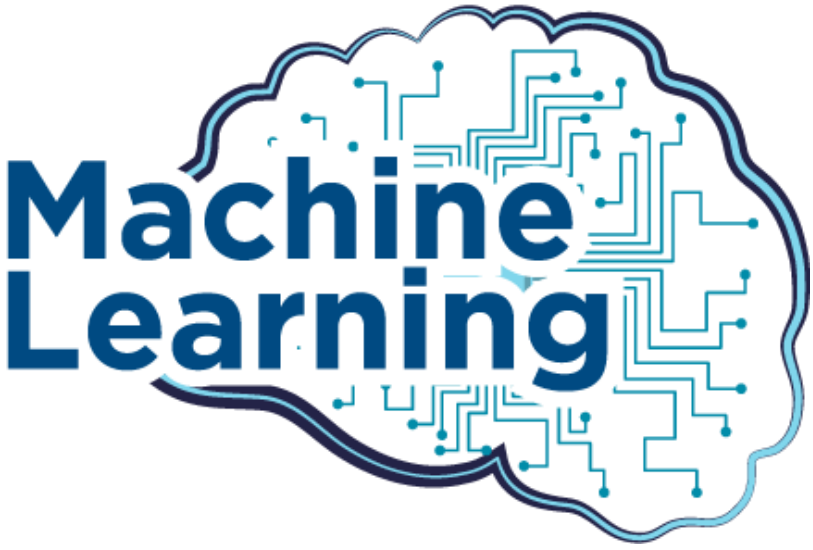


Music Genre Prediction Using Machine Learning Algorithms

Machine Learning Project Report
April 15th, 2022

TABLE OF CONTENT

INTRODUCTION

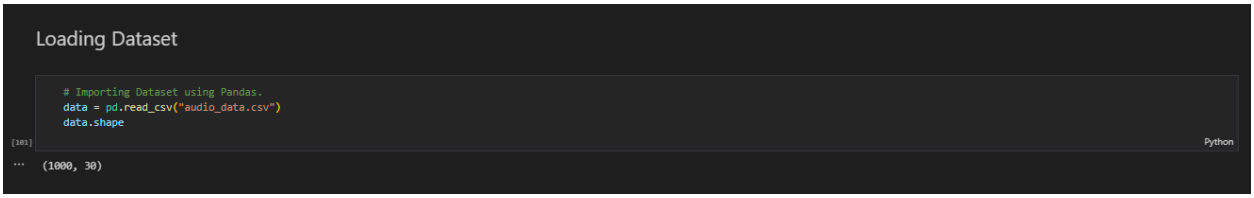
A crucial problem in music information retrieval is music genre classification, which tries to automatically categorize songs into their respective genre categories. Many real-world uses, including music recommendation systems, music search engines, and custom playlists, depend on the ability to correctly forecast the genre of a particular musical composition. Using the GTZAN dataset, which is a frequently used dataset in music genre classification study, we want to predict the musical genre in this project. To create models that can reliably categorize music genres using audio features taken from the GTZAN dataset, we will apply a variety of machine learning techniques. This report's goal is to share our research and analysis on the effectiveness of the models created for this project.

This project is a multi-class classification problem where our target variable “class name” is having about 10 classes which are names of different genres. For the accurate prediction of our target variable, we are using different classifiers including decision tree, random forest, naïve bayes classifier, logistic regression, boosting models etc.

Our goal is to build a robust and accurate prediction model that can help the music industry to identify genres of music. So, for that reason, we are using different optimization techniques such as K fold cross validation to improve accuracy of the model and to validate our models and ensure they are not overfitting the data and Grid search CV to get best hyperparameters in this project. To get the best performing features we select features with random forest feature selector.

DATASET

The dataset we are using in this case study is from Kaggle named as “GTZAN data for music genre prediction” and contains feature information of music. The database contains 1000 patients’ records and 30 features. We are using “Class label” as our target variable. It is a multi-class classification problem, where value of classes are different Genres. Dataset is balanced and consists of 30 attributes other than target attribute.



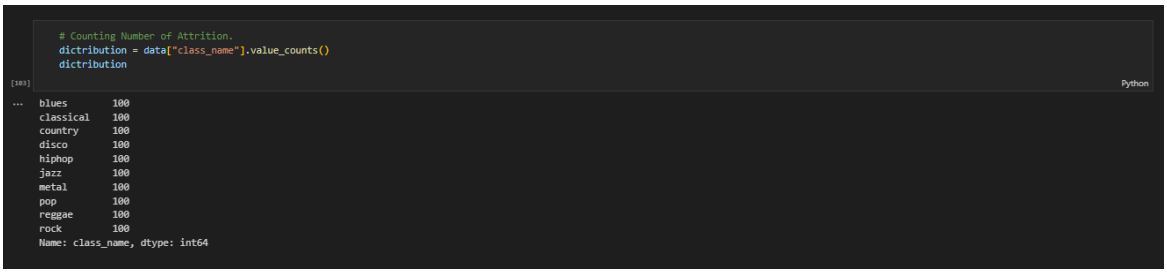
DATA PREPROCESSING

Before using the data in machine learning algorithms, preparing the data is a crucial step. Data preprocessing is a data mining technique used to turn raw data into a format that is both practical and effective.

For our dataset, we are also performing data preprocessing. The steps we performed for preprocessing of data are below:

1- Distribution of Class Label

The first step we are performing in data preprocessing is checking the distribution of class label. We are checking whether the classes are balanced or imbalanced, but all the classes are balance and each genre contains 100 records.

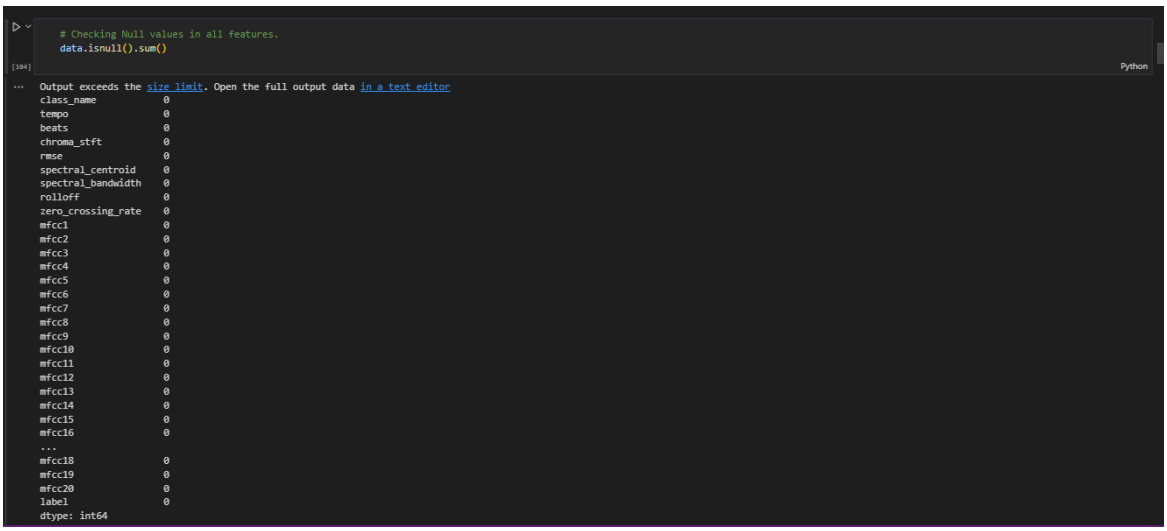


```
# Counting Number of Attrition.
dictribution = data["class_name"].value_counts()
dictribution
```

```
[103]                                                                    Python

blues       100
classical   100
country     100
disco       100
hiphop      100
jazz        100
metal       100
pop         100
reggae      100
rock        100
Name: class_name, dtype: int64
```

2- Checking Null Values

After checking the distribution of the class label, we are now checking the null values in our dataset.

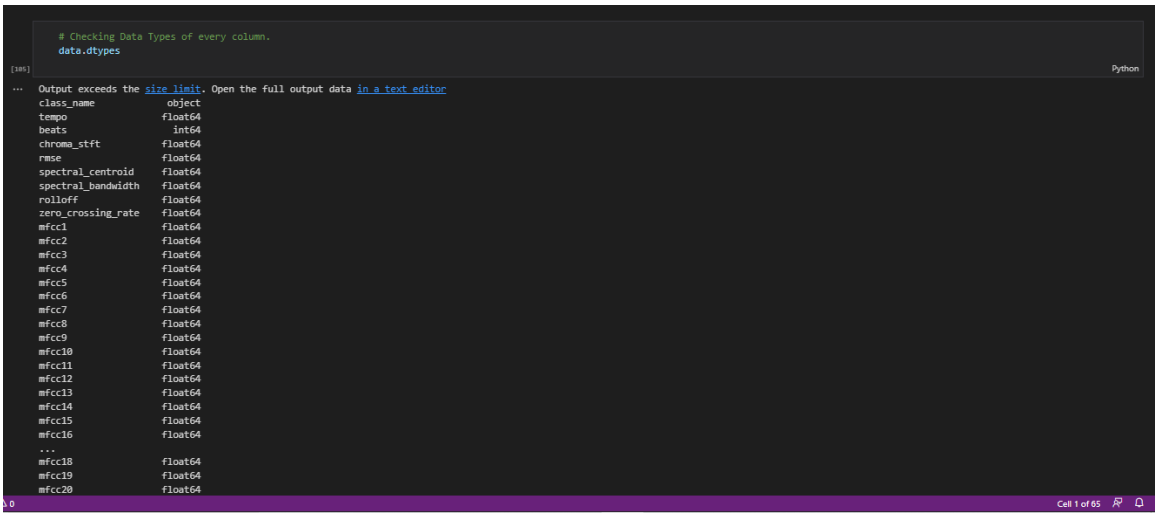


```
# Checking Null values in all features.
data.isnull().sum()
```

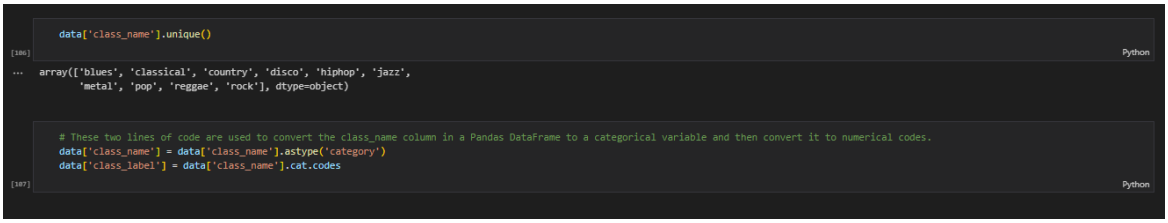
```
[104]                                                                    Python

Output exceeds the size limit. Open the full output data in a text editor
class_name            0
tempo                 0
beats                 0
chroma_stft           0
rmse                  0
spectral_centroid     0
spectral_bandwidth    0
rolloff               0
zero_crossing_rate    0
mfcc1                 0
mfcc2                 0
mfcc3                 0
mfcc4                 0
mfcc5                 0
mfcc6                 0
mfcc7                 0
mfcc8                 0
mfcc9                 0
mfcc10                0
mfcc11                0
mfcc12                0
mfcc13                0
mfcc14                0
mfcc15                0
mfcc16                0
...
mfcc18                0
mfcc19                0
mfcc20                0
label                 0
dtype: int64
```

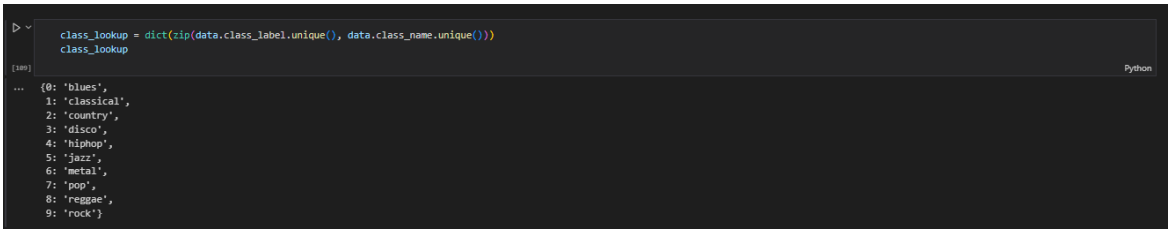
3- Checking Data Types of Features

So, after checking the null values our next step is to check the data types of the features. We have to check whether our features are in numerical format or in string format if any of our feature is in a string format then we have to convert it into a numerical format from the screenshot below we can see that one of our class labels is in string format so now we have to convert it.



Converting label column from string to integer format.



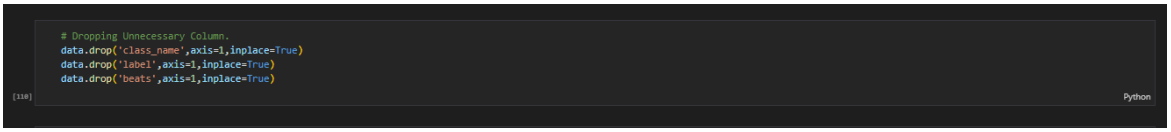


Now we have all our features in the numerical format.

4- Drop Unnecessary Columns

Before using machine learning on a dataset, it is frequently a good idea to remove pointless columns. This can aid in reducing the dataset's dimensionality, making it simpler to deal with and enhancing the effectiveness of the machine learning algorithms.

Columns that contain redundant or irrelevant data, have a large percentage of missing values, or have little to no impact on the target variable are examples of unnecessary columns. We may concentrate on the most crucial features and increase the precision of our machine learning models by deleting these columns.



TRAIN TEST SPLIT

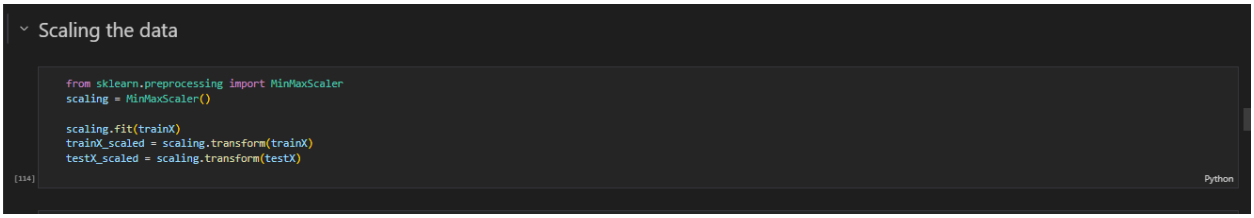
The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. Here we are splitting our dataset into train dataset and test dataset with the ratio of 70-30% where 70% is our train dataset and 30% is our test dataset.

SCALING DATASET

The process of normalizing the input features of a dataset so that they have a comparable scale or range is known as feature scaling, and it is a crucial preprocessing step in machine learning. This

prevents features with bigger magnitudes from dominating the learning process and makes sure that each feature contributes equally to the model's final output.

We are using Min Max scaler for the feature scaling. Minmax scaler is a type of feature scaling technique in machine learning that scales the features to a fixed range - usually between 0 and 1.



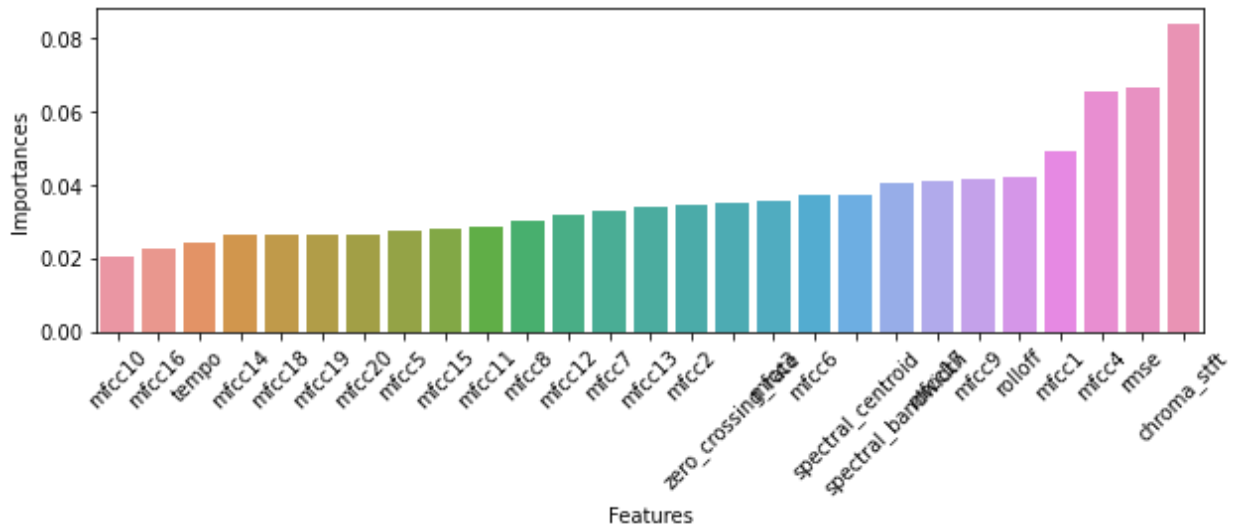
Grid Search Hyper Parameter Tuning

To determine the ideal set of hyperparameters for a particular model, we implemented the grid search hyperparameter tuning technique. Hyperparameters are model parameters that are selected by the user prior to training rather than being learned from the data. In our project, we are using Grid search technique to get the best possible hyper parameters for the model to increase its efficiency.

Feature Selection Using Random Forest

Feature selection using random forest involves using a random forest model to identify the most impactful features in a data. Random forest is an ensemble learning algorithm that works by forming multiple decision trees and combining their outputs predictions. In feature selection using random forest, the model determines the importance of each feature by calculating the decrease in impurity (e.g., Gini index or entropy) that is achieved when that feature is used to split the data at a node in the decision tree. The random forest algorithm builds the decision trees and then determines the average decrease in impurity for each feature across all the trees. Characteristics

are more significant if their average impurity decrease is higher. A subset of the most crucial traits might be chosen for additional study based on this rating. Because of the decreased overfitting and enhanced interpretability, this can improve model performance.



Out of all the features, we are selecting 12 most impactful features and then implementing machine learning models on the selected features.

```
    model_tree = RandomForestClassifier(n_estimators=100,random_state=42)

    # use RFE to eleminate the less importance features
    sel_rfe_tree = RFE(estimator=model_tree, n_features_to_select=12, step=1)    # Selecting 9 imporatant Features.
    X_train_rfe_tree = sel_rfe_tree.fit_transform(trainX, trainy)
    print(sel_rfe_tree.get_support())
    print(sel_rfe_tree.ranking_)
```

[129] Python

```
[False  True  True  True  True False  True False  True  True False
  True False False  True False False  True False False False False  True
 False False False]
[13  1  1  1  1  1  3  1  5  1  1 11  1  6  7  1 14  2  1  4 15 10 16  1
  8 12  9]
```

Model Implementation: -

First, we are implementing different machine learning classifiers on our dataset first without any feature selection and then after doing selection with methods of random forest feature selection. We have 27 attributes at first and 12 after selection, and our data is 70-30% divided into train and test sets.

1- Random Forest Classifier

The second model we are implementing after decision tree is random forest classifier. A Random Forest Classifier is a supervised machine learning algorithm that uses an ensemble of decision trees to perform classification tasks. The random forest algorithm is an extension of the decision tree algorithm, which overcomes some of the limitations of decision trees such as overfitting and instability.

On our dataset, random forest is performing well as compared to other models, giving accuracy of 62% accuracy on full feature dataset.

2- Gradient Boosting Classifier

One of the famous boosting algorithms, which combines different weak learners into strong by minimizing the loss function. It is an ensemble technique. The goal of gradient boosting classifier is to iteratively add new models to the ensemble that correct the errors of the previous models.

In our project Gradient boosting is not performing much well on both full feature and selected feature dataset. It is getting the highest accuracy of 57% only.

3- XGBoost Classifier

A well-known machine learning technique that is a member of the gradient boosting family is called XGBoost (Extreme Gradient Boosting) Classifier. It is a development of the conventional gradient boosting approach, which minimizes a loss function to combine several weak models into a strong one.

High performance and accuracy are hallmarks of XGBoost, particularly when applied to structured data issues like feature engineering and tabular data. In comparison to other

gradient boosting methods, XGBoost key benefit is its scalability, which enables it to handle huge datasets with millions of rows and columns.

In our project Xtreme Gradient boosting is performing better than all the other models, maybe because of the reason that XGBoost can capture non-linear relationships between the features and the target variable. It gets the highest accuracy of 64% before feature selection but its accuracy decreases after the feature selection.

Voting Classifier (Ensemble Model)

Voting classifier is basically an ensemble technique which unites different base classifiers and based on the result of these classifiers it computes new result. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. We are using a soft voting technique here which computes probability based on probabilities and weights associated with different classifiers.

In voting classifier, we are using our best models. We are using 3 models which are random forest models, gradient boosting model and XGBOOST which were giving us best results. So, after combining all our best performing models we create a voting classifier model, which gives us the goof results but not better than random forest model which has better accuracy.

Saving Best Model

Pickle is a Python library used to serialize and deserialize Python objects. It can be used to save the state of a Python object to a file, and then later load that state back into memory. We are saving our random forest model using pickle, so we can later deploy the model on the streamlet web app for genre prediction.

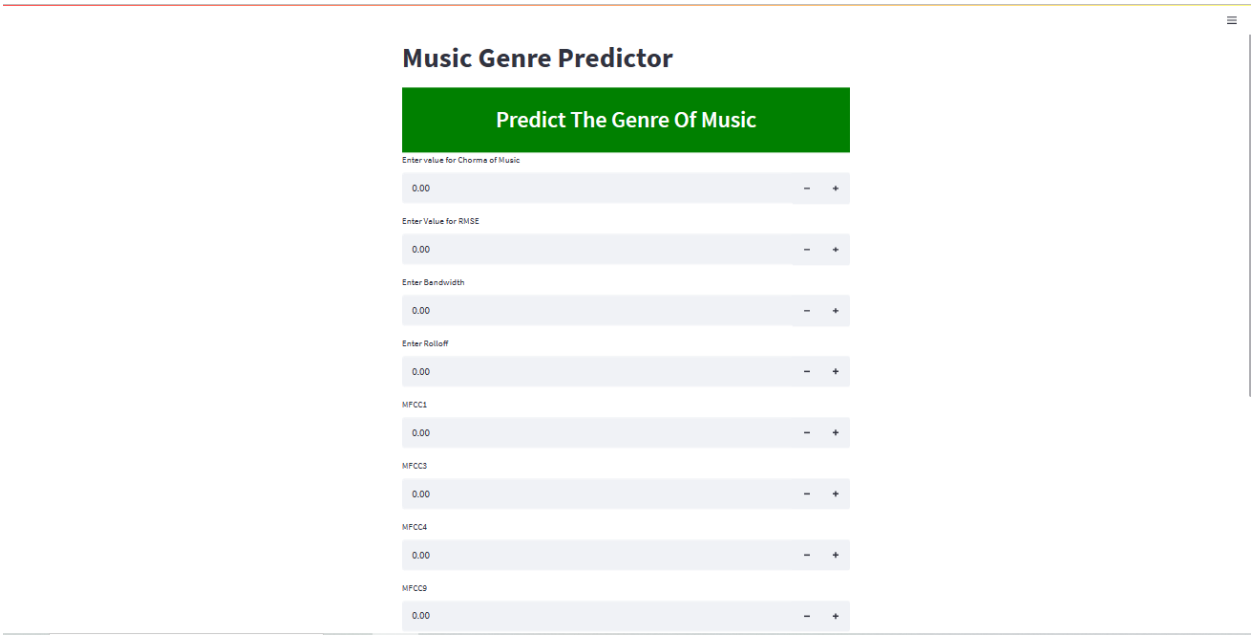
Saving Random Forest Model

```
import pickle
pickle_out = open("rf.pkl","wb")
pickle.dump(rf, pickle_out)
pickle_out.close()
```

[142] Python

STREAMLIT WEB APPLICATION

After implementing different machine learning models, we finally got our best performing model which is a voting classifier so now we are saving our model with the help of pickle and deploying our model on a web app created on streamlet. This web app will take different inputs which are features of our data set and predict the genre of music based on the provided information.



CONCLUSION

In conclusion, our project successfully explored the challenge of predicting the genre of music using machine learning algorithms. We implemented different classification models such as Random Forest, XGBoost, and Gradient Boosting, along with a Voting Classifier model to achieve the best results. For the optimization of results, we performed feature selection using Random Forest and fine-tuned our models through Hyperparameter tuning using Grid Search. Additionally, we created a stream lit web application that allows users to predict the genre of music by providing feature information. Through our project, we demonstrated the effectiveness of ensemble methods in predicting the genre of music and highlighted the importance of feature selection and hyperparameter tuning. Overall, our project showcases the potential of machine learning in the field of music genre prediction and opens opportunities for further research and development in this area.

LIMITATION AND DISCUSSION

Due to the high number of classes, base models are not getting much high accuracy.

The available data for training was quite low. More experiments and better learning can be done if larger data is available.

We also implemented other base model such as decision tree, logistic regression, naïve bayes but their performance was very poor.