...	477268.0	Grim Prospe...	AR3WM...	Schizoid		0.349192218	166.007...	0.23	1.58691
20	SODLF...	Like A Deck ...	527484.0	Meet the F@...	ARNFT...	Bedwetters		0.414079473	202.918...	0.017	1.66772
21	SOOCU...	Short On Id...	221479.0	Pezcore	ARG8D...	Less Than ...	0.407233015	0.554564302	257.018...	0.748	0.26527
22	SOHM...	The Burden ...	204336.0	The Warrior ...	ARD2N...	Dropkick M...	0.651569826	0.568491815	175.960...	0.175	1.16764
23	SOJHG...	Bad Man St...	795316.0	Beenie Man -...	ARVUN...	Beenie Man	0.301681757	0.481339309	179.513...	0.436	0.10281
24	SOZEH...	Always And ...	128122.0	Discovering ...	AR10U...	Silverstein	0.658623927	0.605507136	229.642	0.551	0.82669
25	SOCUD...	Til The Victo...	68914.0	Walking Mira...	ARP32...	Vanessa B...		0.408646827	231.548...	0.019	1.6086
26	SOYLVT...	Letters to N...	469438.0	Out of the Cl...	ART8B...	Planetakis		0.357049054	234.735...	0.138	2.58307
27	SONYR...	Burning In T...	706005.0	A Match & So...	ARWYV...	The Suicide...	0.528782481	0.467538444	95.68608	0.987	0.90181
28	SOYFVF...	Armageddo...	214393.0	Bondage Go...	ARNNQ...	Belphegor	0.624839791	0.473083192	308.766...	0.035	0.37025

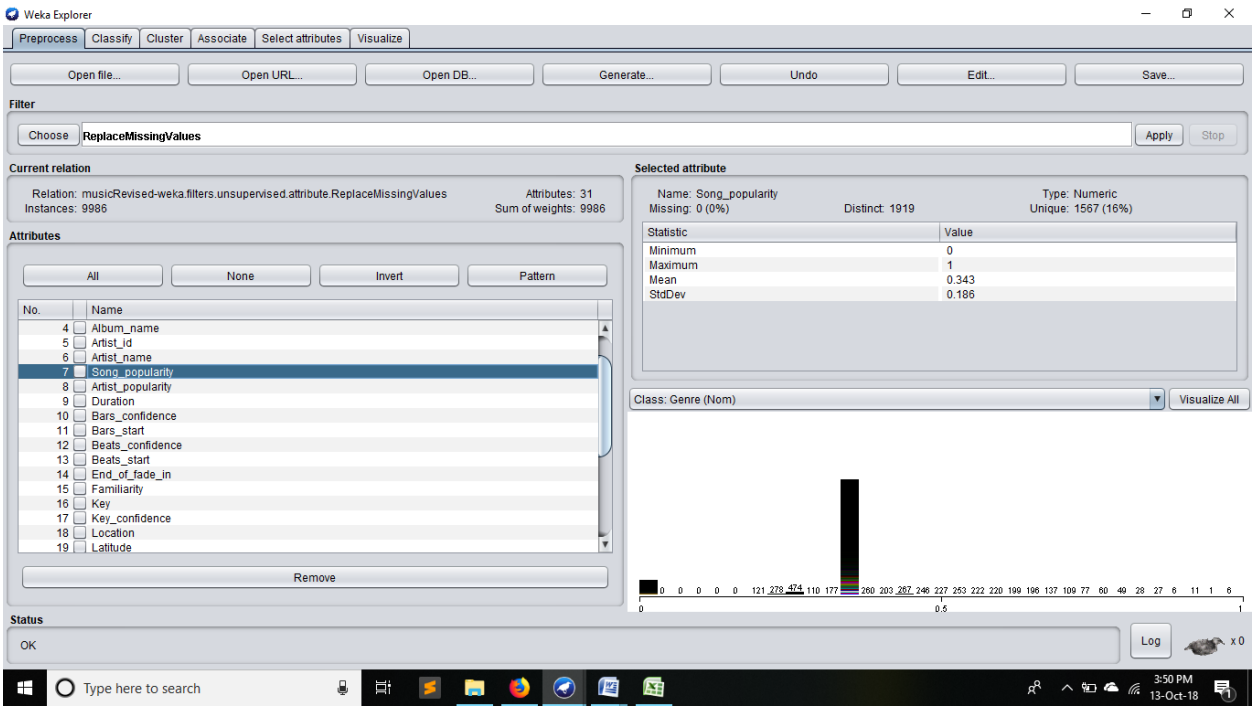
Dataset Before Cleaning

Song_popularity



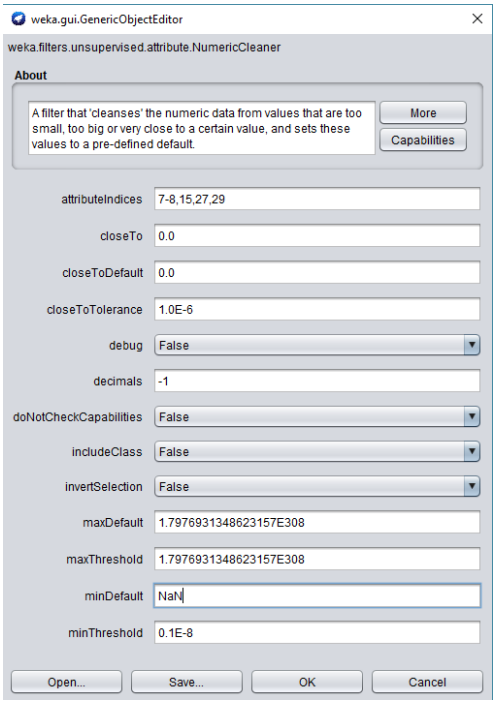
After ReplaceMissingValues, under unsupervized.attribute. ReplaceMissingValues.

Song_popularity



Zero values can be marked as missing values in Weka using the NumericalCleaner filter.

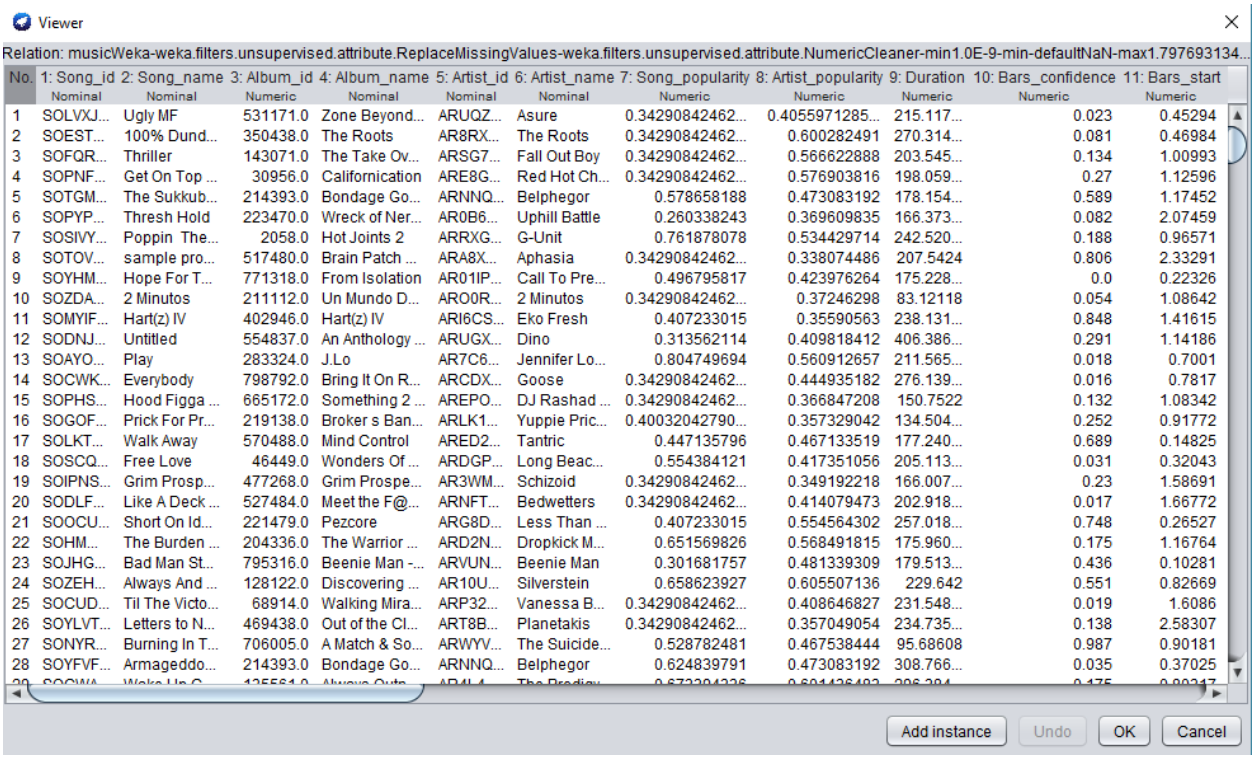
The attributes for marking zero values as null values are: Song_popularity, Artist_popularity, Familiarity, Tempo, Time_signature



We can see that the zero values are now converted as missing/null values. Now we can apply the ReplaceMissingValues filter.

Now there are no null and no zero values in the dataset.

The Dataset Obtained



No.	1: Song_id	2: Song_name	3: Album_id	4: Album_name	5: Artist_id	6: Artist_name	7: Song_popularity	8: Artist_popularity	9: Duration	10: Bars_confidence	11: Bars_start
	Nominal	Nominal	Numeric	Nominal	Nominal	Nominal	Numeric	Numeric	Numeric	Numeric	Numeric
1	SOLVXJ...	Ugly MF	531171.0	Zone Beyond...	ARUQZ...	Asure	0.34290842462...	0.4055971285...	215.117...	0.023	0.45294
2	SOEST...	100% Dund...	350438.0	The Roots	AR8RX...	The Roots	0.34290842462...	0.600282491	270.314...	0.081	0.46984
3	SOFQR...	Thriller	143071.0	The Take Ov...	ARSG7...	Fall Out Boy	0.34290842462...	0.566622888	203.545...	0.134	1.00993
4	SOPNF...	Get On Top ...	30956.0	Californication	ARE8G...	Red Hot Ch...	0.34290842462...	0.576903816	198.059...	0.27	1.12596
5	SOTGM...	The Sukkub...	214393.0	Bondage Go...	ARNNQ...	Belphegor	0.578658188	0.473083192	178.154...	0.589	1.17452
6	SOPYP...	Thresh Hold	223470.0	Wreck of Ner...	AR0B6...	Uphill Battle	0.260338243	0.369609835	166.373...	0.082	2.07459
7	SOSIVY...	Poppin The...	2058.0	Hot Joints 2	ARRXG...	G-Unit	0.761878078	0.534429714	242.520...	0.188	0.96571
8	SOTOV...	sample pro...	517480.0	Brain Patch ...	ARA8X...	Aphasia	0.34290842462...	0.338074486	207.5424	0.806	2.33291
9	SOYHM...	Hope For T...	771318.0	From Isolation	AR01IP...	Call To Pre...	0.496795817	0.423976264	175.228...	0.0	0.22326
10	SOZDA...	2 Minutos	211112.0	Un Mundo D...	ARO0R...	2 Minutos	0.34290842462...	0.37246298	83.12118	0.054	1.08642
11	SOMYIF...	Hart(z) IV	402946.0	Hart(z) IV	ARI6CS...	Eko Fresh	0.407233015	0.35590563	238.131...	0.848	1.41615
12	SODNJ...	Untitled	554837.0	An Anthology ...	ARUGX...	Dino	0.313562114	0.409818412	406.386...	0.291	1.14186
13	SOAYO...	Play	283324.0	J.Lo	AR7C6...	Jennifer Lo...	0.804749694	0.560912657	211.565...	0.018	0.7001
14	SOCWK...	Everybody	798792.0	Bring It On R...	ARCDX...	Goose	0.34290842462...	0.444935182	276.139...	0.016	0.7817
15	SOPHS...	Hood Figga ...	665172.0	Something 2 ...	AREPO...	DJ Rashad ...	0.34290842462...	0.366847208	150.7522	0.132	1.08342
16	SOGOF...	Prick For Pr...	219138.0	Broker s Ban...	ARLK1...	Yuppie Pric...	0.40032042790...	0.357329042	134.504...	0.252	0.91772
17	SOLKT...	Walk Away	570488.0	Mind Control	ARED2...	Tantric	0.447135796	0.467133519	177.240...	0.689	0.14825
18	SOSCQ...	Free Love	46449.0	Wonders Of ...	ARDGP...	Long Beac...	0.554384121	0.417351056	205.113...	0.031	0.32043
19	SOIPNS...	Grim Prosp...	477268.0	Grim Prospe...	AR3WM...	Schizoid	0.34290842462...	0.349192218	166.007...	0.23	1.58691
20	SODLF...	Like A Deck ...	527484.0	Meet the F@...	ARNFT...	Bedwetters	0.34290842462...	0.414079473	202.918...	0.017	1.66772
21	SOOCU...	Short On Id...	221479.0	Pezcore	ARG8D...	Less Than ...	0.407233015	0.554564302	257.018...	0.748	0.26527
22	SOHM...	The Burden ...	204336.0	The Warrior ...	ARD2N...	Dropkick M...	0.651569826	0.568491815	175.960...	0.175	1.16764
23	SOJHG...	Bad Man St...	795316.0	Beenie Man -...	ARVUN...	Beenie Man	0.301681757	0.481339309	179.513...	0.436	0.10281
24	SOZEH...	Always And ...	128122.0	Discovering ...	AR10U...	Silverstein	0.658623927	0.605507136	229.642	0.551	0.82669
25	SOCUD...	Til The Victo...	68914.0	Walking Mira...	ARP32...	Vanessa B...	0.34290842462...	0.408646827	231.548...	0.019	1.6086
26	SOYLVT...	Letters to N...	469438.0	Out of the Cl...	ART8B...	Planetakis	0.34290842462...	0.357049054	234.735...	0.138	2.58307
27	SONYR...	Burning In T...	706005.0	A Match & So...	ARWYV...	The Suicide...	0.528782481	0.467538444	95.68608	0.987	0.90181
28	SOYFVF...	Armageddo...	214393.0	Bondage Go...	ARNNQ...	Belphegor	0.624839791	0.473083192	308.766...	0.035	0.37025

ii. Algorithm Implementation

- ✓ Two algorithms – KNN (K Nearest Neighbor) and Random Forest are implemented.

- ✓ KNN: The KNN uses the concept of analogy. The basic idea is that the instance is that what it resembles. So if a particular instance is to be classified, at its K-closest (most similar) neighbors are considered. The point is classified as the majority class in those neighbors.

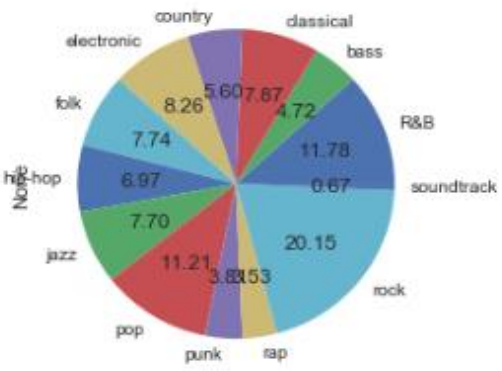
- ✓ As the name suggests, random forest is a forest of random decision tress. The algorithm randomly selects k features from total m features. Among these k features, the node d is calculated which is the best split point. The best split concept is used to split the node into daughter nodes. These steps are repeated a n number of times to get n trees. Basically, Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. The random algorithm used in wide varieties applications namely, banking, medicine, e – commerce, stock market etc.

- ✓ Python tool is used for the implementation.

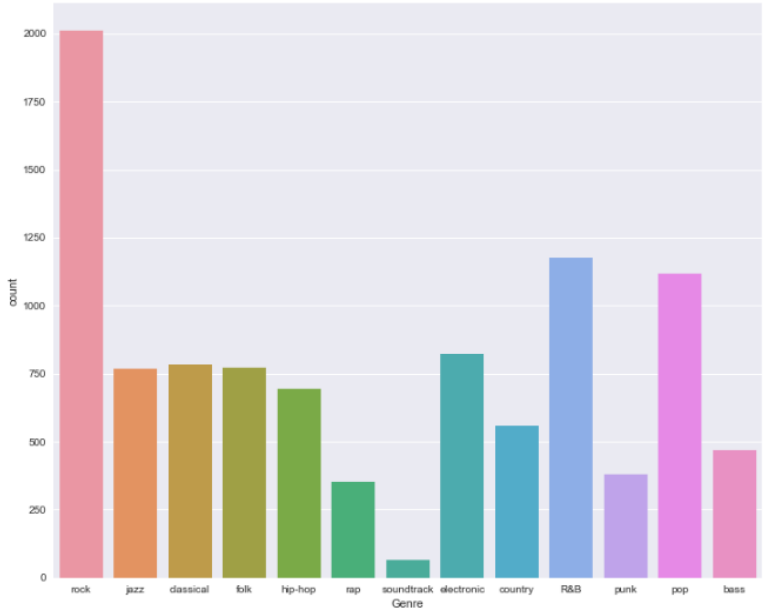
iii. Data Visualization

Using Jupyter and Python Libraries

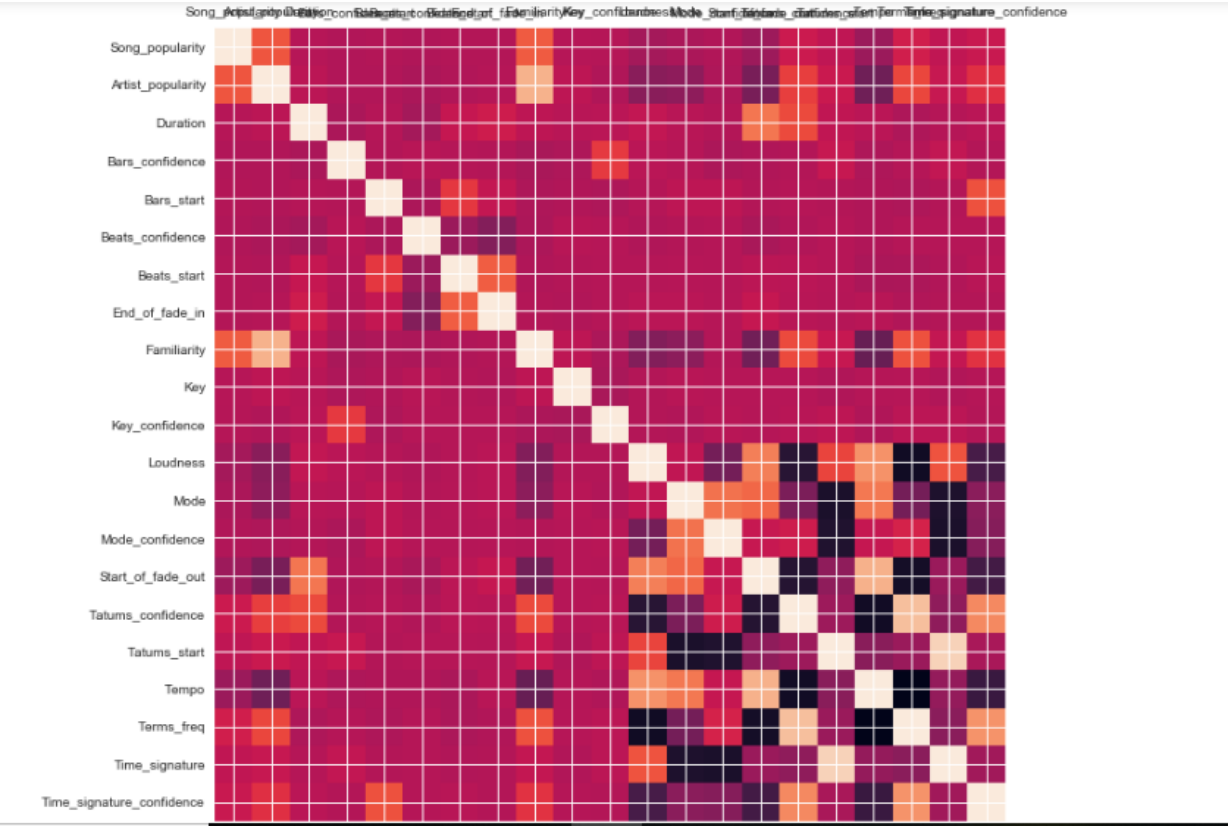
Pie Chart



Bar Chart

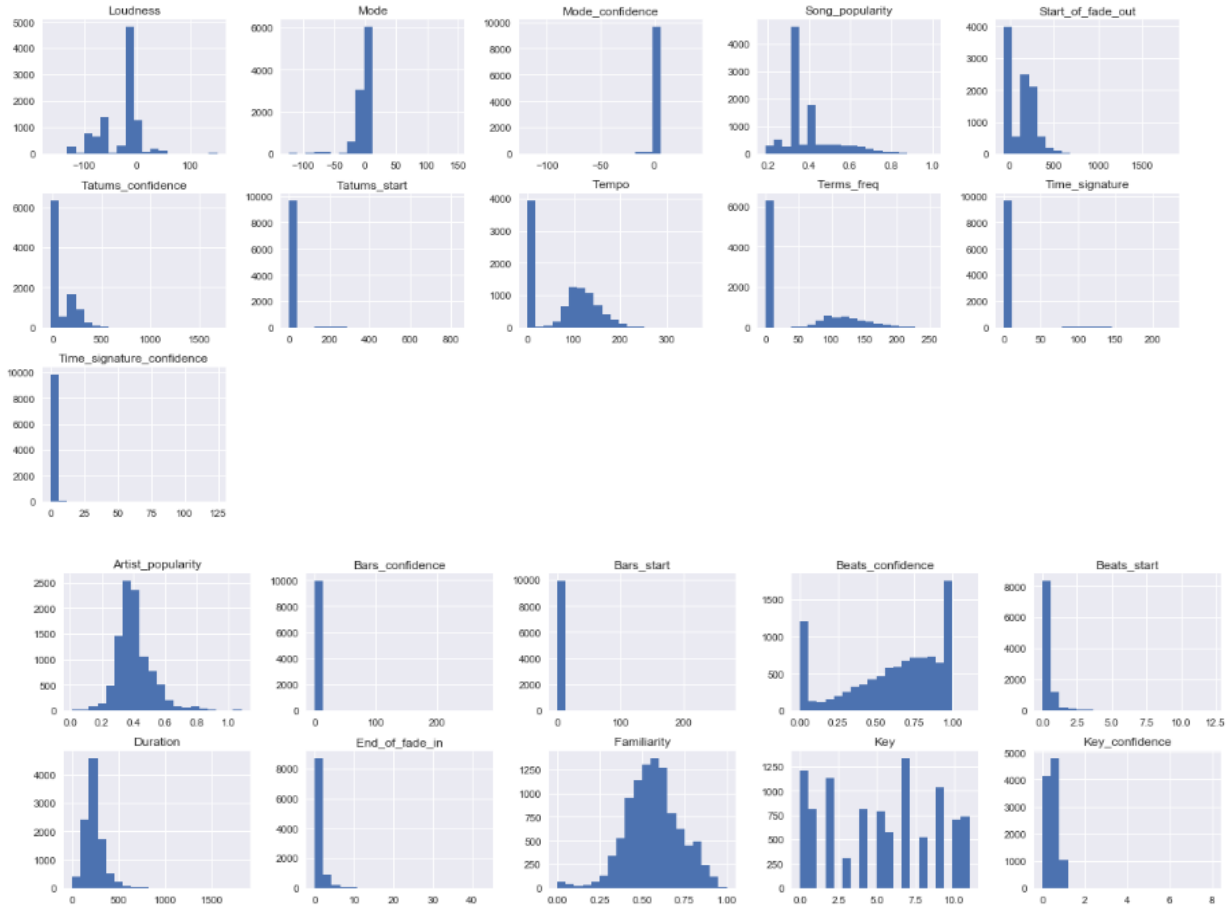


Identifying correlation in data graphically



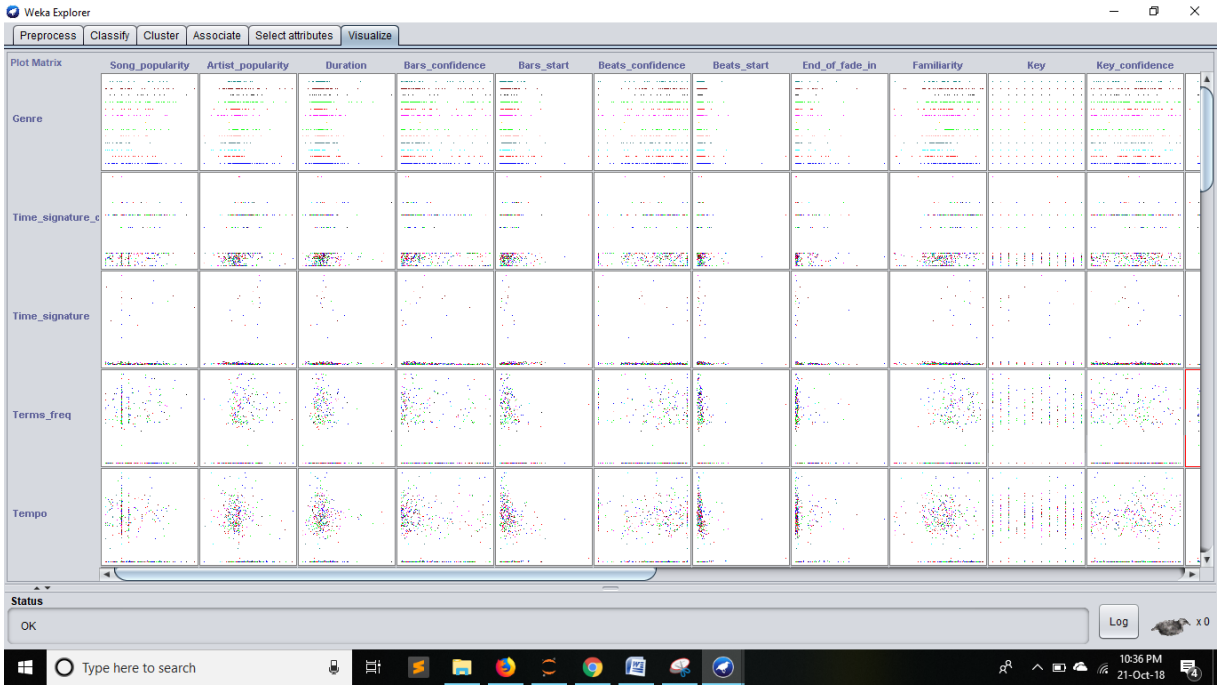
In the above visualization plot, light color represents maximum correlation and the dark color represents minimum correlation. We can see none of the variable have proper correlation with any of the other variables.

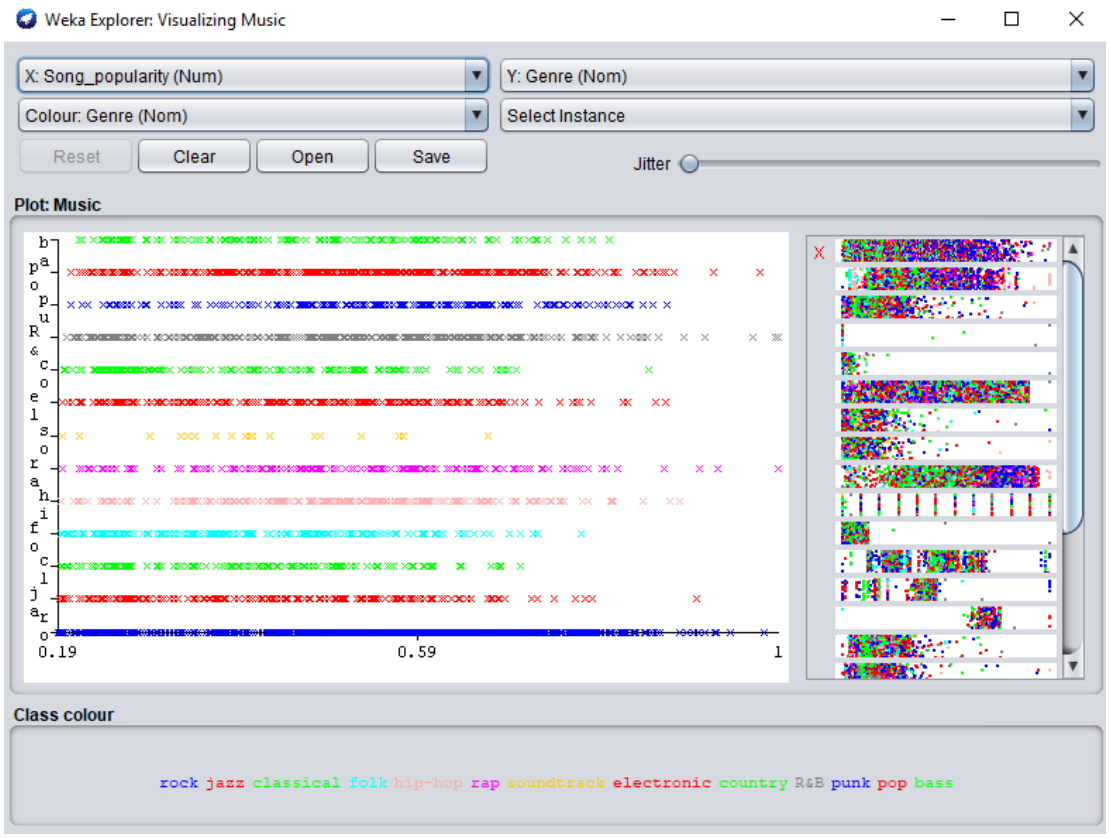
Histogram

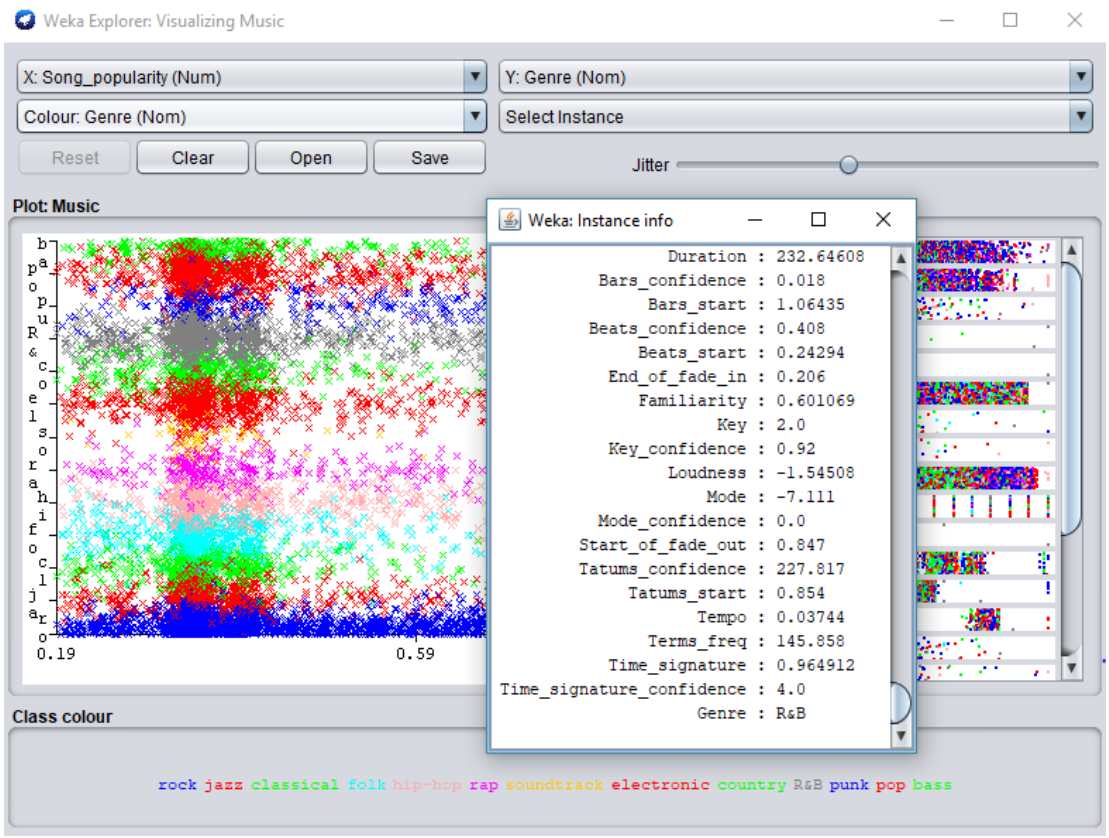


Using WEKA

Scatter Plots







Jitter is useful when you have a lot of dots overlaying each other and it is hard to see what is going on. Jitter will add some random noise to the data in the plots, spread out the points a bit and help you see what is going on.

8. Evaluation measures

K Nearest Neighbor

```
Accuracy Score= 0.22069138276553107
[[144   0  34   1  20  24   2   1  20   0   0 220   0]
 [ 26   0  19   0  32  11   1   1  10   1   0  80   0]
 [ 23   0  60   0  10  24   0   0  27   0   0 147   0]
 [ 63   0  18   1   3  13   1   0  22   2   0 113   0]
 [ 19   0  19   0  60  18   3   1  37   0   0 187   0]
 [ 18   0  29   0  20  29   2   4  33   0   0 163   0]
 [ 24   0  14   0  26  19   2   2  25   2   0 174   0]
 [ 41   0  33   0  28  12   3   1  27   1   0 159   0]
 [ 58   0  40   0  12  28   2   1  46   1   0 243   0]
 [ 26   0  11   0   5   5   0   0  11   6   0  89   0]
 [  7   0   8   0   1   6   1   0   9   0   0 103   0]
 [127   0  49   1  29  34   5   3  52   2   0 532   0]
 [  2   0   5   0   0   0   0   0   1   0   0  22   0]]
```

```
Classification Report
             precision    recall  f1-score   support

        R&B       0.25      0.31      0.28       466
       bass       0.00      0.00      0.00       181
  classical       0.18      0.21      0.19       291
    country       0.33      0.00      0.01       236
 electronic       0.24      0.17      0.20       344
       folk       0.13      0.10      0.11       298
    hip-hop       0.09      0.01      0.01       288
       jazz       0.07      0.00      0.01       305
        pop       0.14      0.11      0.12       431
       punk       0.40      0.04      0.07       153
        rap       0.00      0.00      0.00       135
       rock       0.24      0.64      0.35       834
 soundtrack       0.00      0.00      0.00        30

avg / total       0.19      0.22      0.16      3992
```

Varying the value of k

```
Accuracy is  15.33066132264529 % for k-Value: 5
Accuracy is  19.514028056112224 % for k-Value: 25
Accuracy is  20.66633266533066 % for k-Value: 50
Accuracy is  22.069138276553108 % for k-Value: 100
Accuracy is  20.691382765531063 % for k-Value: 500
Accuracy is  20.415831663326653 % for k-Value: 1000
Accuracy is  20.89178356713427 % for k-Value: 5000
```

Random Forest

The confusion matrix gives the predicted value vs actual value comparison. Based on the confusion matrix, sensitivity, specificity and accuracy can be calculated.

```
Confusion Matrix
[[164   7  20  21  18  20   5  24  28   5   3  70   1]
 [ 27  30  12   4  17   8   9   7  16   1   1  22   0]
 [ 31   8  98  12  13  26   9  16  16   1   0  28   1]
 [ 42   4  19  43  10  10   6   9  14   1   2  22   0]
 [ 45  19  22  12  57  11  11  13  21   1   6  52   0]
 [ 27  10  42   9  25  36  14  13  24   2   1  45   1]
 [ 33  11  17   4  20  14  46   9  16   2   4  54   2]
 [ 45   9  37  10  20  20   8  45  23   1   3  33   1]
 [ 48  21  32   9  33  21  17  14  85   3   6  89   0]
 [ 19   5   4   3   8   9   4   5  15  17   2  40   1]
 [ 14   5   1   3   6   5   6   5   9   7   9  26   0]
 [ 98  14  34  24  63  36  24  27  51  23  11 279   1]
 [  0   0   4   1   1   1   1   0   2   1   0   2   3]]
```

```
Classification Report
             precision    recall  f1-score   support

        R&B       0.28      0.42      0.34       386
       bass       0.21      0.19      0.20       154
  classical       0.29      0.38      0.33       259
    country       0.28      0.24      0.26       182
 electronic       0.20      0.21      0.20       270
       folk       0.17      0.14      0.15       249
    hip-hop       0.29      0.20      0.23       232
       jazz       0.24      0.18      0.20       255
        pop       0.27      0.22      0.24       378
       punk       0.26      0.13      0.17       132
        rap       0.19      0.09      0.12        96
       rock       0.37      0.41      0.39       685
 soundtrack       0.27      0.19      0.22        16

avg / total       0.27      0.28      0.27      3294
```

The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights recall more than precision by a factor of beta. beta == 1.0 means recall and precision are equally important.

The support is the number of occurrences of each class in y_true.

9. Results and discussions

K Nearest Neighbor

This is the output for KNN classifier. Predicted and Actual Genre are shown for all 10k instances.

```
The approximate split ratio is:  0.67
Train set: 6716
Test set: 3263

==================k-Nearest Neighbour Classifier====================

Predicted Genre= 'R&B', Actual Genre= 'jazz'
Predicted Genre= 'rock', Actual Genre= 'rock'
Predicted Genre= 'classical', Actual Genre= 'classical'
Predicted Genre= 'rap', Actual Genre= 'hip-hop'
Predicted Genre= 'jazz', Actual Genre= 'electronic'
Predicted Genre= 'rock', Actual Genre= 'jazz'
Predicted Genre= 'jazz', Actual Genre= 'jazz'
Predicted Genre= 'country', Actual Genre= 'country'
Predicted Genre= 'pop', Actual Genre= 'R&B'
Predicted Genre= 'classical', Actual Genre= 'folk'
Predicted Genre= 'classical', Actual Genre= 'folk'
Predicted Genre= 'pop', Actual Genre= 'rock'
Predicted Genre= 'pop', Actual Genre= 'rock'
Predicted Genre= 'classical', Actual Genre= 'pop'
```

```
Predicted Genre= 'rock', Actual Genre= 'rock'
Predicted Genre= 'pop', Actual Genre= 'rock'
Predicted Genre= 'electronic', Actual Genre= 'rock'
Predicted Genre= 'rock', Actual Genre= 'hip-hop'
Predicted Genre= 'bass', Actual Genre= 'rock'
Predicted Genre= 'rock', Actual Genre= 'pop'
Predicted Genre= 'R&B', Actual Genre= 'R&B'
Predicted Genre= 'rock', Actual Genre= 'rock'
Predicted Genre= 'rock', Actual Genre= 'punk'
Predicted Genre= 'pop', Actual Genre= 'rock'
Predicted Genre= 'classical', Actual Genre= 'pop'
Predicted Genre= 'rock', Actual Genre= 'pop'
Predicted Genre= 'rap', Actual Genre= 'rap'
Predicted Genre= 'country', Actual Genre= 'R&B'
Predicted Genre= 'pop', Actual Genre= 'rock'
Predicted Genre= 'country', Actual Genre= 'jazz'
Predicted Genre= 'R&B', Actual Genre= 'rock'
Predicted Genre= 'classical', Actual Genre= 'rock'
Predicted Genre= 'punk', Actual Genre= 'pop'
The Accuracy of the Model: 18.90897946674839%
```

The accuracy of the model is 18.91%

Using libraries, we can also obtain other results

```
Accuracy Score= 0.22069138276553107
[[144   0  34   1  20  24   2   1  20   0   0 220   0]
 [ 26   0  19   0  32  11   1   1  10   1   0  80   0]
 [ 23   0  60   0  10  24   0   0  27   0   0 147   0]
 [ 63   0  18   1   3  13   1   0  22   2   0 113   0]
 [ 19   0  19   0  60  18   3   1  37   0   0 187   0]
 [ 18   0  29   0  20  29   2   4  33   0   0 163   0]
 [ 24   0  14   0  26  19   2   2  25   2   0 174   0]
 [ 41   0  33   0  28  12   3   1  27   1   0 159   0]
 [ 58   0  40   0  12  28   2   1  46   1   0 243   0]
 [ 26   0  11   0   5   5   0   0  11   6   0  89   0]
 [  7   0   8   0   1   6   1   0   9   0   0 103   0]
 [127   0  49   1  29  34   5   3  52   2   0 532   0]
 [  2   0   5   0   0   0   0   0   1   0   0  22   0]]
```

```
Classification Report
             precision    recall  f1-score   support

        R&B       0.25      0.31      0.28       466
       bass       0.00      0.00      0.00       181
  classical       0.18      0.21      0.19       291
    country       0.33      0.00      0.01       236
 electronic       0.24      0.17      0.20       344
       folk       0.13      0.10      0.11       298
    hip-hop       0.09      0.01      0.01       288
       jazz       0.07      0.00      0.01       305
        pop       0.14      0.11      0.12       431
       punk       0.40      0.04      0.07       153
        rap       0.00      0.00      0.00       135
       rock       0.24      0.64      0.35       834
 soundtrack       0.00      0.00      0.00        30

avg / total       0.19      0.22      0.16      3992
```

We get optimal result where the k value is the square root of the number of instances (at 100)

```
Accuracy is  15.33066132264529 % for k-Value: 5
Accuracy is  19.514028056112224 % for k-Value: 25
Accuracy is  20.66633266533066 % for k-Value: 50
Accuracy is  22.069138276553108 % for k-Value: 100
Accuracy is  20.691382765531063 % for k-Value: 500
Accuracy is  20.415831663326653 % for k-Value: 1000
Accuracy is  20.89178356713427 % for k-Value: 5000
```

Random Forest

This is the output for Random Forest classifier. Accuracy is shown for varying number of trees.

```
_____________Random Forest Implementation - 1.5k instances_____________

Number of Folds: 5
Value of n_features: 4

_____________________________________________________________________

Trees: 1
Scores: [29.333333333333332, 23.333333333333332, 31.0, 24.333333333333336, 30.0]
Mean Accuracy: 27.600%

Trees: 5
Scores: [28.999999999999996, 37.0, 29.333333333333332, 34.0, 36.666666666666664]
Mean Accuracy: 33.200%

Trees: 10
Scores: [34.333333333333336, 34.0, 32.666666666666664, 34.333333333333336, 27.666666666666668]
Mean Accuracy: 32.600%
```

```
Confusion Matrix
[[164   7  20  21  18  20   5  24  28   5   3  70   1]
 [ 27  30  12   4  17   8   9   7  16   1   1  22   0]
 [ 31   8  98  12  13  26   9  16  16   1   0  28   1]
 [ 42   4  19  43  10  10   6   9  14   1   2  22   0]
 [ 45  19  22  12  57  11  11  13  21   1   6  52   0]
 [ 27  10  42   9  25  36  14  13  24   2   1  45   1]
 [ 33  11  17   4  20  14  46   9  16   2   4  54   2]
 [ 45   9  37  10  20  20   8  45  23   1   3  33   1]
 [ 48  21  32   9  33  21  17  14  85   3   6  89   0]
 [ 19   5   4   3   8   9   4   5  15  17   2  40   1]
 [ 14   5   1   3   6   5   6   5   9   7   9  26   0]
 [ 98  14  34  24  63  36  24  27  51  23  11 279   1]
 [  0   0   4   1   1   1   1   0   2   1   0   2   3]]
```

```
Classification Report
             precision    recall  f1-score   support

        R&B       0.28      0.42      0.34       386
       bass       0.21      0.19      0.20       154
  classical       0.29      0.38      0.33       259
    country       0.28      0.24      0.26       182
 electronic       0.20      0.21      0.20       270
       folk       0.17      0.14      0.15       249
    hip-hop       0.29      0.20      0.23       232
       jazz       0.24      0.18      0.20       255
        pop       0.27      0.22      0.24       378
       punk       0.26      0.13      0.17       132
        rap       0.19      0.09      0.12        96
       rock       0.37      0.41      0.39       685
 soundtrack       0.27      0.19      0.22        16

avg / total       0.27      0.28      0.27      3294
```

10.Conclusion

The system proposed in this report uses two classifiers namely, K Nearest Neighbor and Random Forest to predict Music Genre. K Nearest Neighbor and Random Forest are both classification algorithms and come under supervised learning i.e. training data is needed to build the model.

The dataset used for this project is a subset of the million song dataset. Initially, the dataset contained 32 attributes and 10k instances. The attributes of the dataset are further reduced as the dataset goes through the process of data cleaning.

The KNN uses the concept of analogy. The basic idea is that the instance is that what it resembles. So if a particular instance is to be classified, at its K-closest (most similar) neighbors are considered. The point is classified as the majority class in those neighbors.

Random forest is a forest of random decision tress. The algorithm randomly selects k features from total m features. Among these k features, the node d is calculated which is the best split point. The best split concept is used to split the node into daughter nodes. These steps are repeated a n number of times to get n trees. Basically, Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

In this project, multiple tools are used. Weka is used for data pre-processing and data visualization. Python (jupyter notebook) is used implementing the classifiers.

11.References

- Omar Diab, Anthony Manero, and Reid Watson. Musical Genre Tag Classification With Curated and Crowdsourced Datasets. Stanford University, Computer Science, 1 edition, 2012.
- N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. IEEE Signal Process. Mag., 23(2):133{141, 2006.
- Sox.sourceforge.net. Sox - sound exchange | homepage, 2015.
- Bob L. Sturm. Classification accuracy is not enough. Journal of Intelligent Information Systems, 41(3):371-406, 2013.
- G. Tzanetakis and P. Cook. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5):293{302, 2002.
- Jan Wulfing and Martin Riedmiller. Unsupervised learning of local features for music classification. In ISMIR, pages 139{144, 2012.
- Changsheng Xu, MC Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. In Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on, volume 5, pages V{429. IEEE, 2003.
- Dataset Link: https://think.cs.vt.edu/corgis/csv/music/music.html