Teaching machines to classify melodies

Name:	**Abhinav Sethi**
Registration No./Roll No.:	19006
Institute/University Name:	IISER Bhopal
Program/Stream:	DSE
Problem Release date:	January 19, 2022
Date of Submission:	MArch 24, 2022

1 Introduction

Music is an abstract art that uses sound as a means to reflect human emotions in real life. It can help people concentrate better, ease stress at work and school, and improve physical and mental health. For the longest time, music enthusiasts and experts have been trying to understand music and what makes certain kinds of music different from others. In this project, we attempted to discover a feature selection framework that helps identify relevant spectral features and classify new data, using machine learning techniques. We have a collection of ten genres and music data from 1000 audio files (900 training+100 testing), all having a length of 30 seconds. The dataset contains the mean and variance of multiple features extracted from an audio file. Our study focuses on two main feature selection mechanisms and draws a comparison between them across various models.

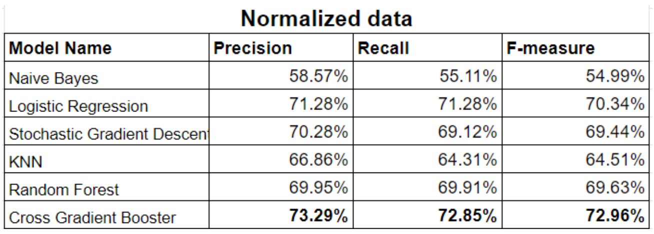
2 Methods

The task at hand is one of multi-class classification. We decided to use 6 Machine Leaning classifiers, namely, *Naive Bayes, Logistic Regression, K-Nearest Neighbours, Random Forest, Stochastic Gradient Descent and Cross Gradient Booster (XGBoost).*

- Firstly we explored the data, and realised that columns 'Unnamed: 0'(a column used for indexing) and 'length'(a column that just tells us that the particular data point(music) is of length 30 seconds) can be removed as they provide no useful information to the models, and may even misguide the models. The dataset has fifty-eight features, however not all of them would provide equally useful information. Thus, we intend to reduce the number of required features to a handful, whilst maintaining a model that gives good results, using the inter-feature correlations(2.1), and Recursive Feature Extraction(2.2).

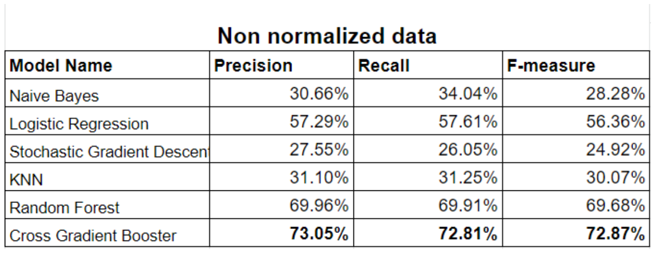
- We set up our baseline models by tuning each model and fitting it to the train data using *k-fold cross validation* and *RandomizedSearchCV*.

- Next, we normalized the data and then tuned each model and fit it to this new normalized data. From the tables it is clear that the models perform significantly better when the data is normalized. Thus, we stick to normalized data for the rest of the experimentations. [All results are macro-averaged.]



Normalized data

Model Name	Precision	Recall	F-measure
Naive Bayes	58.57%	55.11%	54.99%
Logistic Regression	71.28%	71.28%	70.34%
Stochastic Gradient Descen	70.28%	69.12%	69.44%
KNN	66.86%	64.31%	64.51%
Random Forest	69.95%	69.91%	69.63%
Cross Gradient Booster	**73.29%**	**72.85%**	**72.96%**



Non normalized data

Model Name	Precision	Recall	F-measure
Naive Bayes	30.66%	34.04%	28.28%
Logistic Regression	57.29%	57.61%	56.36%
Stochastic Gradient Descen	27.55%	26.05%	24.92%
KNN	31.10%	31.25%	30.07%
Random Forest	69.96%	69.91%	69.68%
Cross Gradient Booster	**73.05%**	**72.81%**	**72.87%**

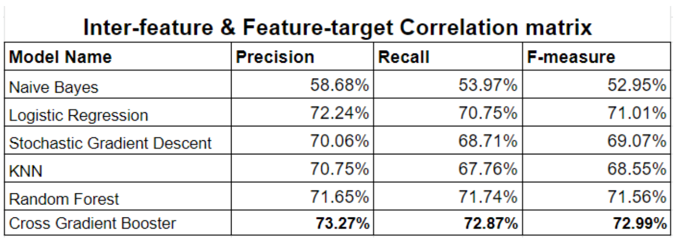
It can be seen that Cross Gradient Booster (XGBoost) gives the best performance on the raw data.

- Next, We tried out two methods for feature selection.

2.1 Inter-feature and feature-target correlation:

- Firstly, we looked at the inter-feature correlation, and checked which features are highly correlated (correlation more than 0.9) with atleast one other feature.
- Next, we looked at the inter-feature correlation between this subset of features and also, of these features with the target labels. Our intuition is that, a higher correlation with the class labels would indicate a similar trend, and thus it makes more sense to preserve that feature. Thus, giving us a way to discard other features.

  We ended up discarding just 3 columns(out of 58 columns) after our manual analysis. The results our provided in the table below.



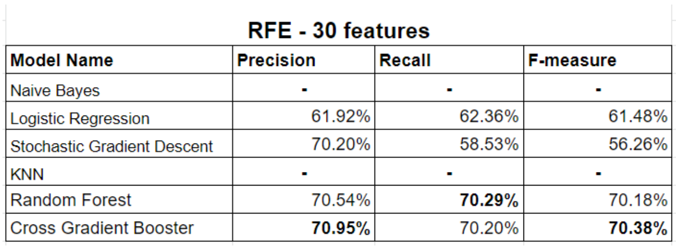
Inter-feature & Feature-target Correlation matrix

Model Name	Precision	Recall	F-measure
Naive Bayes	58.68%	53.97%	52.95%
Logistic Regression	72.24%	70.75%	71.01%
Stochastic Gradient Descent	70.06%	68.71%	69.07%
KNN	70.75%	67.76%	68.55%
Random Forest	71.65%	71.74%	71.56%
Cross Gradient Booster	**73.27%**	**72.87%**	**72.99%**

2.2 Recursive Feature Extraction(RFE

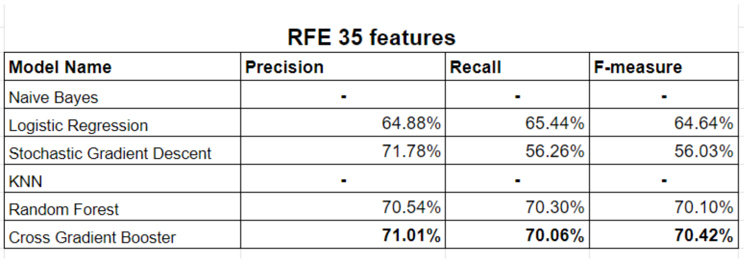
A method that starts with all the features in the dataset and recursively:

- Ranks all the features according to the provided models ranking method (built-in).
- Then, it removes x features at a time, until z are left. Here, x and z are provided by us.(The experimental results are provided below)



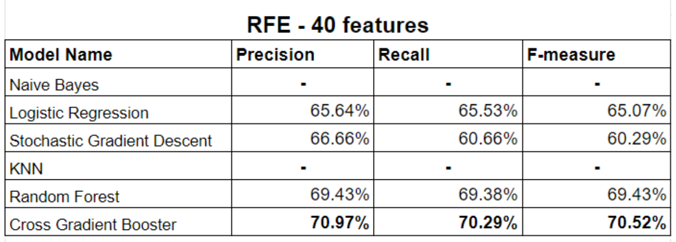
RFE - 30 features

Model Name	Precision	Recall	F-measure
Naive Bayes	-	-	-
Logistic Regression	61.92%	62.36%	61.48%
Stochastic Gradient Descent	70.20%	58.53%	56.26%
KNN	-	-	-
Random Forest	70.54%	**70.29%**	70.18%
Cross Gradient Booster	**70.95%**	70.20%	**70.38%**



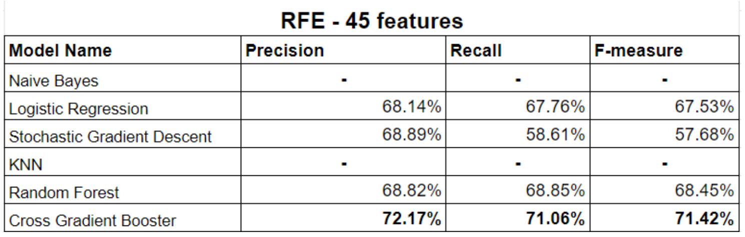
RFE 35 features

Model Name	Precision	Recall	F-measure
Naive Bayes	-	-	-
Logistic Regression	64.88%	65.44%	64.64%
Stochastic Gradient Descent	71.78%	56.26%	56.03%
KNN	-	-	-
Random Forest	70.54%	70.30%	70.10%
Cross Gradient Booster	**71.01%**	**70.06%**	**70.42%**



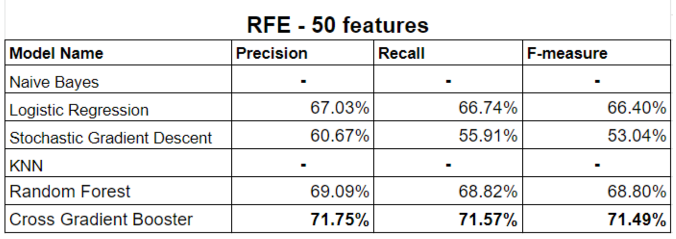
RFE - 40 features

Model Name	Precision	Recall	F-measure
Naive Bayes	-	-	-
Logistic Regression	65.64%	65.53%	65.07%
Stochastic Gradient Descent	66.66%	60.66%	60.29%
KNN	-	-	-
Random Forest	69.43%	69.38%	69.43%
Cross Gradient Booster	**70.97%**	**70.29%**	**70.52%**



RFE - 45 features

Model Name	Precision	Recall	F-measure
Naive Bayes	-	-	-
Logistic Regression	68.14%	67.76%	67.53%
Stochastic Gradient Descent	68.89%	58.61%	57.68%
KNN	-	-	-
Random Forest	68.82%	68.85%	68.45%
Cross Gradient Booster	**72.17%**	**71.06%**	**71.42%**



RFE - 50 features

Model Name	Precision	Recall	F-measure
Naive Bayes	-	-	-
Logistic Regression	67.03%	66.74%	66.40%
Stochastic Gradient Descent	60.67%	55.91%	53.04%
KNN	-	-	-
Random Forest	69.09%	68.82%	68.80%
Cross Gradient Booster	**71.75%**	**71.57%**	**71.49%**

Note: RFE is not possible for KNN and Naive Bayes, as they don't have an inbuilt feature selection metric.

From our experiments, it is seen that even after removing 30 features (more than half the features) using RFE with Cross Gradient Booster as the feature selecting model, there is only a 2.5% loss in f-measure. A very impressive result in my opinion. It shows how good the inbuilt feature-ranking criteria of XGBoost is.

From these 2 feature selection methods, and all our experiments it seems best to go with the manual inter-feature and feature-target correlation method, and thus we train that tuned XGBoost model on the entire train data and use it to predict on the test dataset.

3 Discussions

- The task of music genre classification, is one which is very hard to do, and thus it is even more important for us to be able to find a way to teach machines to look in between the data and figure out non-explicit information and tie these bits of information to a scheme through which the can identify the genre correctly. Our method does just that; it is an attempt to figure out some new information from the information provided.
- We could not tune the SVM models and from our experiments with SVM(not presented here as we didn't have results for every feature selection method) seemed to perform very well on the data (even better than XGBoost) and so, we would like to use better systems and then use slightly more complex models.
- Having used RandomizedSearchCV, which picks features at random and tries to give a good enough model, whilst keeping the complexity less was a good step we took as we realised it would take too long to run gridssearch for everything. But, this also means that there is scope of improvement in the tuning phase as well.
- In future, more techniques could be explored which efficiently find the hyperparameters and thus, help in reducing the running time of the code.
- Also, we can try to see how to select features for KNN and Naive Bayes, so as to be able to use them with RFE, and maybe those results could be interesting as well.
- Our similar process can be used for any classification task, and thus, we feel, we have contributed a good workflow for use by other aspiring data scientists or ML engineers.