Music Genre Recognition Using Weighted Majority Algorithm

Jivjot Singh
Department of Computing Science
Simon Fraser University
jivjots@sfu.ca

Mangesh Bhangare
Department of Computing Science
Simon Fraser University
mbhangar@sfu.ca

Anuj Issar
Department of Computing Science
Simon Fraser University
aissar@sfu.ca

Bikramdeep Singh
Department of Computing Science
Simon Fraser University
bsa61@sfu.ca

Abstract

This project presents an approach for the music genre classification problem. The proposed approach uses temporal feature vector and weighted voting to improve the prediction accuracy. Classical machine learning algorithms such as Naïve Bayes, K-Nearest Neighbors, and Support Vector Machines are employed and weighted voting procedures were employed in order to enhance final prediction results. Experiments were carried out on a dataset obtained from Music Analysis, Retrieval and Synthesis for Audio Signals (MARSYAS) which is an open source software framework and data collection of audio files. The dataset contains 1,000 audio files categorized in 10 musical genres. Experimental results show that the proposed ensemble approach produces better results than the ones obtained from individual classifiers.

1 Introduction

Duke Ellington who once very wisely said: There are simply two kinds of music, good music and the other kind. The only yardstick by which the result should be judged is simply that of how it sounds. If it sounds good, it's successful; if it doesn't it has failed. Exploring different genres of music is about wanting to know a little bit more about some of the things we humans can do, the feelings we didn't want to leave unsaid; the messages we've wanted to get across. Today with the amount of music we have, automatic procedures capable of dealing with large amounts of music in digital formats are imperative, and Music Information Retrieval (MIR) [6] has become an important research area. An important task in MIR is Music Genre Classification problem, music genres are categorical labels created by experts in order to identify the style of the music. The music genre is a descriptor that is largely used to organize collections of digital music. It is not only a crucial metadata in large music databases and electronic music distribution (EMD).

The music can be considered as a high-dimensional digital time-variant signal and considering the amount of music data we have today; it is a good opportunity to automate music genre classification using temporal feature vectors. The Dataset is obtained from Music Analysis, Retrieval and Synthesis for Audio Signals (MARSYAS) [1]. The approach involves classical machine algorithms such as Naïve-Bayes, k Nearest-Neighbors, and Support Vector Machines and using weighted voting procedures to improve final prediction results. The Naïve Bayes is implemented with 10-fold cross validation using ‘e1071’ [9] package in R. For Support Vector Machines, we have implemented ‘libSVM’ [7] with 10-fold cross validation. The K-Nearest Neighbor is implemented with ‘kkNN’ [8] package of R.

2 Dataset and Features

2.1 Dataset

The dataset we are using is from Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals) which is an open source software framework for audio processing with specific emphasis on Music Information Retrieval applications. The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050Hz Mono 16-bit audio files in .wav format.

2.2 Preprocessing

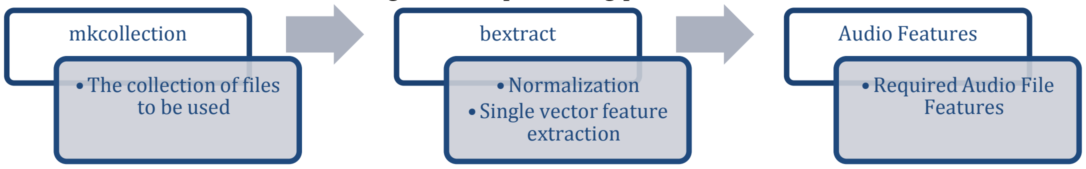
The first step in preprocessing is to divide the dataset in training, testing and validation sets. For this we have written a python script, which divides the dataset for each genre into 70 songs for testing, 30 songs for testing and 30 songs for training. So finally we have 150 songs in training set and 150 each in validation and training set. After that we have used ‘mkcollection’ which is a simple utility provided by Marsyas framework for creating collection files. The ‘mkcollection’ utility takes the folder containing all songs as an input and the audio files residing in that directory or any subdirectories are added to the collection.

2.3 Feature Extraction

For features extraction, we are using MARSYAS which is implemented in C++ and retains the ability to output feature extraction data to ARFF format. With the tool bextract.exe, the following features are extracted: Zero Crossings, Spectral Centroid [3], Spectral Flux, Spectral Rolloff [4], Mel-Frequency Cepstral Coefficients (MFCC), and chroma features [2]. The total number of features extracted are 124.

A zero-crossing is a point where the sign of a mathematical function changes (e.g. from positive to negative), represented by a crossing of the axis (zero value) in the graph of the function. The spectral centroid is a measure used in digital signal processing to characterize a spectrum. It indicates where the "center of mass" of the spectrum is. Perceptually, it has a robust connection with the impression of "brightness" of a sound. Spectral flux is a measure of how quickly the power spectrum of a signal is changing, calculated by comparing the power spectrum for one frame against the power spectrum from the previous frame. Spectral rolloff point is defined as the Nth percentile of the power spectral distribution, where N is usually 85% or 95%. The rolloff point is the frequency below which the N% of the magnitude distribution is concentrated. In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave.

Figure 1. Preprocessing process


mkcollection
• The collection of files to be used
bextract
• Normalization
• Single vector feature extraction
Audio Features
• Required Audio File Features

3 Techniques

3.1 K-nearest neighbors

The first machine learning technique used is the simple K nearest neighbor. We used R library to implement the classifier. The results by the standard R library were not satisfactory, and we tried the new KKNN, a library for weighted k-nearest neighbors classification.

The KKNN uses the Minkowski distance to calculate the nearest neighbors of the given data point. The library provides various options for kernels implementation like "triweight", "cos", "inv", "gaussian", our results were best for the Gaussian kernel. We created a script to determine the best kernel and the optimum number of neighbors for the implementation. The bet results were generated by using a Gaussian kernel with five neighbors.

3.2 Support Vector Machines

Support Vector Machines (SVM) are considered one of the best machine learning algorithm for multiclass classifications along with the neural networks. Our dataset being small, SVM classifier was expected to perform best among all the techniques used.

The libsvm library is used to implement the SVM [5] classifier for the dataset. The libsvm uses the one vs one classification for all the pairs possible to generate the multiclass classifier. Octave is used to implement the libsvm classifier. We used python script grid.py, provided with the library to find the optimal parameters for the radial kernel to train the data. The parameters for which the validation set was lowest were used to create the final model.

3.3 Naïve Bayes

The e1071 library of R is used to implement the classifier for Naïve Bayes. It computes the conditional posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule. It distributes each class variable as the Gaussian distribution and the parameter tuning is done by the library itself.

3.3 Neural Networks

Deep learning is one of the forefront runners in the field of machine learning. The neural network implementation was done using both library and self-written code. But opposite to the initial expectations both the library and ours implementation performed poorly for the data set. We ran various validations and parameter tuning runs but results were not comparable to those of the SVM and K-nearest neighbors. We came to the conclusion that the data was not enough to be classified by the neural network classifier.

3.4 Weighted Majority Algorithm

Weighted Majority Algorithm (WMA) is a meta-learning algorithm used to build a joint prediction model from a list of prediction algorithms, which could be any type of learning algorithms, classifiers, or even real human experts. The algorithm assumes that we have no prior knowledge about the accuracy of the algorithms in the list, but there are sufficient reasons to believe that one or more will perform well*.

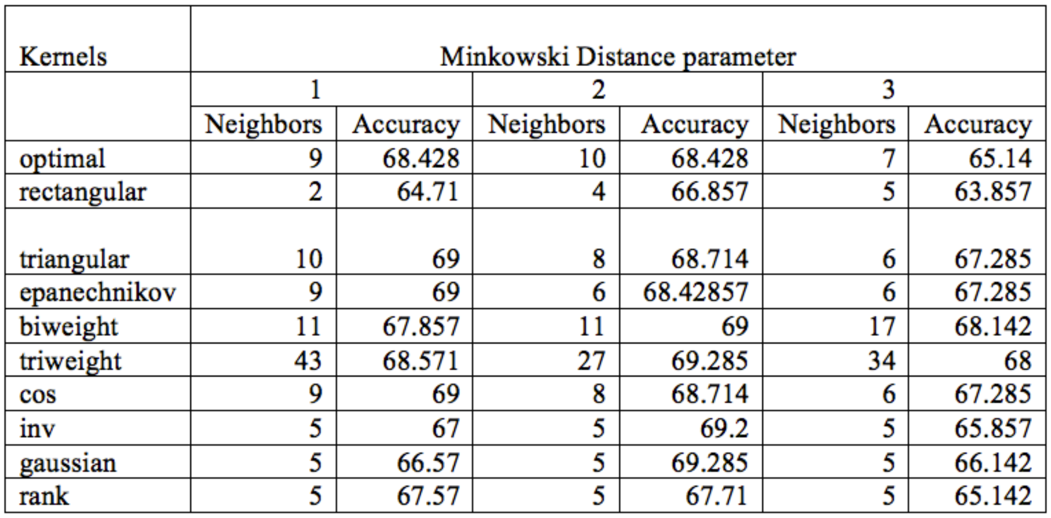
We use all but neural network model to build new classifier. We created a script to find the different optimum weights for K-near neighbors, SVM and Naïve Bayes. It runs for all possible weight combinations with step size of 0.01 with range 0 to 100 for each model. As expected the contribution of Naïve Bayes was the lowest among the three classifier as it is the weakest classifier among the three and provides less accuracy as compared to the other algorithms used.

4 Experiments

4.1 K-nearest Neighbors

To search for the best kernel and its parameters for the data set we created a script and ran for some kernels and range of their parameters. Table1 list the results for the kernels for optimum number of neighbors on cross validation folds. The best cross validation set accuracy was achieved for the Gaussian kernel with 5 neighbors.

Table1: KKNN Cross Validation Accuracy



Kernels	Minkowski Distance parameter					
	1		2		3	
	Neighbors	Accuracy	Neighbors	Accuracy	Neighbors	Accuracy
optimal	9	68.428	10	68.428	7	65.14
rectangular	2	64.71	4	66.857	5	63.857
triangular	10	69	8	68.714	6	67.285
epanechnikov	9	69	6	68.42857	6	67.285
biweight	11	67.857	11	69	17	68.142
triweight	43	68.571	27	69.285	34	68
cos	9	69	8	68.714	6	67.285
inv	5	67	5	69.2	5	65.857
gaussian	5	66.57	5	69.285	5	66.142
rank	5	67.57	5	67.71	5	65.142

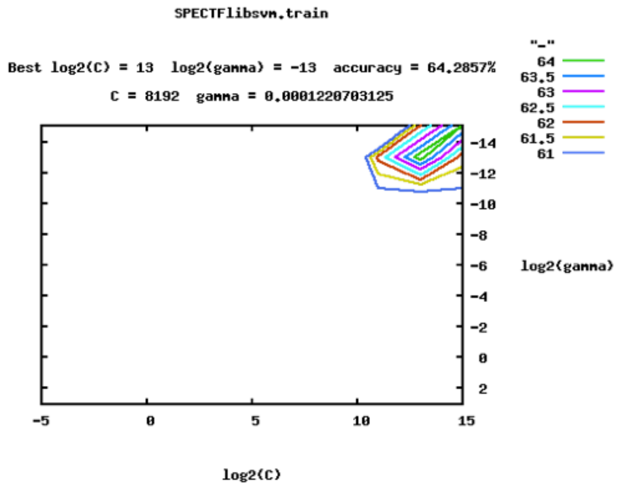
4.2 Naive Bayes

The Naïve Bayes library does not require any parameter tuning and selection. Hence only one run required for the cross validation accuracy of 55.33%, which was the lowest among the three.

4.3 SVM

For SVM model and parameters selection a script gird.py was provided by the libsvm library. Figure 3 shows the run of the script and selection of gamma and C (sigma) values for the radial kernel. The shows the optimal value selection.

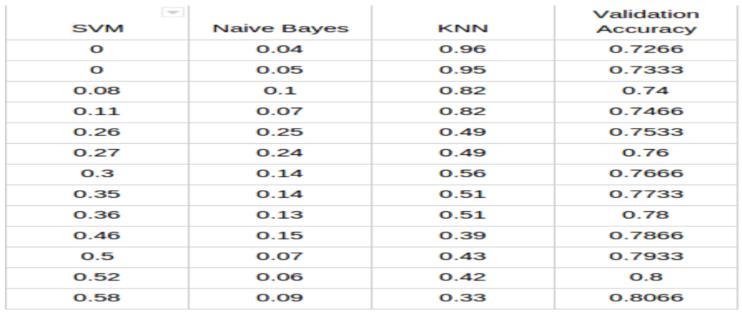
Figure 3: Grid.py run for parameter selection



4.4 Weighted Majority Algorithm

For optimal weighing of the various models we wrote a script which ran for all the possible weighing options from 0 – 100 for all the models. The Table 2 shows some of the runs of the algorithm. The last row of the table is the best weighing distribution for the models.

Table 2: WMA weights distribution

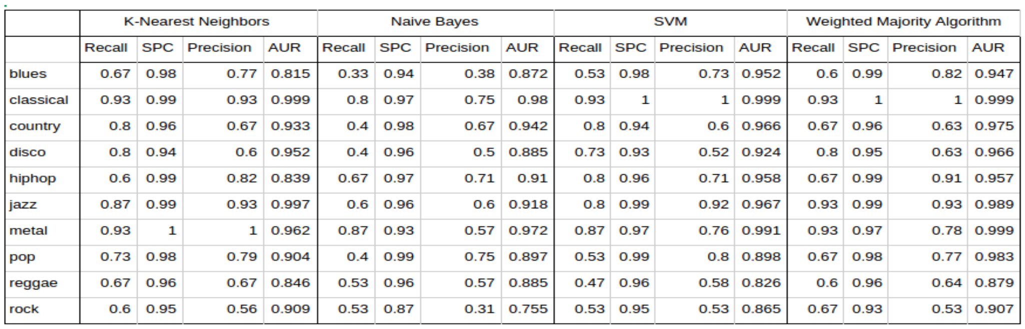


SVM	Naive Bayes	KNN	Validation Accuracy
0	0.04	0.96	0.7266
0	0.05	0.95	0.7333
0.08	0.1	0.82	0.74
0.11	0.07	0.82	0.7466
0.26	0.25	0.49	0.7533
0.27	0.24	0.49	0.76
0.3	0.14	0.56	0.7666
0.35	0.14	0.51	0.7733
0.36	0.13	0.51	0.78
0.46	0.15	0.39	0.7866
0.5	0.07	0.43	0.7933
0.52	0.06	0.42	0.8
0.58	0.09	0.33	0.8066

5 Conclusion

Once we have found the best weights using the validation set, we calculate the model accuracy on test set. Test accuracies for SVM, Naive Bayes, KNN, WMA were 70%, 55.33 %, 76 %, 74.67 % respectively. But once we take a closer look into Test data ROC curves for each labels, we observe that area under the curve for WMA is consistently better than the other models.

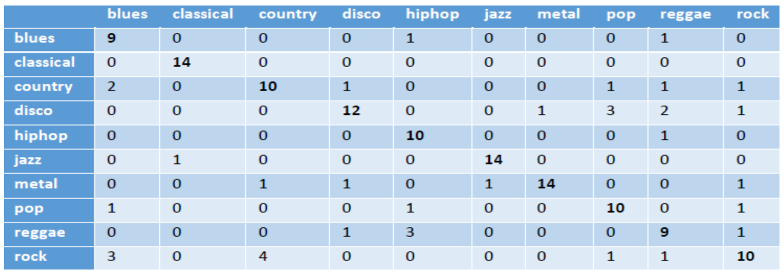
Table 3: Sensitivity, Specificity and Area under ROC curve (AUR)



	K-Nearest Neighbors				Naive Bayes				SVM				Weighted Majority Algorithm			
	Recall	SPC	Precision	AUR	Recall	SPC	Precision	AUR	Recall	SPC	Precision	AUR	Recall	SPC	Precision	AUR
blues	0.67	0.98	0.77	0.815	0.33	0.94	0.38	0.872	0.53	0.98	0.73	0.952	0.6	0.99	0.82	0.947
classical	0.93	0.99	0.93	0.999	0.8	0.97	0.75	0.98	0.93	1	1	0.999	0.93	1	1	0.999
country	0.8	0.96	0.67	0.933	0.4	0.98	0.67	0.942	0.8	0.94	0.6	0.966	0.67	0.96	0.63	0.975
disco	0.8	0.94	0.6	0.952	0.4	0.96	0.5	0.885	0.73	0.93	0.52	0.924	0.8	0.95	0.63	0.966
hiphop	0.6	0.99	0.82	0.839	0.67	0.97	0.71	0.91	0.8	0.96	0.71	0.958	0.67	0.99	0.91	0.957
jazz	0.87	0.99	0.93	0.997	0.6	0.96	0.6	0.918	0.8	0.99	0.92	0.967	0.93	0.99	0.93	0.989
metal	0.93	1	1	0.962	0.87	0.93	0.57	0.972	0.87	0.97	0.76	0.991	0.93	0.97	0.78	0.999
pop	0.73	0.98	0.79	0.904	0.4	0.99	0.75	0.897	0.53	0.99	0.8	0.898	0.67	0.98	0.77	0.983
reggae	0.67	0.96	0.67	0.846	0.53	0.96	0.57	0.885	0.47	0.96	0.58	0.826	0.6	0.96	0.64	0.879
rock	0.6	0.95	0.56	0.909	0.53	0.87	0.31	0.755	0.53	0.95	0.53	0.865	0.67	0.93	0.53	0.907

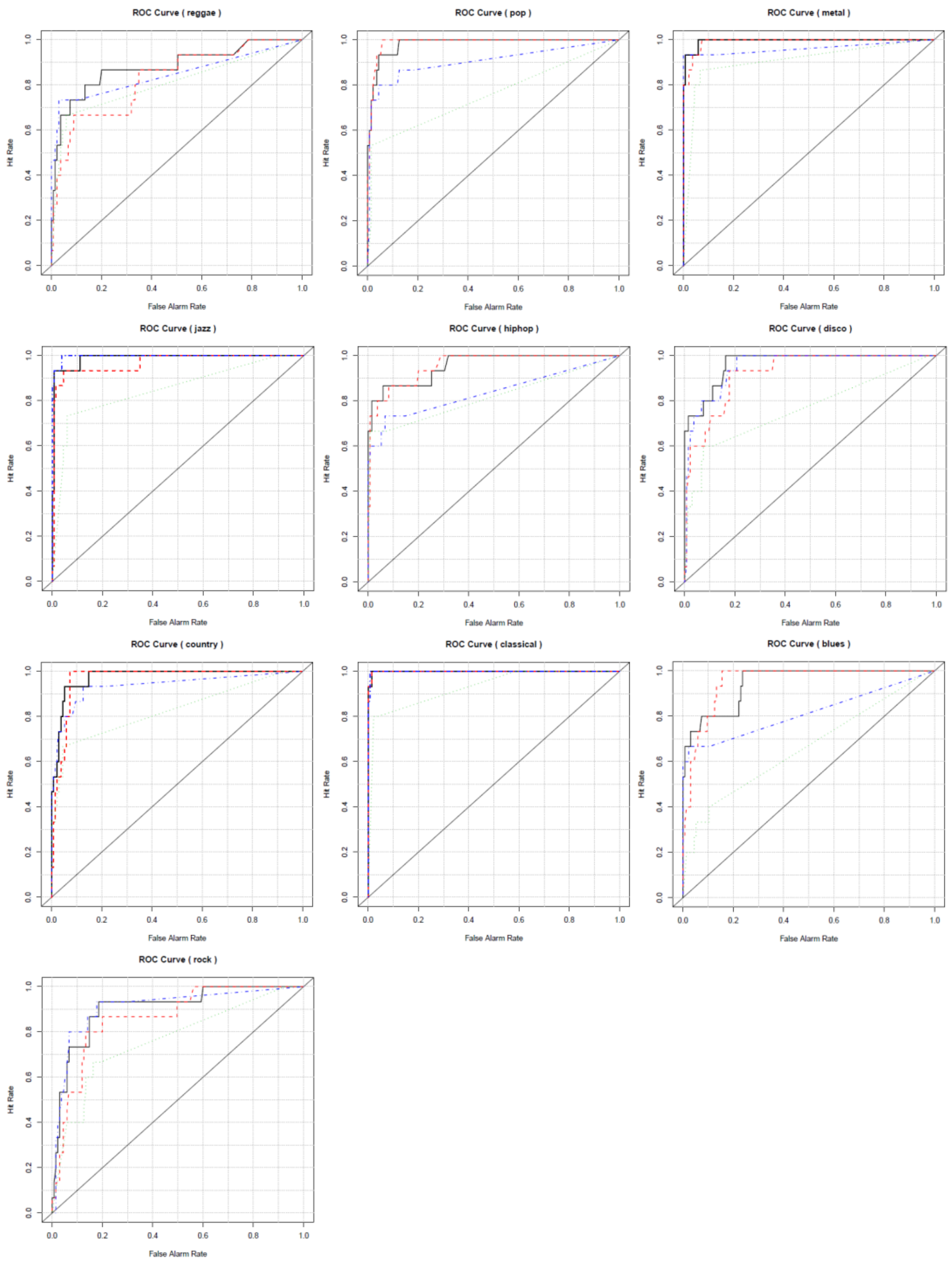
Although overall Test Accuracy for KNN is better than WMA, but story changes when we see the respective recall and precision values. Recall values for KNN is marginally better than WMA, whereas Precision values for WMA is consistently better than KNN.
In conclusion we state that WMA is more robust and a better model. By using WMA, we can improve test accuracy which would have not been possible we using traditional classification techniques.

Figure 4: Confusion matrix WMA



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	**9**	0	0	0	1	0	0	0	1	0
classical	0	**14**	0	0	0	0	0	0	0	0
country	2	0	**10**	1	0	0	0	1	1	1
disco	0	0	0	**12**	0	0	1	3	2	1
hiphop	0	0	0	0	**10**	0	0	0	1	0
jazz	0	1	0	0	0	**14**	0	0	0	0
metal	0	0	1	1	0	1	**14**	0	0	1
pop	1	0	0	0	1	0	0	**10**	0	1
reggae	0	0	0	1	3	0	0	0	**9**	1
rock	3	0	4	0	0	0	0	1	1	**10**

Figure 5: ROC curves for all genres for KNN(Blue), WMA(Black), SVM(Red), Naïve Bayes(Green)



6 Future Work

We have approached this problem by dividing the music into 10 genres, but in reality music does not belong one particular class. There is no fine distinction between genres and sometimes, it is difficult for even humans to clearly separate different genres, for instance rock or metal, pop or hip hop. Accuracy reported here does not capture similarity of different genres.

The work done by us could be extended on various dimensions. The dataset used by us was very small to be able to work properly for the neural network model. The training data could be increased by two methods, (a) collect more data or (b) split the single audio files into multiple files. In our analysis collecting more audio files would be a better approach since reducing the length of audio for training data could reduce the accuracy. Since there might be parts of the songs which may be of different genre than the song as a whole. The feature set used by us is of length 124, this could also be increased. One can calculate these features for fixed intervals of the audio and collate then as new feature set. Other classification algorithms such as decision trees and random forest could also be created and added to the WMA algorithm.

7 Contributions

Table 4: Contributions

Task	**Member**
Data Set Selection	Jivjot, Mangesh, Anuj, Bikram
Feature Set Extraction Library Comparisons	Jivjot, Mangesh, Anuj, Bikram
Naïve Bayes implementation	Mangesh
SVM implementation	Jivjot
Neural Networks implementation	Anuj
K- Nearest Neighbor implementation	Bikram
Weighted Majority Algorithm	Jivjot
Poster	Jivjot, Mangesh, Anuj, Bikram
Report	Jivjot, Mangesh, Anuj, Bikram

8 References

[1] Marsyas. ”Data Sets” http://opihi.cs.uvic.ca/sound/genres.tar.gz

[2]Chroma Featureshttp://labrosa.ee.columbia.edu/matlab/chroma-ansyn/

[3]Spectral Centroid: Grey, J. M., Gordon, J. W., 1978. Perceptual effects of spectral modifications on musical timbres. Journal of the Acoustical Society of America 63 (5), 1493–1500, doi:10.1121/1.381843

[4]Spectral Rolloff: http://sovarr.c4dm.eecs.qmul.ac.uk/wiki/Spectral_Rolloff

[5] Deepa, P.L. and Suresh, K., "An optimized feature set for music genre classification based on Support Vector Machine", Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE

[6] Changsheng Xu, Namunu C. Maddage, and Xi Shao, "Automatic Music Classification and Summarization",IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, VOL. 13, NO. 3, MAY 2005

[7] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM library for support vector machines. ACM Transactions on Intelligent Systems and Technology"
http://www.csie.ntu.edu.tw/~cjlin/libsvm

[8] Hechenbichler K. and Schliep K.P, “(2004) Weighted k-Nearest-Neighbor Techniques and Ordinal Classification, Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich”, (http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps).

[9] David Meyer, “Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien”,
https://cran.r-project.org/web/packages/e1071/e1071.pdf