

A PROJECT REPORT

on

“Comparison of Feature Selection Pipelines for Music Genre Classification”

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

**BACHELOR’S DEGREE IN
INFORMATION TECHNOLOGY**

BY

PARTHIB GOSWAMI	2006031
SUBHAM MAJI	2006047
ANUKRITI JAIN	2006532
ISHA KESHAV	2006555

UNDER THE GUIDANCE OF

DR. AJAY ANAND



SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

MAY 2024

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024



CERTIFICATE

This is certify that the project entitled

“Comparison of Feature Selection Pipelines for Music Genre Classification”

submitted by

PARTHIB GOSWAMI	2006031
SUBHAM MAJI	2006047
ANUKRITI JAIN	2006532
ISHA KESHAV	2006555

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2022-2023, under our guidance.

Date: / /

(Guide Name)
DR. AJAY ANAND

Acknowledgements

We are profoundly grateful to **DR. AJAY ANAND Sir** of **Assistant Professor** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

PARTHIB GOSWAMI

SUBHAM MAJI

ANUKRITI JAIN

ISHA KESHAV

ABSTRACT

Numerous music listeners make playlists depending on the kidney they're harkening to, leaving out possible uses like playlist operation and recommendation. Music kidney Bracket classifies different musical stripes. We develop a Machine Learning design that classifies different musical stripes from audio lines. We classify these audio lines using their low position frequency features and time sphere. On these tracks, we employ colorful machine literacy styles by applying point Selection and colorful classifiers. For this we need a dataset of audio tracks having analogous size and frequency range. We use the GTZAN kidney bracket dataset, which is the most recommended data set for the music kidney bracket design and was collected for this same cause.

Keywords: DECISION TREE, FEATURE SELECTION, RANDOM FOREST CLASSIFIER, K-BEST CLASSIFIER, GRID SEARCH CLASSIFIER, VARIANCE THRESHOLD

Contents

1	Introduction	7
2	Basic Concepts/ Literature Review	8
	2.1 BASIC CONCEPTS	8
	2.1.1 EDA	8
	2.1.2 DECISION TREE	9
	2.1.3 FEATURE SELECTION	9
	2.1.4 RANDOM FOREST CLASSIFIER	9
	2.1.5 K BEST CLASSIFIER	9
	2.1.6 GRID SEARCH CLASSIFIER	10
	2.1.7 VARIANCE THRESHOLD	10
	2.2 LITERATURE REVIEW	10
3	Problem Statement / Requirement Specifications	12
	3.1 Project Planning.....	12
	3.2 Project Analysis	12
	3.2.1 Libraries used.....	3
	3.3.2 Functions and Classes used.....	3
	3.3 Activity Diagram	
4	Implementation	4
	4.1 Methodology / Proposal	4
	4.2 Testing / Verification Plan	4
	4.3 Result Analysis / Screenshots	4
	4.4 Quality Assurance	4
5	Standard Adopted	5
	5.1 Design Standards	5
	5.2 Coding Standards	5
	5.3 Testing Standards	5
6	Conclusion and Future Scope	6
	6.1 Conclusion	6
	6.2 Future Scope	6
	References	7
	Individual Contribution	8
	Plagiarism Report	9

List of Figures

1.1 IMAGE CAPTION	2
4.1 IMAGE CAPTION	9

Chapter 1

Introduction

The classification of musical genres is important for music lovers who create playlists based on specific genres. Machine learning techniques have been used in previous studies to classify music genres, but there is still room for improvement in classification accuracy. The goal of this project is to develop a genre classifier that accurately categorizes music tracks based on their characteristics.

To achieve this goal, we applied various machine learning approaches to a dataset of audio tracks with similar size and frequency range. The GTZAN genre classification dataset was used, which consists of 1000 audio tracks lasting 30 seconds each, with 10 different music genres including Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae and Rock. Features such as Zero Crossing Rate, Spectral Centroid, Spectral Bandwidth, Spectral Contrast, Spectral Flatness, Spectral Rolloff, RMS, MFCC, MFCC Derivative, Chroma, STFT, Chroma CENS, Chroma CQT, Tonnetz, Polynomial and Wavelet transform were used to define the genres. We perform Exploratory Data Analysis first. We tried plotting all the features to get insights but since they were weak, we applied Decision Tree to get concurrent features.

The data was analyzed using Exploratory Data Analysis and Decision Tree to identify concurrent features, and then Random Forest, K-Best and Variance Threshold with Grid Search were applied to classify the genres. The classification accuracy of each model was assessed and suggestions for future work were made based on the results.

Chapter 2

Basic Concepts/ Literature Review

2.1 BASIC CONCEPTS

2.1.1 EXPLORATORY DATA ANALYSIS

Examining and summarizing a dataset in order to gain knowledge, spot trends, and test hypotheses is known as exploratory data analysis (EDA). It entails applying a variety of statistical and graphical methods to comprehend the data, spot probable outliers or anomalies, and see how variables are related to one another. EDA is a crucial phase of any data analysis project since it aids in understanding the distribution and structure of the data, which informs the choice of suitable statistical methods and models for further studies. The distribution of data can be visualized using histograms, scatter plots, box plots, and density plots, and summary statistics like mean, median, mode, standard deviation, and correlation coefficients can be calculated to quantify associations between variables. EDA seeks to comprehend the data more thoroughly and to spot any problems or restrictions that might affect the reliability of later analysis.

2.1.2 DECISION TREE

Both Classification and Regression problems make up the majority of its application. It resembles a tree and reflects a series of judgements or guidelines that may be applied to categorise or forecast outcomes depending on input data. The tree nodes, which stand in for the decision-making points, and branches, which stand in for the potential outcomes or directions that could be chosen. The decision tree algorithm divides the data into groups that are as homogeneous as feasible with regard to the target variable using a recursive partitioning method. By following the branches from the root to the leaf nodes, which stand in for the projected outcomes, the resulting tree can be utilized for prediction. By following the branches from the root node to the leaf nodes, which stand in for the expected class or value, the resulting tree can be utilised for prediction. Because they are simple to understand and can handle both numerical and categorical data, decision trees are frequently used in data mining and predictive analytics

2.1.3 FEATURE SELECTION

To improve the accuracy and efficiency of machine learning models, it is essential to select only the most relevant features or variables from a larger set of available features. This process of selecting relevant features is called feature selection. It's crucial for enhancing model performance, lowering overfitting, and requiring the least amount of training time and computational resources. Filter methods, wrapper methods, and embedding methods are the three categories into which feature selection approaches can be separated. Wrapper approaches employ a model's performance to assess subsets of features, filter methods use statistical measurements or correlation to assess the significance of features, and embedded methods include feature selection into the model training process. The individual problem and data set being examined determine the feature selection strategy to be used.

2.1.4 RANDOM FOREST CLASSIFIER

Multiple decision trees are built using bootstrapped samples of the training data, and each tree's features are randomly chosen. The algorithm analyzes each split using a subset of features during training to determine which split maximizes information gain. The algorithm aggregates all of the trees' output during prediction to get a conclusion. The feature importance metrics provided by the random forest algorithm are well renowned for their ability to handle noisy data, prevent overfitting, and manage a variety of inputs. It is frequently used in a variety of industries, including marketing, banking, and healthcare.

2.1.5 K BEST CLASSIFIER

The top K most useful features from a given dataset are chosen using the feature selection method known as K-best classifier in machine learning. The objective is to make the feature space less dimensional, which can improve model performance and lessen overfitting. The top K features are then chosen for use in the model by the K-best classifier, which ranks the features according to a statistical metric such the chi-squared test or mutual information. The specific challenge at hand and the amount of the dataset both influence the choice of K. To increase model efficiency and accuracy, the K-best classifier is frequently used in conjunction with other feature selection strategies as wrapper methods or embedding methods.

2.1.6 GRID SEARCH CLASSIFIER

To determine the ideal set of hyperparameters for a particular model, machine learning researchers use the grid search classifier. In order to evaluate the model's performance, it builds a grid of hyperparameter values to search through. It then compares the model's performance for each set of hyperparameters. The grid is built by specifying a range of values for each hyperparameter, and the programme then uses cross-validation to assess the model's performance. Grid search is a potent method for identifying the optimum hyperparameters to optimize model performance, but computationally expensive. In machine learning applications such as image identification, natural language processing, and finance, grid search is frequently employed.

2.1.7 VARIANCE THRESHOLD

Variance threshold is a feature selection technique used to filter out low-variance features before applying a machine learning algorithm. Each feature's variance is calculated, and any features with variances below a predetermined threshold are removed. To improve model performance, variance threshold is frequently combined with other feature selection techniques like grid search. Variance threshold can be used as a preprocessing step before using grid search to lower the amount of features and boost the effectiveness of the hyperparameter tuning procedure. Additionally, it can assist in lowering overfitting and enhancing model accuracy by removing low-variance features. The problem domain and the distribution of the feature values are often taken into consideration when selecting the threshold value.

2.2 LITERATURE REVIEW

[1] In this study, it outlines the efforts they made to develop classification techniques that would enable them to recognise a certain genre from audio features. To train the dataset, SVM and softmax regression with optimized parameters is used. These two models are used as the study's baseline. The use of additional strategies like neural networks and K-nearest neighbors is then tested. Each model's classification accuracy is assessed for errors, and subsequent work is suggested based on the results. The music genre categorization system is constructed using the free dataset Free Music Archive (FMA), and other system calibrations including error analysis are implemented. It has 106,574 musical songs that are organized into 16 genres using a hierarchical taxonomy. There

are 9 audio features and 518 attributes total in each track. These characteristics are generated by using the Python tool LibROSA to preprocess the data of FMA music recordings.

[2] In this essay, they concentrated on tags related to western music genres. Modern auto-tagging models do not recognise music with tags that are outside the top-50, such as "classical music" in the Million Song Dataset1 (MSD) and "blues" in the MagnaTagATune Dataset2 (MTT), because they concentrate on the top-50 tags of a dataset. They suggest a transfer learning method to address this. The target task of classifying music genres can be accomplished using the knowledge gathered from the pre-trained music auto-tagging models. Music auto-tagging served as the source task for pre-trained models. They can attain excellent performance metrics and successfully adapt to the intended task. They observe that Pop and R&B are easily misclassified. Pop is a broad genre that describes all music that is popular. Thus it is difficult to separate Pop from other genres. Likewise, R&B encompasses elements from Pop, Rap, Electronic Music and so on. It is possible to be misclassified into these genres. We assume that using a thresholding method instead of majority voting could alleviate the song-level misclassification problems.

[3] The reliability of traditional algorithms is continually put to the test by the increasing complexity of genres and the frequent blending of several genres. As a result, the necessity for a more sophisticated genre classification system has arisen. In this paper, a brand-new active transfer music genre categorization method (ATMGCM) is proposed. The ATMGCM algorithm provides superior accuracy in large databases or with noises when compared to SVM and Random Forest.

The simulation findings also show that the ATMGCM has a marked decline in niche genres. As a result, active learning is used to establish useful labels for a small portion of samples from unlabeled datasets, and transfer learning is used to transfer information from the source dataset to the target dataset. This paper demonstrates through trials that Genres can be effectively classified using the ATMGCM algorithm. Additionally, this method can still significantly improve performance even if only 10% to 15% of the data is labeled. The RNN model (LSTM) will be used in future research to enhance the performance of classification findings.

[4] They put out a deep learning transfer learning strategy and tested it on six audio and music information retrieval tasks. In order to handle the problems of genre classification, vocal/non-vocal classification, emotion prediction, speech/music classification, and acoustic event classification,

the pre-trained convnet was first taught to predict music tags. Contrary to the typical transfer learning strategy, they suggested using the features from each convolutional layer after applying an average-pooling to minimize the size of each feature map. The pre-trained convnet feature performed well overall in the testing. It surpassed the baseline MFCC feature, which is a widely common feature in music information retrieval tasks since it provides reasonable baseline performance in many tasks, for all six of the tests.

[5]The effectiveness of listening to music has been documented in the literature, but it is still unclear what elements and what kinds of music have therapeutic effects or how music therapists should choose music. Here, we offer a study that uses machine learning techniques to identify the key predictors of the relaxing benefits of listening to music. 320 participants had their degrees of relaxation gauged using a visual analogue scale (VAS) before and after the listening session. The individuals were subsequently split into three groups based on whether their level of relaxation had increased, decreased, or remained constant. To forecast how listening to music will affect relaxation, a decision tree was created. A decision tree was created with an overall accuracy of 0.79. The decision tree's structure was analyzed to draw some conclusions about the most crucial variables that predict the impact of listening to music, including beginning levels of relaxation, the mix of education and musical training, age, and frequency of listening. Finding predictive elements that affect the results of therapeutic music listening is made feasible by the decision tree that results from the examination of this interpretable model. Because therapeutic music listening is so subjective, using machine learning techniques to support music therapy practice is a crucial and cutting-edge idea.

Chapter 3

Problem Statement/Requirement specification

3.1 Project Planning

The flow of project is as follows,

- 1) Exploratory Data Analysis : Plotting of scatter plots for all the 10 genres and 57 features of the data and then from the plots the taking out the insights to find the weak and strong features that will be used later in the models.
- 2) Feature selection : Use of four feature selection models that are Decision Tree, SelectKBest, Principle component analysis and Variance threshold with grid to find features in the set of 5, 10 and 15 each.
- 3) Accuracy calculation : All the features in the set of 5, 10 and 15 will be trained in the Random Forest model to find the set with highest accuracy.
- 4) Prediction model : The set of feature with the most accuracy will be used in the prediction model to predict genre.

3.2 Project Analysis

3.2.1 Libraries

1) Pandas

- It's a software package and set of tools for analysing and manipulating data.
- Thanks to vectorization, many built-in methods for fast data manipulation are now available.
- It provides DataFrame object for working with multivariate data using built-in indices.
- Series object for working with univariate data using built-in indices.
- In-memory data structures and data reading and writing tools for various file formats.

- Data alignment and integrated missing data processing.
- Inserting and removing columns from data structures.
- Split-apply-combine actions on data sets are supported by the engine group.
- Merge and combine records.

2) Numpy

- NumPy is a Python library that adds support for large multidimensional arrays and matrices, and a large number of advanced mathematical functions for manipulating these arrays.
- provides functional parity with MATLAB and allows users to write fast programs as long as most operations are performed on arrays or matrices rather than scalars.
- It partially addresses the slowdown problem by providing multidimensional arrays and methods and operators that efficiently manipulate arrays.
- Multichannel images are simply stored as 3D arrays, which can be indexed, sliced, or masked with other arrays to access specific pixels in the image.

3) Matplotlib

- Matplotlib is a Python package that can be used to create static, animated, and interactive visualizations.
- A map library that supports arbitrary points, lines, polygons, image transformations, and object-oriented map projection definitions.
- Just pass the desired backend as an argument while storing the number using the backend keyword.
- matplotlib.pyplot is a state-based Matplotlib interface that provides an implicit MATLAB-like plotting method.
- also acts as a character GUI manager, opening characters on the screen. pyplot is primarily designed for interactive charting and basic programmatic plot generation.

4) Sklearn

- scikit-learn is an open-source Python library that implements a set of machine learning, preprocessing, cross-validation, and visualization algorithms through a unified interface.
- A simple and efficient tool for data mining and data analysis.
- It has different algorithms such as random forests, support vector machines, gradient boosting, k-means etc.
- Easily accessible and can be reused in different contexts.

- Commercially Used.

5) Pydot

- Pydot is a Python module that provides a Python interface to the Graphviz visualization software. Graphviz is his package of powerful open source graph visualization software that allows users to create and visualize graphs and networks in various formats such as SVG, PDF, PNG, etc.
- Pydot provides a simple, easy-to-use interface to the Graphviz software. Pydot allows users to create and manipulate graphs and networks in Python code and visualize those graphs with Graphviz.
- Pydot supports a wide variety of features such as node and edge styling, color and font customization, layout customization, and support for multiple output formats.
- Pydot is widely used in various fields such as network analysis, data science, and machine learning. It is also commonly used in visualization applications such as web development and scientific publishing.

6) Librosa

- Librosa is a Python package for music and audio analysis. It offers various audio data analysis and processing functions such as time-frequency analysis, feature extraction, and visualization.
- Librosa is built on several other scientific computing packages in Python such as NumPy, SciPy and matplotlib.
- Librosa allows users to extract useful features from audio data, such as Spectral features, rhythm and beat information, and Mel-frequency cepstral coefficients (MFCC). These features can be used in a variety of applications, such as music information retrieval, sound classification, and speech recognition.
- One of librosa's greatest strengths is its ease of use. It offers a simple and intuitive interface for common audio analysis tasks and also offers advanced features for more specialized applications.

3.2.2 Functions and Classes

- `pandas.read_csv()` : This function reads CSV file data.

Syntax: `pd.read_csv(filepath_or_buffer, sep=',', header='infer', index_col=None, usecols=None, engine=None, skiprows=None, nrows=None)`

- **Matplotlib.pyplot.scatter()** : The `scatter()` method in the matplotlib library is used to draw a scatter plot. Scatter plots are used to observe relationship between variables and uses dots to represent the relationship between them.

Syntax:

`matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None, **kwargs)`

- **sklearn.tree.export_graphviz()** : This function generates a GraphViz representation of the decision tree and writes it to `out_file`.

Syntax:

`sklearn.tree.export_graphviz(decision_tree, out_file=None, *, max_depth=None, feature_names=None, class_names=None, label='all', filled=False, leaves_parallel=False, impurity=True, node_ids=False, proportion=False, rotate=False, rounded=False, special_characters=False, precision=3, fontname='helvetica')`

- **pydot.graph_from_dot_data()** : It is a function in the Pydot package that takes a Graphviz DOT language string as input and returns a Pydot graph object.

Syntax:

`pydot.graph_from_dot_data(dot_data, **kwargs)`

- **Syntax sklearn.tree.DecisionTreeClassifier()** : It is a class in the scikit-learn machine learning library for Python that implements a decision tree algorithm for classification tasks.

`sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)`

- **RandomForestClassifier()** : It is a class in the scikit-learn machine learning library for Python that implements random forest algorithms for classification tasks. Random forest is an ensemble learning technique that combines multiple decision trees to improve model accuracy and robustness.

Syntax:

```
sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)
```

- **Accuracy_score()** : Function in the sklearn.metrics package compute accuracy scores for a set of predicted labels compared to the true labels.

Syntax :

```
sklearn.metrics.accuracy_score(y_true, y_pred, *, normalize=True, sample_weight=None)
```

- **VarianceThreshold()**: This is a class in the scikit-learn machine learning library for Python that removes low variance features. It is often used as a preprocessing step before applying machine learning algorithms to reduce data dimensionality and improve model performance.

Syntax :

```
sklearn.feature_selection.VarianceThreshold(threshold=0.0)
```

- **selector.get_support()** : A method in the scikit-learn machine learning library for Python that returns a boolean mask indicating which features were selected by the feature selection algorithm.

Syntax :

```
selector.get_support()
```

- **GridSearchCV()** : A function that performs a grid search on a given parameter grid for a given machine learning estimator. This function will exhaustively try all combinations of parameter

values in the raster and return the best combination based on the given scoring metric.

Syntax :

```
sklearn.model_selection.GridSearchCV(estimator, param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False)
```

- **SelectKBest()** : A class in the scikit-learn library for Python that performs univariate feature selection by selecting the top k features with the highest scores. This class takes a score function as input and computes a score for each feature based on the score function.

Syntax :

```
sklearn.feature_selection.SelectKBest(score_func=<function f_classif>, k=10)
```

- **linalg.eig()** : It can compute the eigenvalues and right eigenvectors of a given square array. It takes a square array as a parameter and returns two values. The first is the eigenvalues of the array and the second is the right eigenvectors of a given square array.

Syntax : `numpy.linalg.eig()`

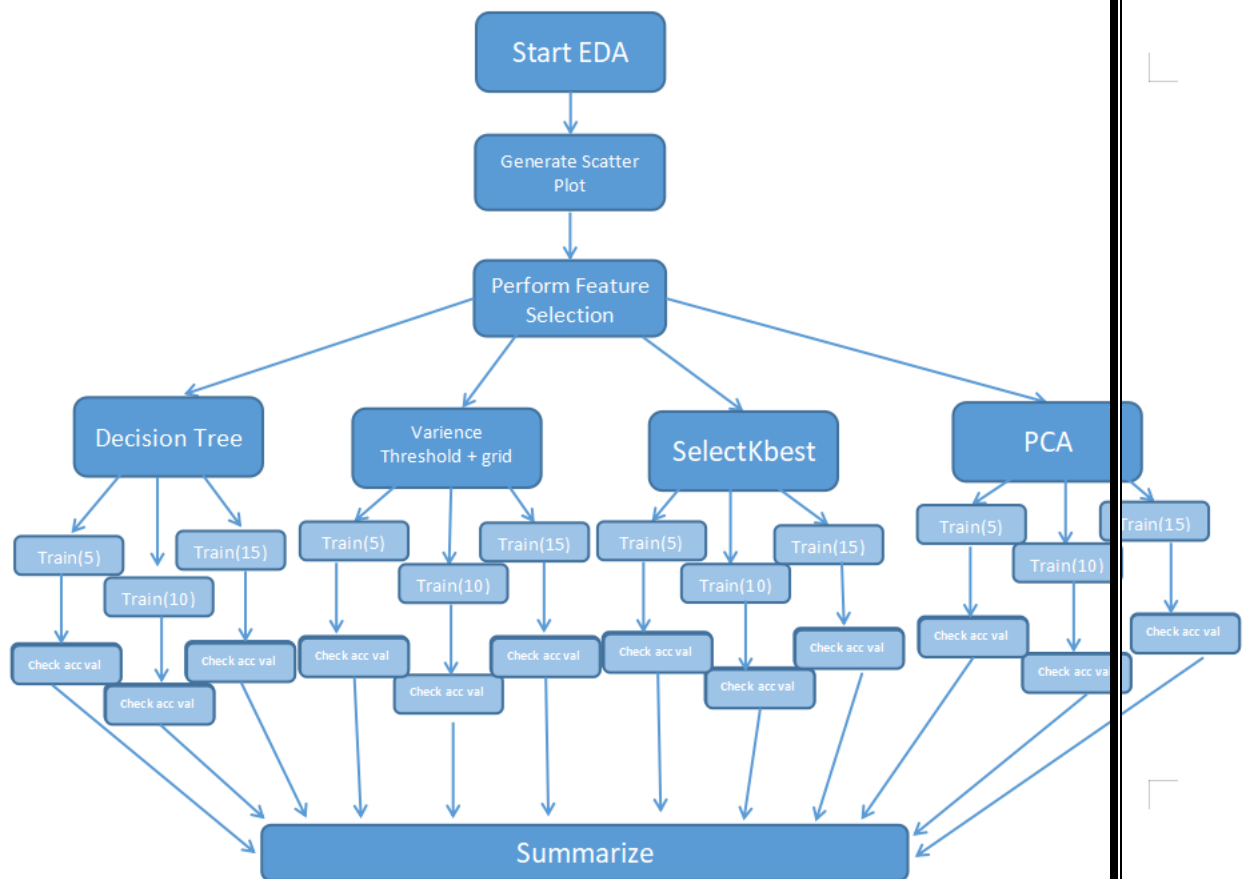
- **librosa.feature.spectral_contrast()** : A function in the Librosa library for Python that computes the spectral contrast of a spectrogram. It is commonly used in audio signal processing to analyze the timbre of sounds and extract features for classification or clustering.

```
Syntax :librosa.feature.spectral_contrast(*, y=None, sr=22050, S=None, n_fft=2048, hop_length=512, win_length=None, window='hann', center=True, pad_mode='constant', freq=None, fmin=200.0, n_bands=6, quantile=0.02, linear=False)
```

- **librosa.feature.spectral_rolloff()** : A function in the Librosa library for Python that computes the spectral rolloff of a spectrogram. Spectral rolloff is a measurement below the frequency that contains a certain percentage of the total energy in the signal.

```
Syntax :librosa.feature.spectral_rolloff(*, y=None, sr=22050, S=None, n_fft=2048, hop_length=512, win_length=None, window='hann', center=True, pad_mode='constant', freq=None, roll_percent=0.85)
```

3.3 Activity Diagram



Chapter 4:

4.1: Methodology/ Proposal:

We applied a simple methodology where the data is preprocessed and exploratory data analysis is performed and the data is randomized .The data consists of 10 classes of 100 rows each amounting upto 1000 rows and 57 columns or features. The problem is a tough classification problem consists of 10 classes.Our study focuses on four pipelines through which we are extracting the feature and presenting a comparative analysis and study for the same. After analyzing the scatter plots we understood there are no such features which can stand out on their own to give a satisfactory accuracy . The variance and possession of outliers for most of the features were similar and difference was minimal.We beleive the data we gathered from kaggle was already cleaned out.But to have minimal impact of outliers we did use random forest model at the model layer. The four pipelines used are as follows:

Decision Tree

VarianceTreshold with Grid Search

SelectKBest Features

Principal Component Analysis

We in detail now understand how did we perform our tests.

4.2: Pipelines:

Decision Tree :

A decision tree is a divide-and-conquer strategy that employs a greedy search and identifies the optimal split points while passing through the feature set tree. The process is repeated in a top-down approach in a recursive manner. We are applying the decision tree classifier from the sci-kit learn library and then using Pydot we are plotting the tree and then translating it into the png image. We have extracted three sets of features from this decision tree i.e. a set of 5 features,10 features, and 15 features. We are then measuring the accuracy of the validation set. The feature selection was manually done identifying the indexes from each of the trees and then taking the top 5,10,15 set of features from the higher level of the tree.

Music Genre Classification

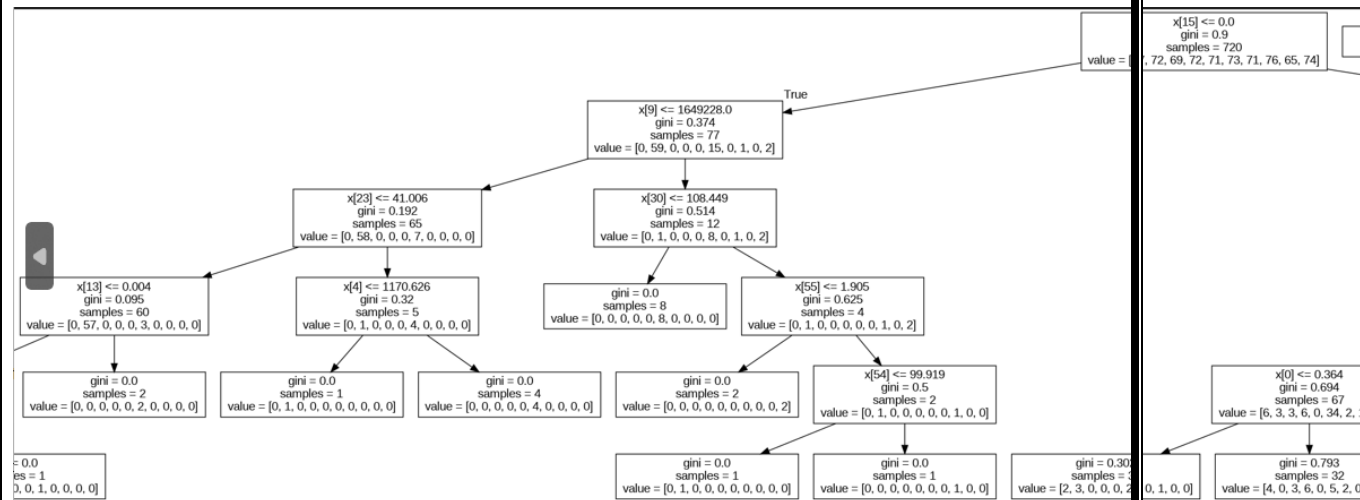


Fig:Decision Tree Model with feature set 5 PART-1

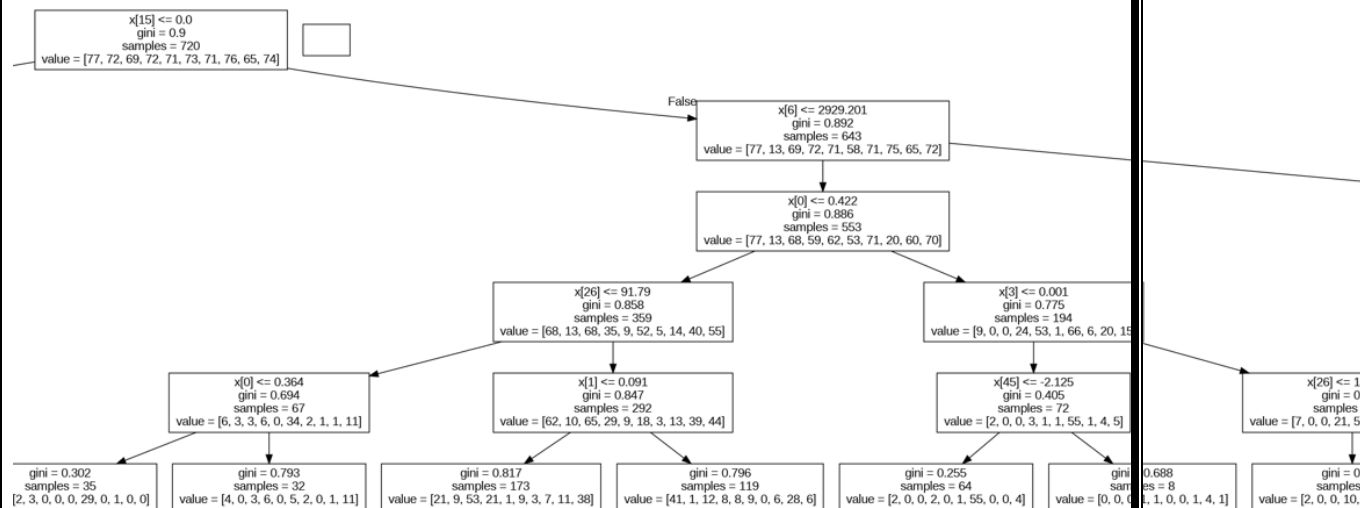


Fig:Decision Tree Model with feature set 5 PART-2

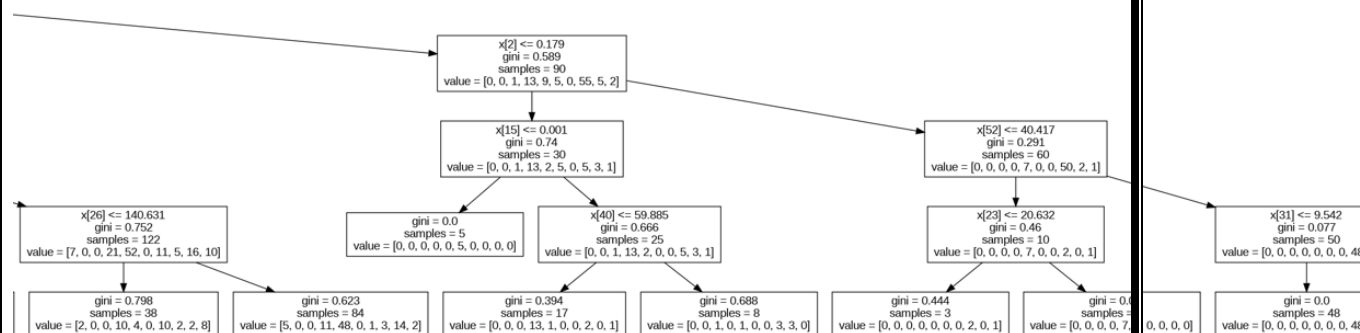


Fig:Decision Tree Model with feature set 5 PART-3

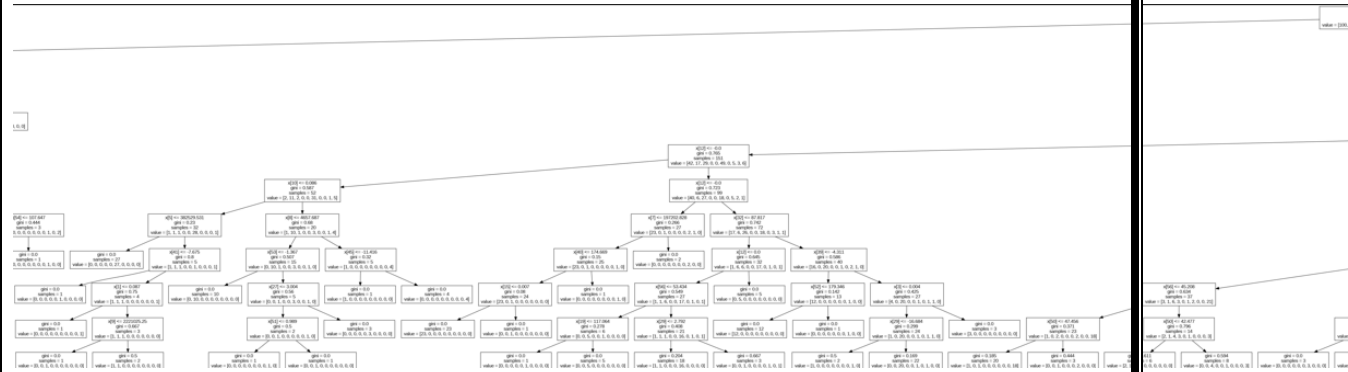


Fig:Decision Tree Model with feature set 10 PART-1

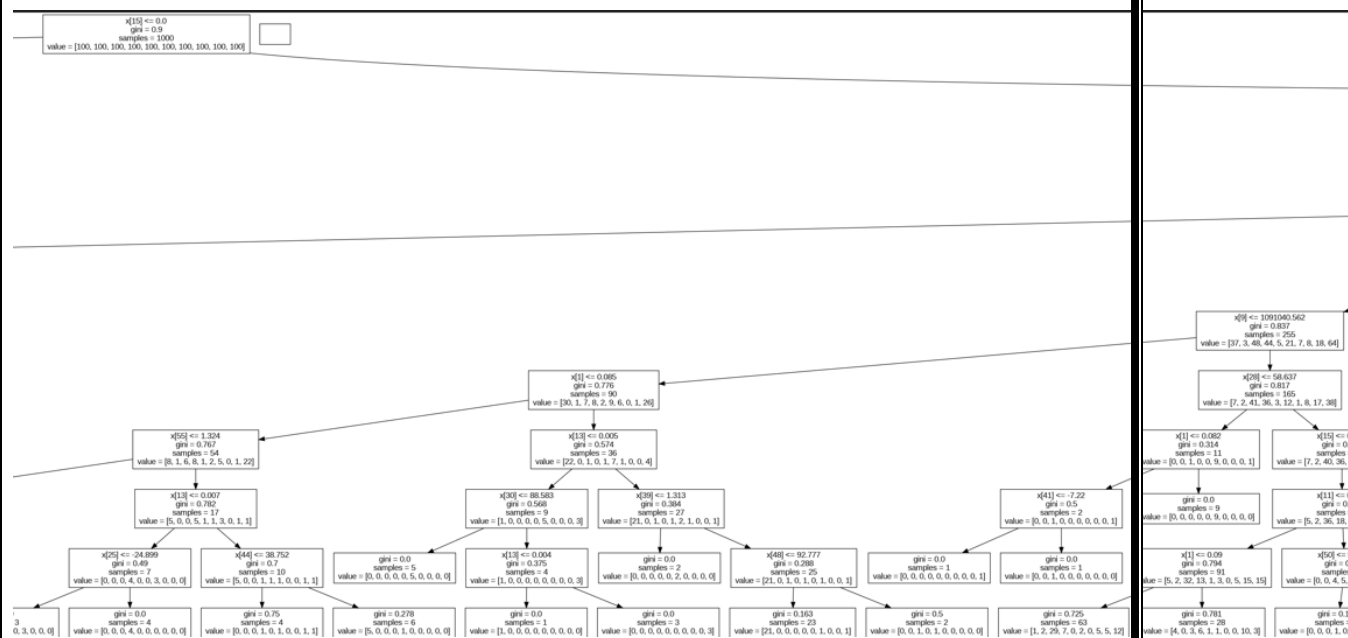


Fig:Decision Tree Model with feature set 10 PART-2

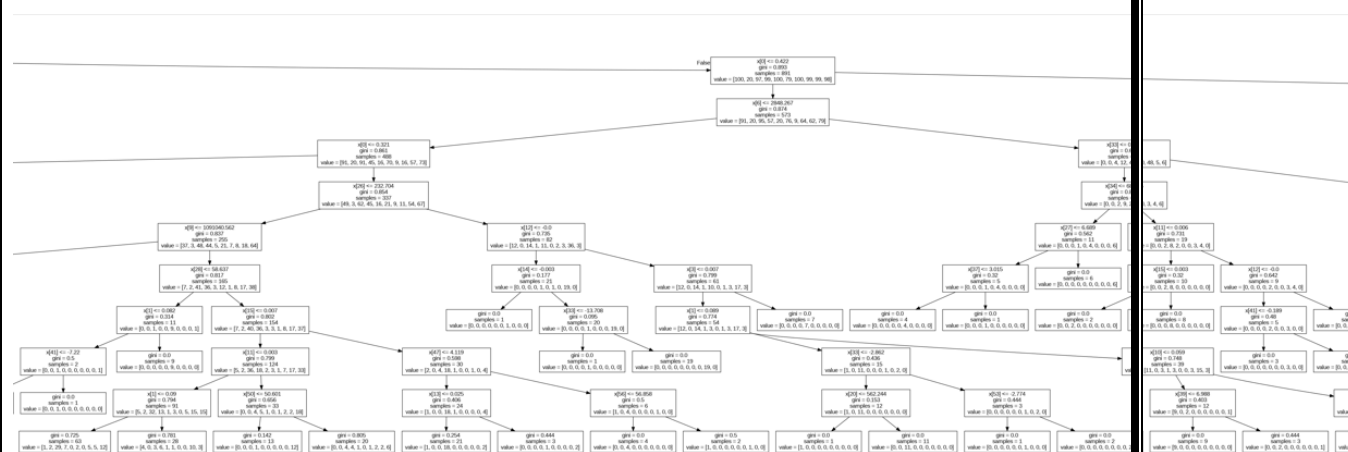


Fig:Decision Tree Model with feature set 10 PART-3

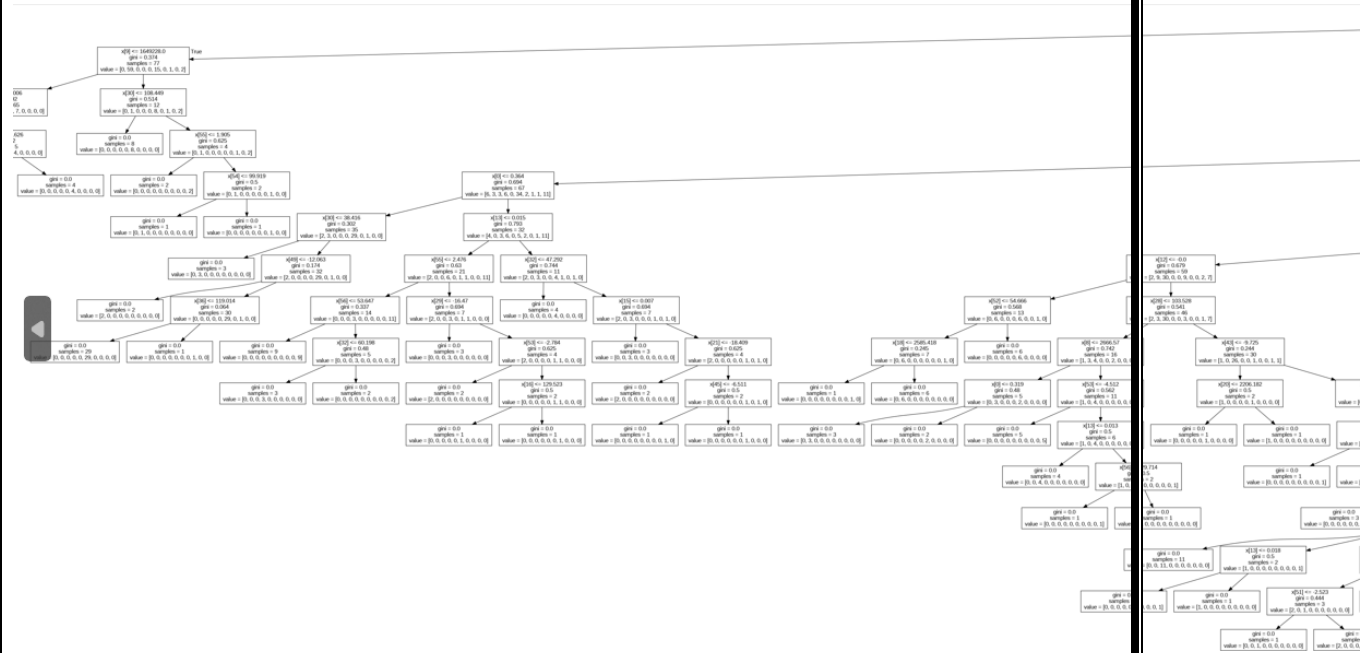


Fig:Decision Tree Model with feature set 15 PART-1

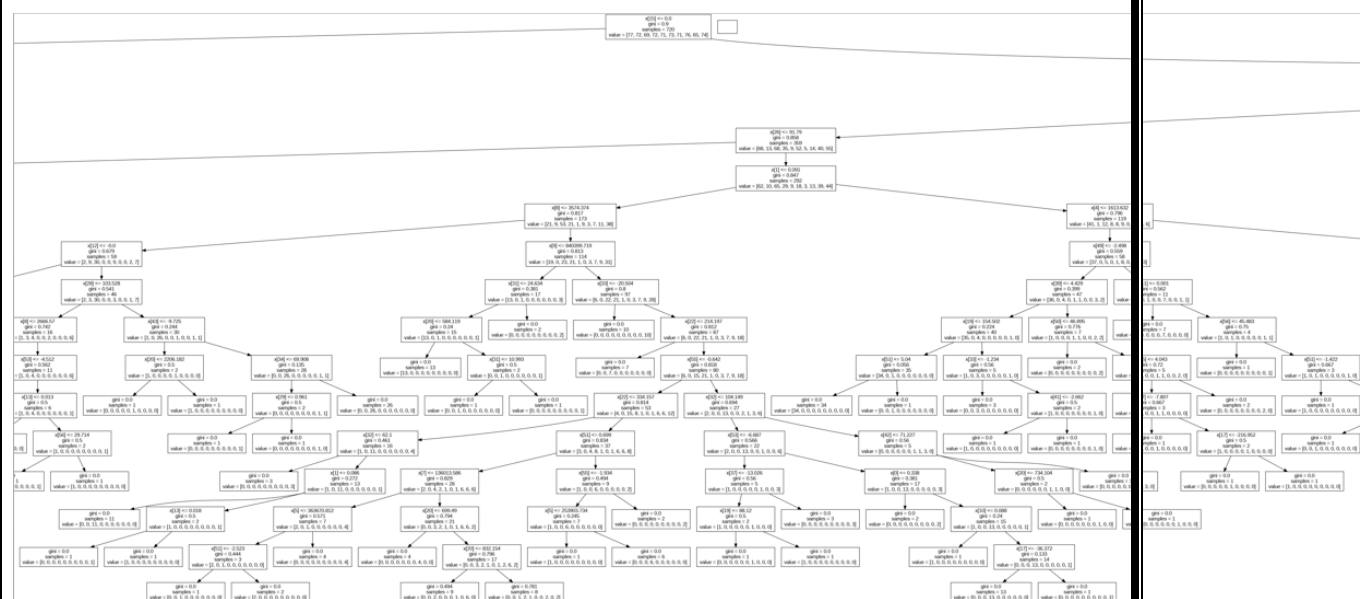


Fig:Decision Tree Model with feature set 15 PART-2

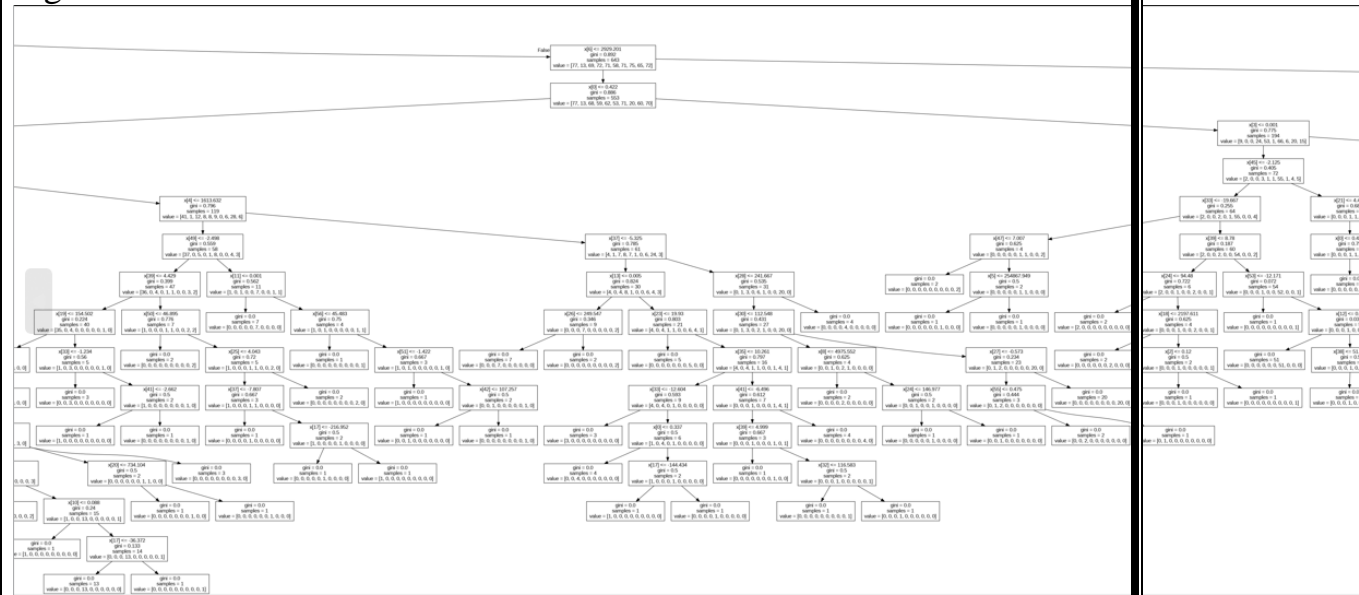


Fig:Decision Tree Model with feature set 15 PART-3

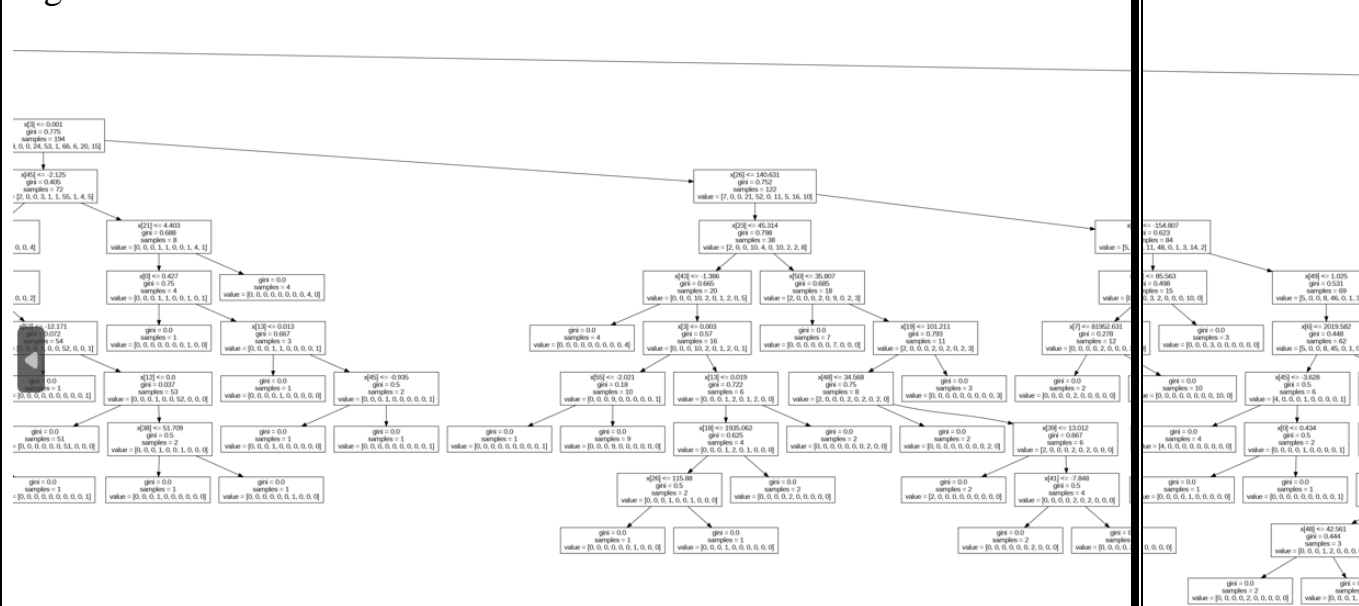
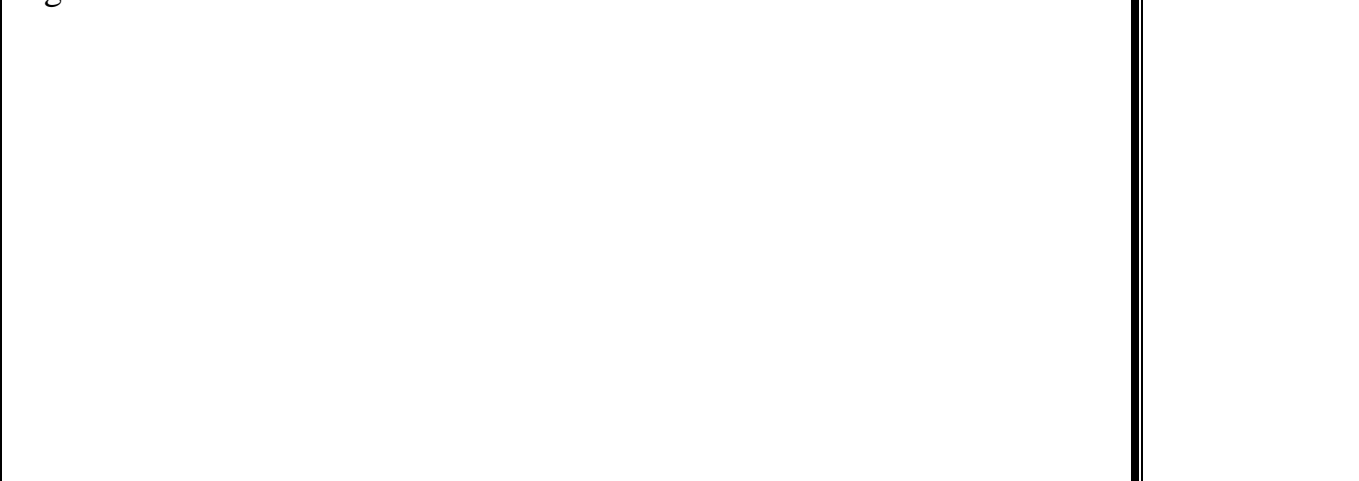


Fig:Decision Tree Model with feature set 15 PART-4



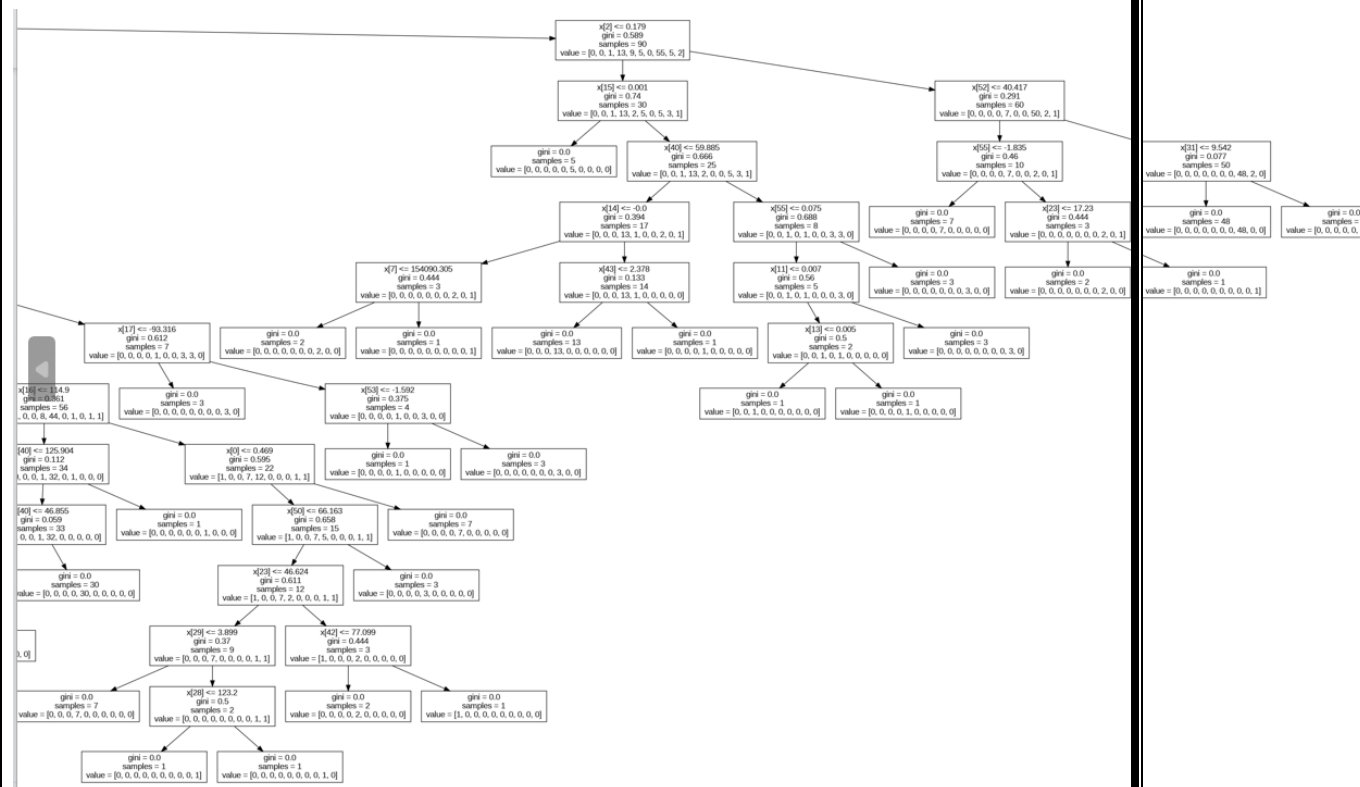


Fig:Decision Tree Model with feature set 15 PART-5

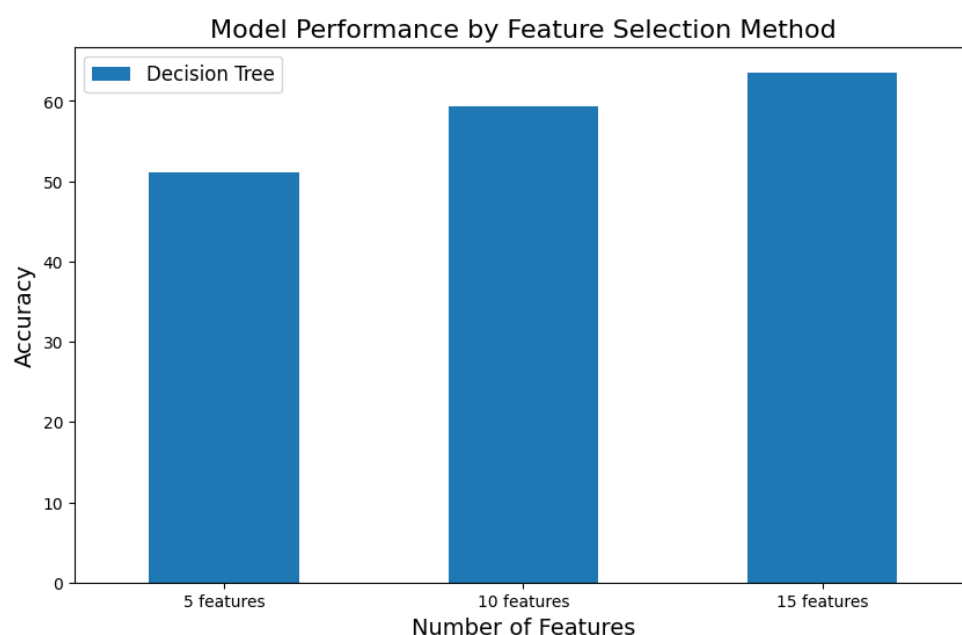


Fig:Comparison of accuracy with no of features for decision tree.

Variance Threshold With Grid Search :

The variance threshold is a feature selection method where the significance level is calculated using the variance. The lower the score the lower the importance. It is a very simple function that takes a

significant level as input in the constructor. In our case, this no is 0.05. That means any score lesser than 0.05 will not be taken into consideration.

Now out of the features that are retained, we have selected 5,10,15 sets of features respectively into consideration. To improve our model a bit we used grid search too on the random forest model to find out the optimal hyperparameters. We need to keep in mind that grid search is not a technique of feature selection so individually alone it is not possible for the grid search to select a set of features. Hence variance threshold comes into the picture.

The parameters used in grid search are:-

n_estimators: It specifies the no of trees the random forest model should use. In our case, we tested it using three values 10,20,30

max_depth: It specifies the maximum depth the tree should be. In our case, we tested it using 3,5,10

min_samples_split: The min_sample_split specifies the minimum no of samples that are taken into consideration for splitting a node. In our case, the values taken into consideration are 2,5,10

min_samples_leaf=The min_sample_split specifies the minimum no of samples that is req to be at a leaf node. The values taken into consideration are 1,2,4

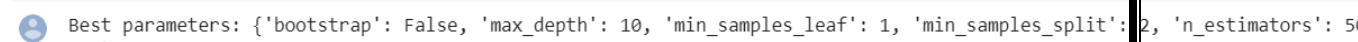
bootstrap: This specifies whether or not the bootstrap sample is being used. The bootstrap sample is a sample that is taken by some kind of sampling from a larger sample. It has a boolean value.

The other parameters that were specified are

cv:-It specifies the folds that are necessary for cross-validation. In our case, we divided it into 5 folds.

n_jobs:-Is an instance attribute that specifies the no of jobs that run in parallel for both fit and predict. We used value -1 which signifies that it is set automatically mapping to no of cores.

The features selected are then passed on to the random forest model.



```
Best parameters: {'bootstrap': False, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
```

Fig:The best HyperParameters found out.

```
[ ] x_selected_test=selector.transform(x_test)
indices = selector.get_support(indices=True)[:5]
X_top5 = x_selected_test[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_test)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

Best parameters: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Best accuracy score: 0.4821428571428571

Fig:Model for 5 features.

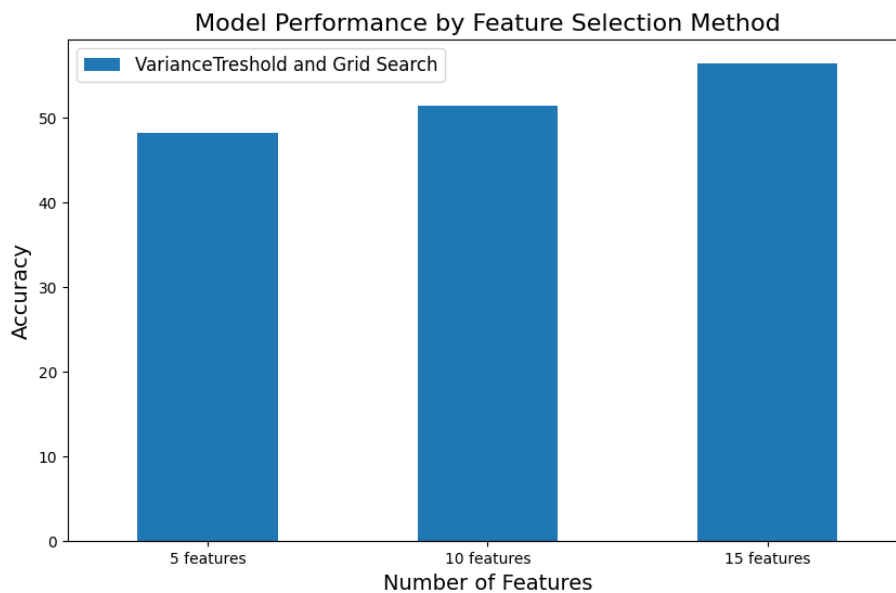


Fig:Comparison of accuracy with no of features for VarianceThreshold with Grid Search.

SelectKBest Feature Selection:

K-Best is a machine learning feature selection strategy that chooses the K best features from a dataset using a scoring function. A statistical metric

called the scoring function is often used to quantify how relevant a characteristic is to the target variable.

It selects the top K features by ranking the features based on their ratings. When working with high-dimensional datasets, where many characteristics could be redundant or irrelevant and could cause overfitting and poor model performance, this strategy is helpful.

K-Best is an example of a univariate feature selection, which means that each feature is taken into account separately from the others.

Here in this project we used this model to identify the most informative features from the 'features_30_sec.csv' file for gaining better accuracy and to omit other unnecessary features present in the dataset. It also reduces the computational cost as well as memory and time needed for testing the model.

Mainly, the Scikit learn library provides this SelectKBest class for extracting the best features of the given dataset.

```
In [30]: from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
selector = SelectKBest(score_func=mutual_info_classif, k=5)
X_new = selector.fit_transform(x_train, y_train)
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
X_new_test=selector.transform(x_test)
# evaluate performance on test set
accuracy = rfc.score(X_new_test, y_test)
print(f"Accuracy: {accuracy}")
```

Accuracy: 0.4857142857142857

For scoring function calculation we imported the mutual_info_classif function and random forest classifier from sklearn.ensemble library.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
```

A.

1. At first step, we selected top 5 features from the training set. those selected features accordingly extracted by the selector object. The resulting feature matrix is stored in X_new variable.

```
X_new = selector.fit_transform(x_train, y_train)
```

2. It creates a Random Forest classifier named rfc with 100 estimators and random seed 79 and trains it on X_new and y_train using fit() method.

```
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
```

3. Now time to test the trained model on the test dataset for checking the accuracy. For that X_new_test variable used to store the testing data which is shaped using transform() method to only those specified 5 features.

```
X_new_test=selector.transform(x_test)
```

4. Finally accuracy is calculated, coming 48.57% for k = 5 features.

B.

Secondly, we changed the K paramter value and choose 10 top features from the dataset, and done the same steps accordingly to the step A process. and calculated the accuracy that is coming 57.85% .

Again, selecting top 15 features we calculated that accuracy = 58.92%

```
In [23]: selector = SelectKBest(score_func=mutual_info_classif, k=15)
X_new = selector.fit_transform(x_train, y_train)
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
X_new_test=selector.transform(x_test)
# evaluate performance on test set
accuracy = rfc.score(X_new_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.5892857142857143
```

So finally we reached into a decision, that If we stepwise increase the top selected features value accordingly then the accuracy value is testing also increases in a parallel manner. Thus, we tried to minimize the imbalanced dataset to a finalized compact top most features contained training and

testing set that gives the better accuracy than previous one feature selection 'Decision Tree' methods 3 pipelines.

Comparisinal analysis using Barplot for k= 5, 10, 15 selected featured pipelines:

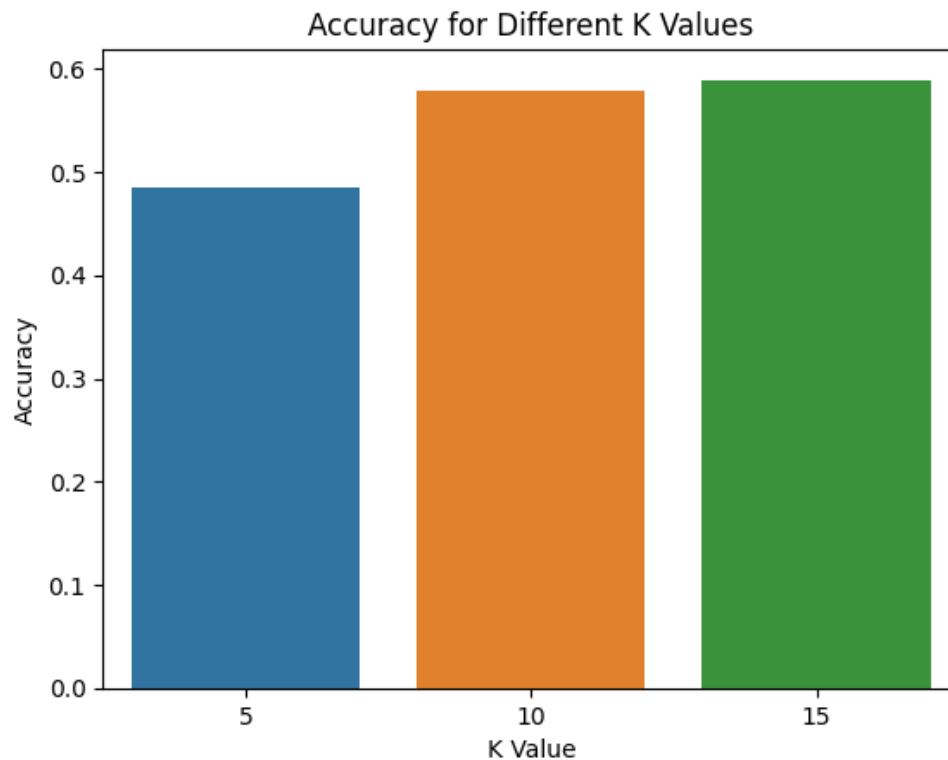


Fig:Comparison of accuracy with no of features for SelectKbest.

Principal component analysis :

This is a feature engineering technique where we find out new features from the existing ones by combining the existing features. Basically used for dimensionality reduction. It transforms humongous datasets into smaller datasets by finding out the eigenvectors after it is transformed into a covariance matrix. The eigenvectors are then taken into consideration for framing out the features. Again we did consider making three sets of features 5,10,15 from the dataset. We then feed the data to the random forest model for classification.

```
[ [ 6.57332584e-02  2.56527459e-01 -7.51809295e-02  6.56530244e-02
   1.40501784e-01]
 [ -1.72003928e-01 -5.97730482e-02 -9.27370880e-03  8.85482457e-02
   2.79860990e-02]
 [ -2.32429707e-02  2.35244831e-01 -4.98011448e-02 -3.20929440e-02
  -7.16191393e-02]
 [ -1.38621051e-01  1.68619953e-01 -5.47527969e-02  3.26039354e-02
  -6.49441736e-02]
 [ -1.84175372e-02  2.72200916e-01 -4.89402281e-02 -1.48131270e-01
  -1.52417304e-01]
 [ -1.73843894e-01  1.92907145e-01 -8.34402793e-03  6.26793500e-02
  -3.06965382e-02]
 [ -5.73011534e-02  2.71808975e-01  1.30443829e-02 -1.22673431e-01
  -9.09080751e-02]
 [ -1.56876427e-01  6.94846290e-02  4.85427840e-02  1.84191794e-01
   1.12679729e-01]
 [ -2.49047356e-02  2.76691176e-01 -3.66887021e-02 -1.40297816e-01
  -1.28908405e-01]
 [ -1.69540936e-01  1.26646069e-01  3.61808293e-02  1.42615308e-01
   5.85074895e-02]
 [  4.00509426e-02  2.04955166e-01 -9.05567015e-02 -1.66026906e-01
  -2.05240846e-01]
 [ -1.34336150e-01  1.98198237e-01 -5.76739256e-02 -2.40313039e-02
  -1.12441759e-01]
 [ -3.50682747e-02  1.07921150e-02  4.86601676e-02  2.49613768e-04
  -8.09639229e-02]
```

Fig: The eigenvectors of the dataset features.

cov_matrix

```
array([[ 1.00139082, -0.40976256,  0.52350802, ..., -0.46930896,
         0.25585672, -0.54451851],
       [-0.40976256,  1.00139082, -0.05710093, ...,  0.32694612,
        -0.17371853,  0.32068542],
       [ 0.52350802, -0.05710093,  1.00139082, ..., -0.16165175,
         0.12658622, -0.2265514 ],
       ...,
       [-0.46930896,  0.32694612, -0.16165175, ...,  1.00139082,
         0.14686739,  0.8740447 ],
       [ 0.25585672, -0.17371853,  0.12658622, ...,  0.14686739,
         1.00139082,  0.0913444 ],
       [-0.54451851,  0.32068542, -0.2265514 , ...,  0.8740447 ,
         0.0913444 ,  1.00139082]])
```

Fig: The covariance of the dataset features.

We did also print the singular values of the principal component :-

```
[3.98648979e+07  6.36344676e+06  1.16984615e+06  6.78655828e+04
 3.19358661e+04]
```


Fig:The singular values of the 5 features.

```
[3.98648979e+07 6.36344676e+06 1.16984615e+06 6.78655828e+04  
3.19358661e+04 6.32261419e+03 6.11387361e+03 3.62056090e+03  
3.08609493e+03 2.47125109e+03]
```

Fig:The singular values of the 10 features.

```
[3.98648979e+07 6.36344676e+06 1.16984615e+06 6.78655828e+04  
3.19358661e+04 6.32261419e+03 6.11387361e+03 3.62056090e+03  
3.08609493e+03 2.47125109e+03 2.02956945e+03 1.62630393e+03  
1.42982234e+03 1.23693111e+03 1.04364819e+03]
```

Fig:The singular values of the 15 features.

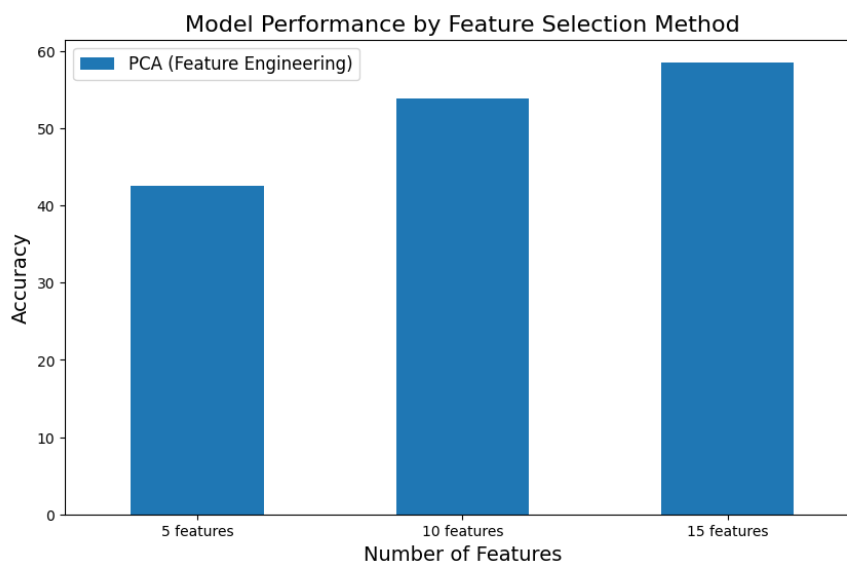
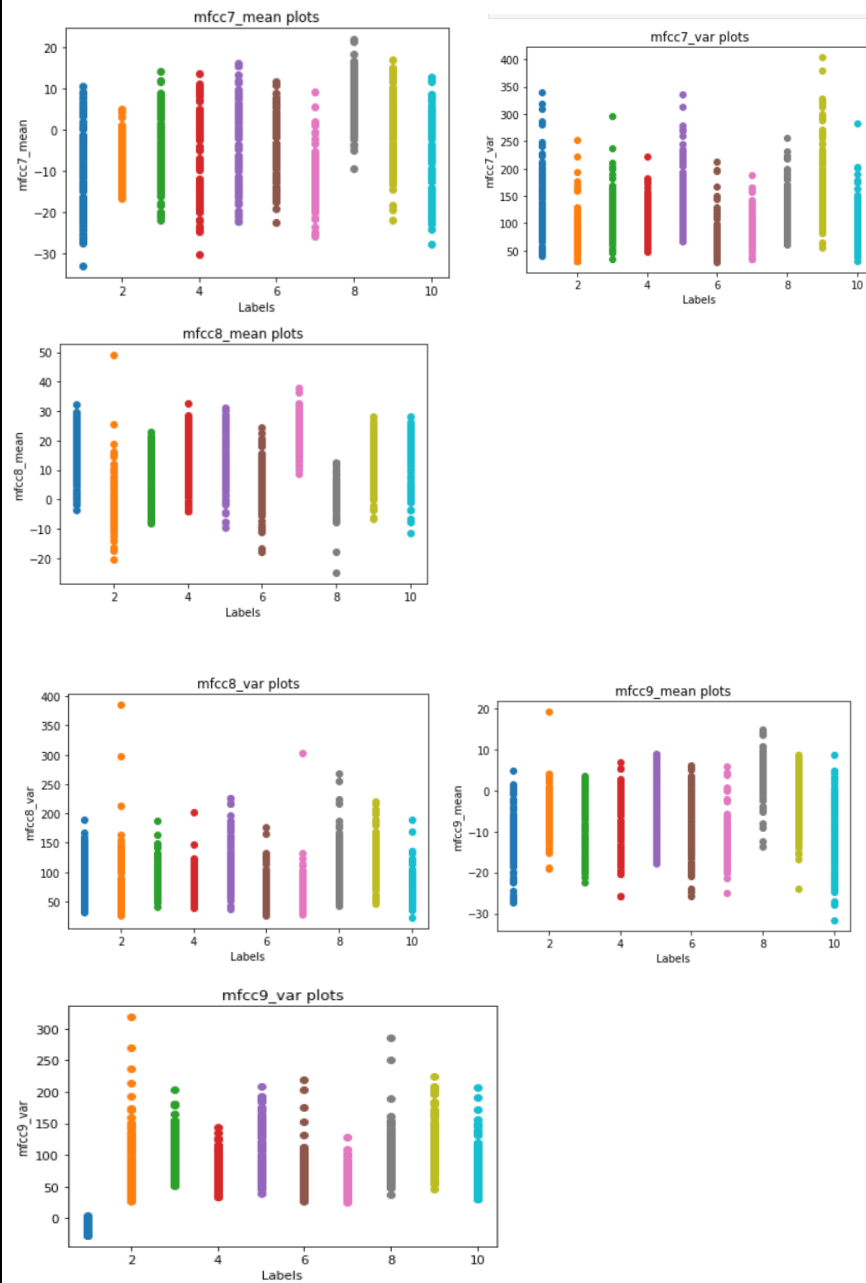


Fig:Comparison of accuracy with no of features for Principal Component Analysis.

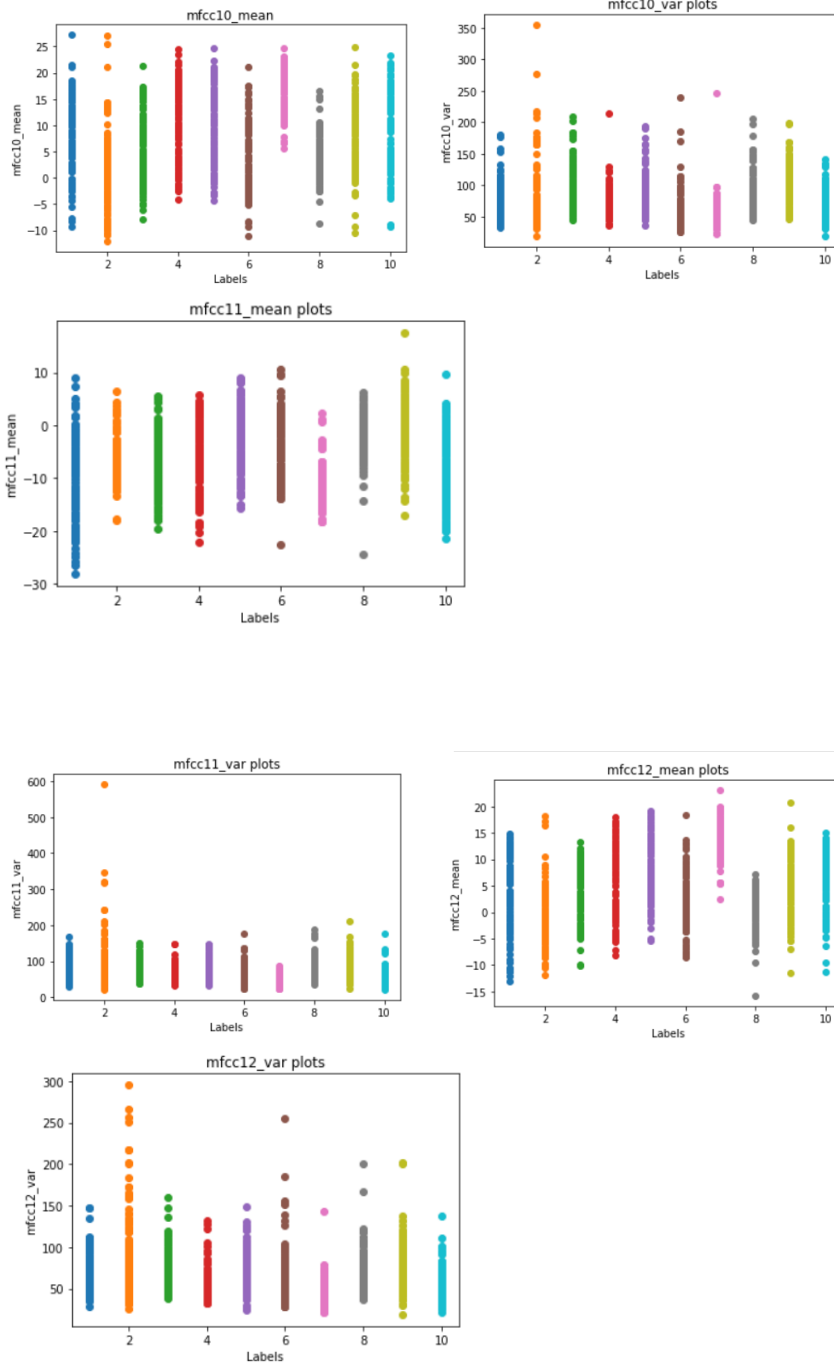
4.3 Result Analysis:

The scatter plots generated during EDA:-

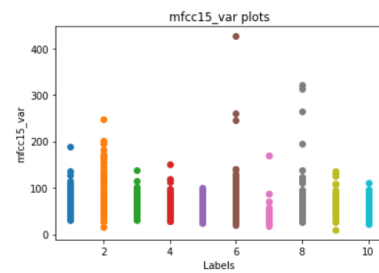
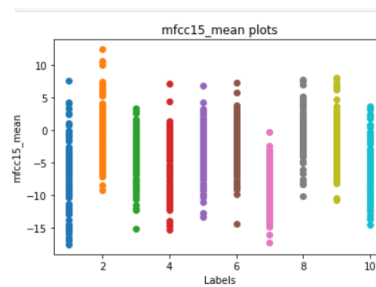
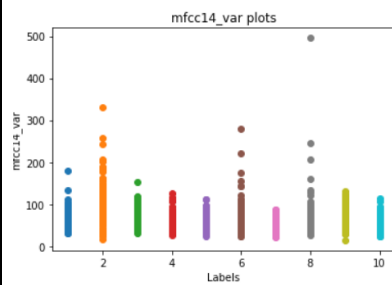
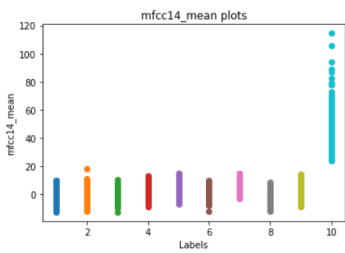
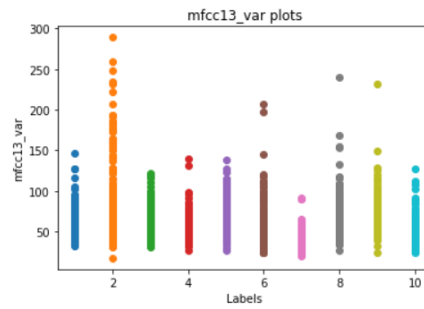
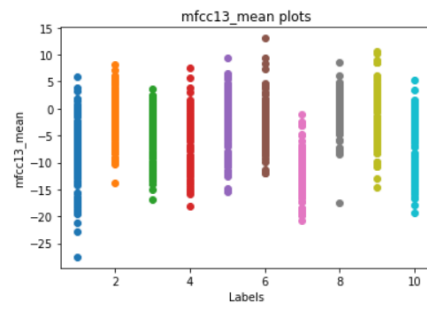
Music Genre Classification



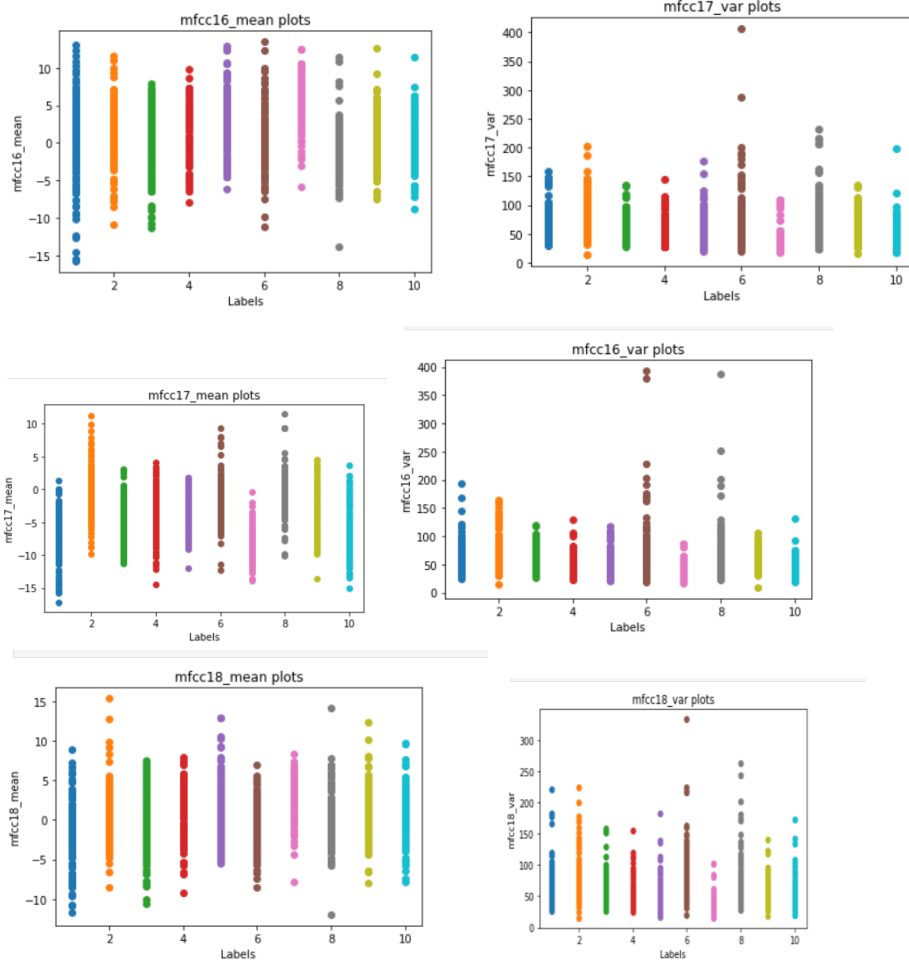
Music Genre Classification



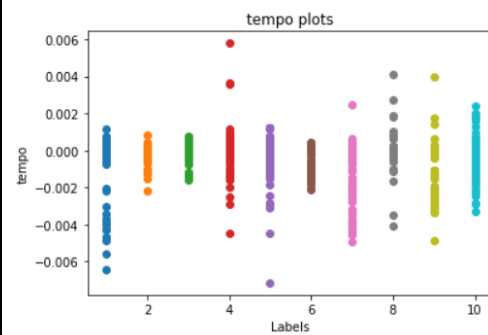
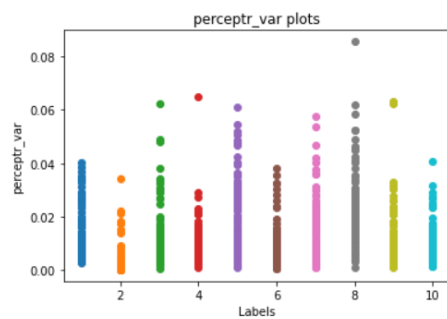
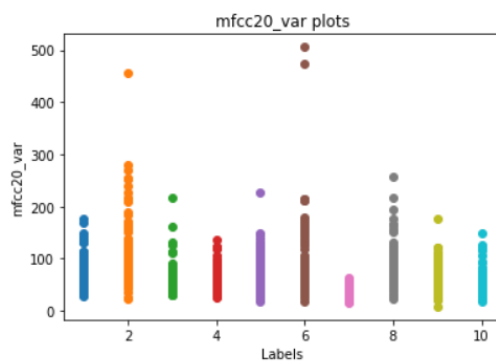
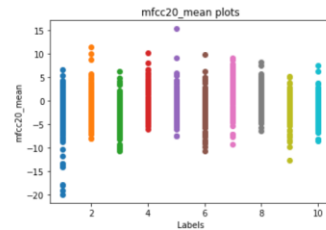
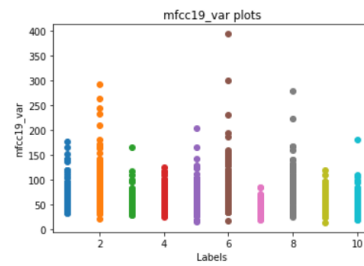
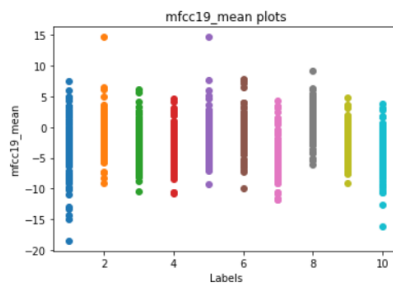
Music Genre Classification



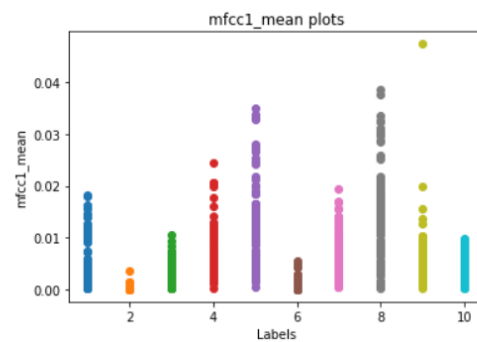
Music Genre Classification



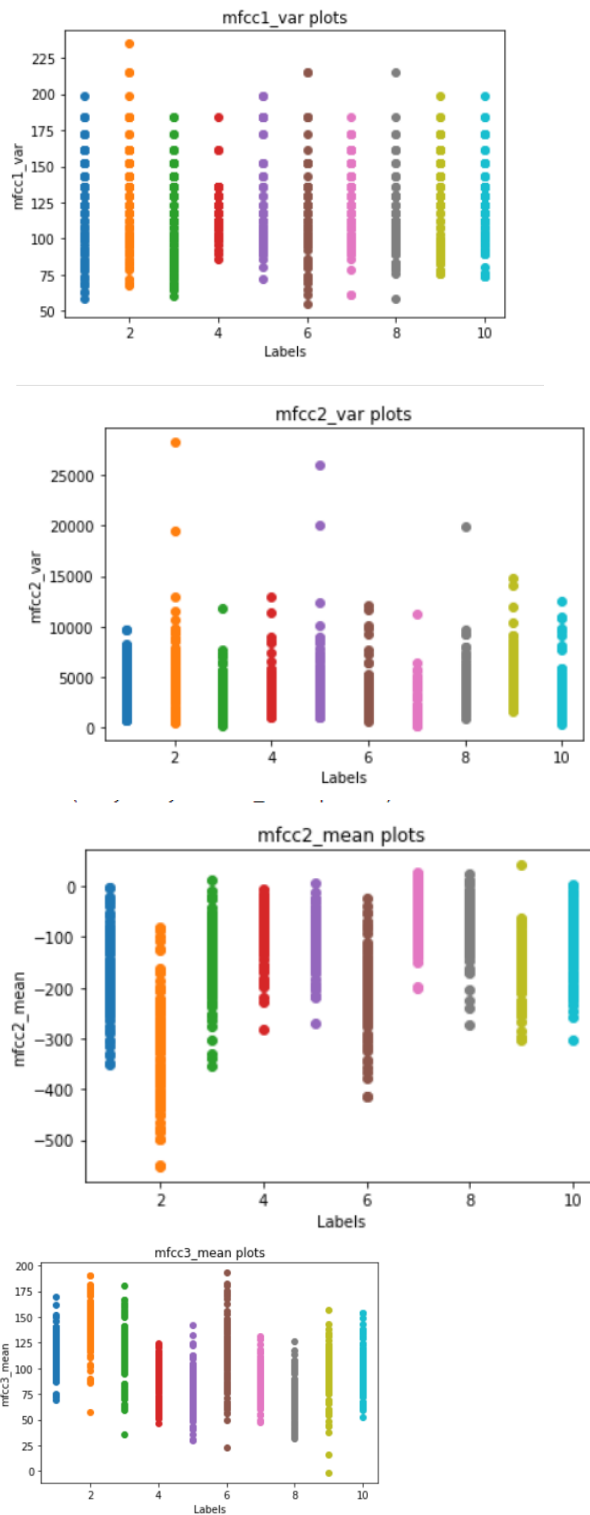
Music Genre Classification



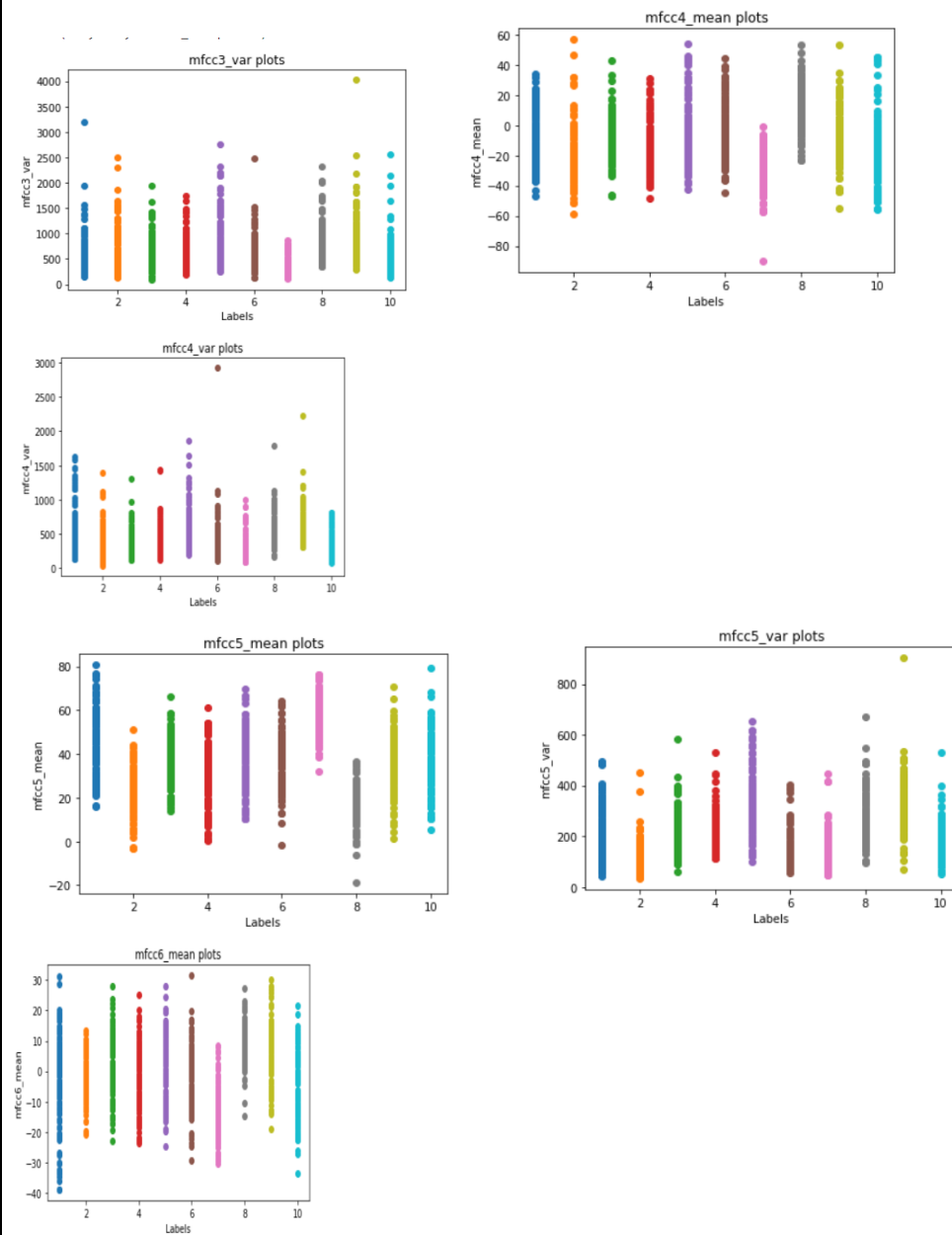
ax[0, 0].text(0.5, 1.0, 'mfcc1_mean plots')



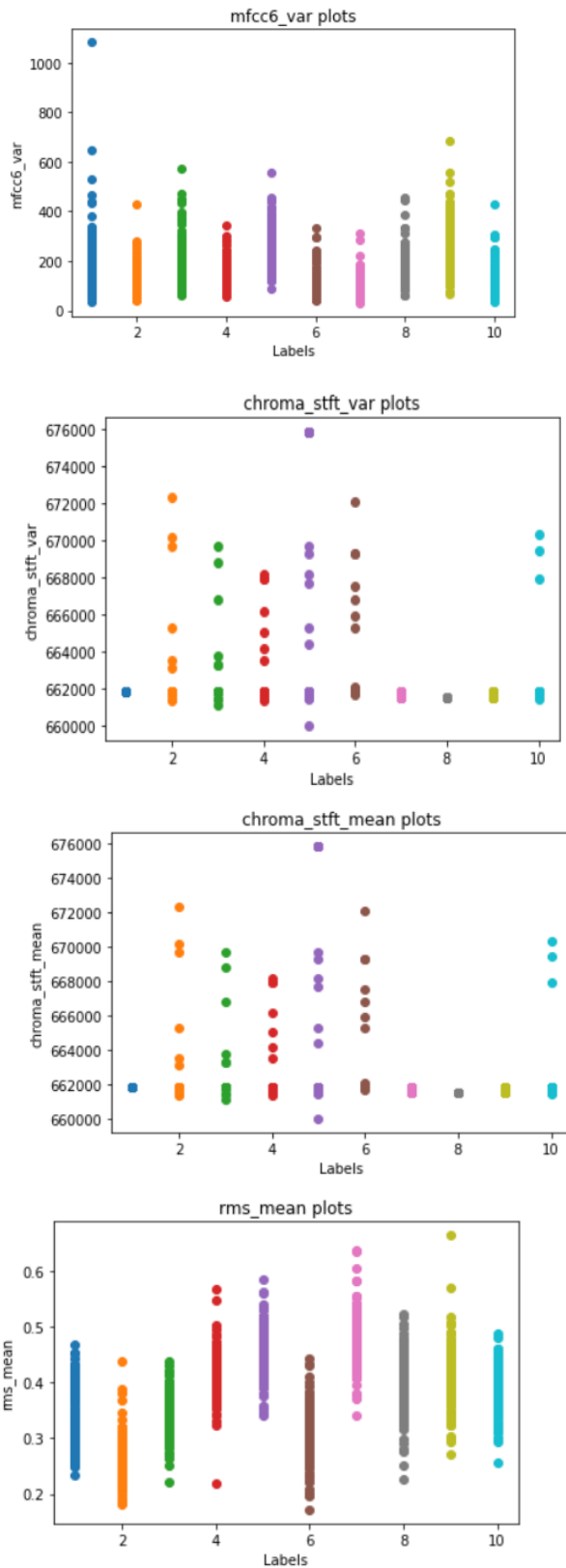
Music Genre Classification



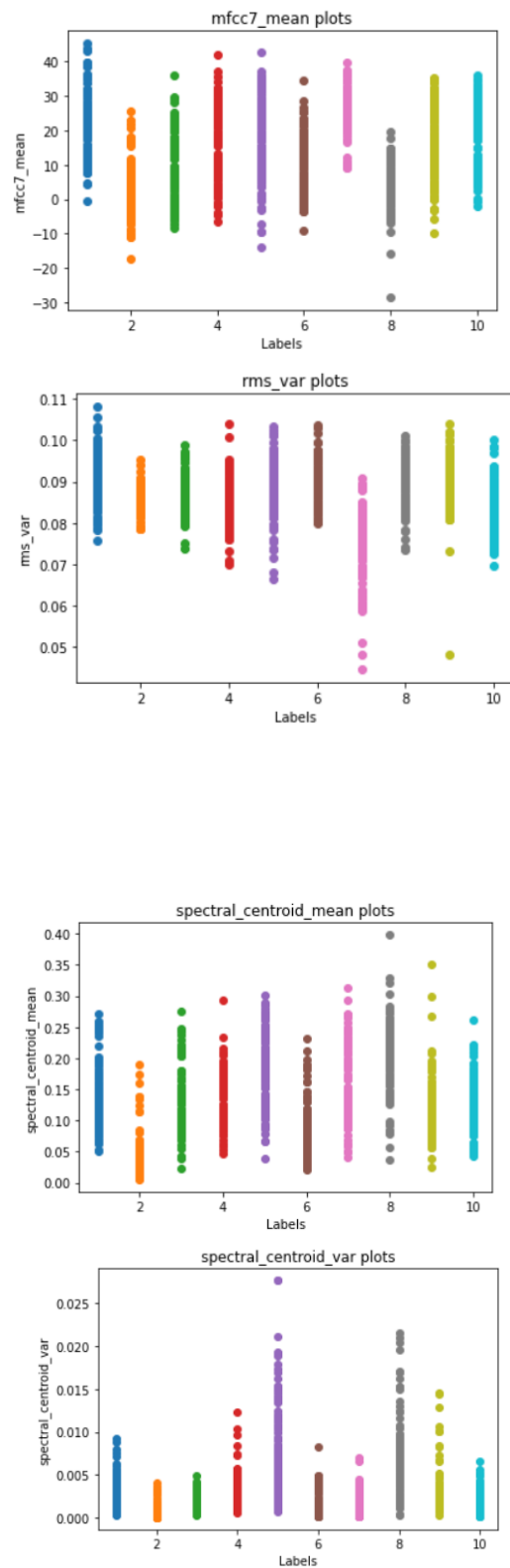
Music Genre Classification



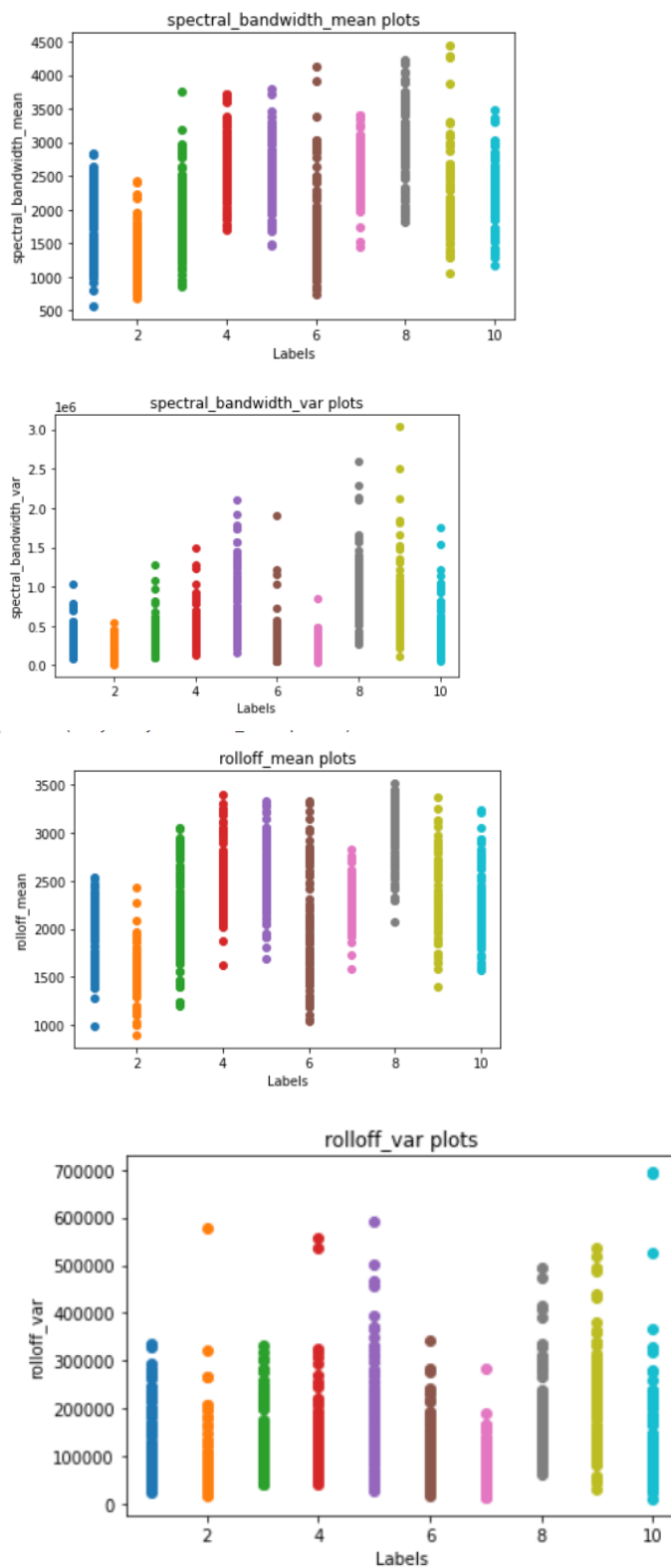
Music Genre Classification



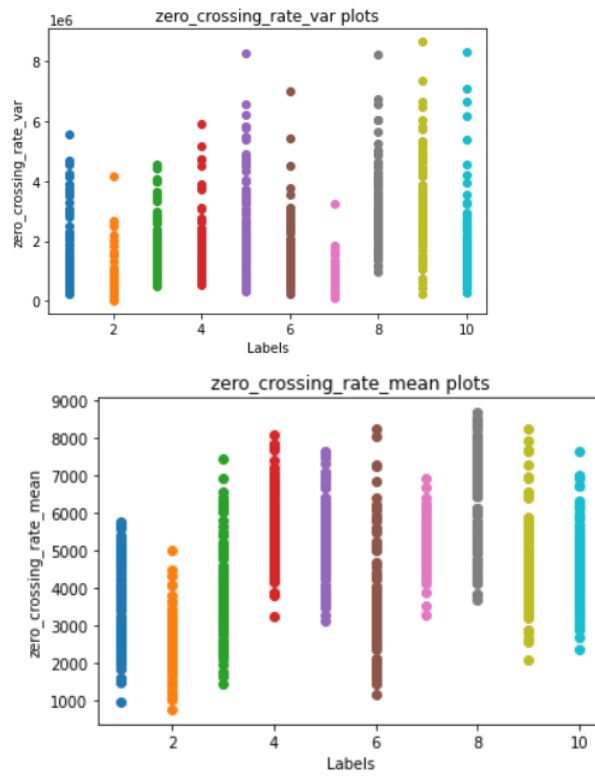
Music Genre Classification

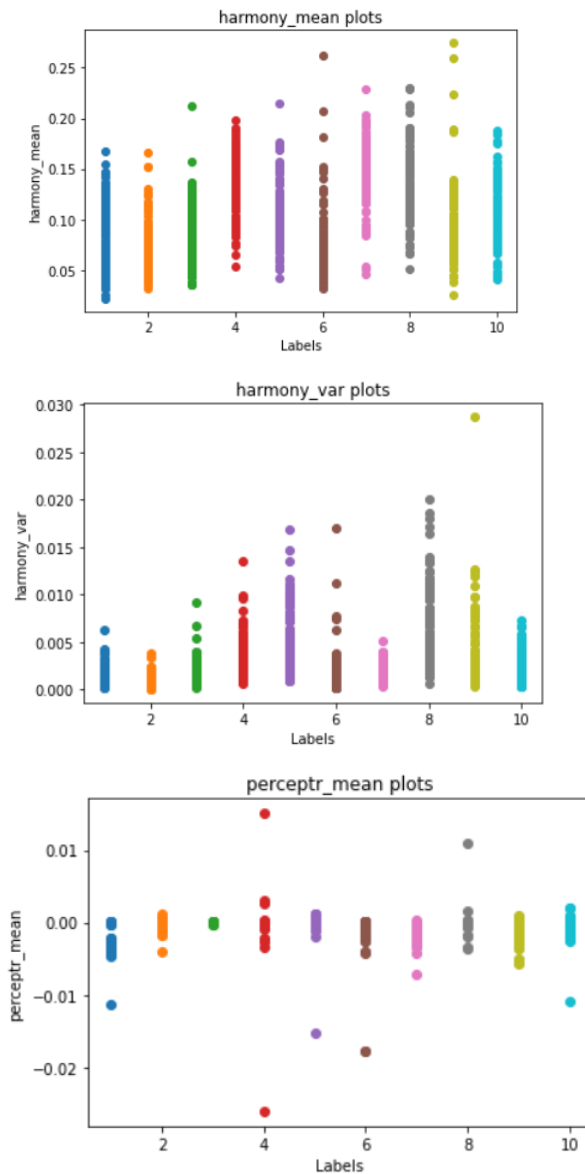


Music Genre Classification



Music Genre Classification





The Result :-

The accuracy in the test set that we have achieved for the three different pipelines is listed as follows:

The Decision tree :

5 features:51.0714

10 features:59.2857

15 features:63.5714

The VarianceTreshold and Grid Search:

5 features:48.214

10 features:51.42857

15 features:56.428

The SelectKbest Feature selection using mutual_info_classif

5 features:48.5714

10 features:57.85714

15 features:58.92857

The PCA (Feature Engineering:) results:

5 features:42.5

10 features:53.928

15 features:58.57

The study thus show the best way to do feature selection for music genre classification where the model used is Random Forest with random state 79 being

the best one according to this dataset is The Decision Tree.

The order from best to worst :Decision Tree>Select-K-Best>Variance Treshold.

The PCA performed bad when applied on 5 feature set,but did catch up a nd perform better than variance threshold on 10 and 15 which denotes with increasing no of features the performance of PCA increased significantly.

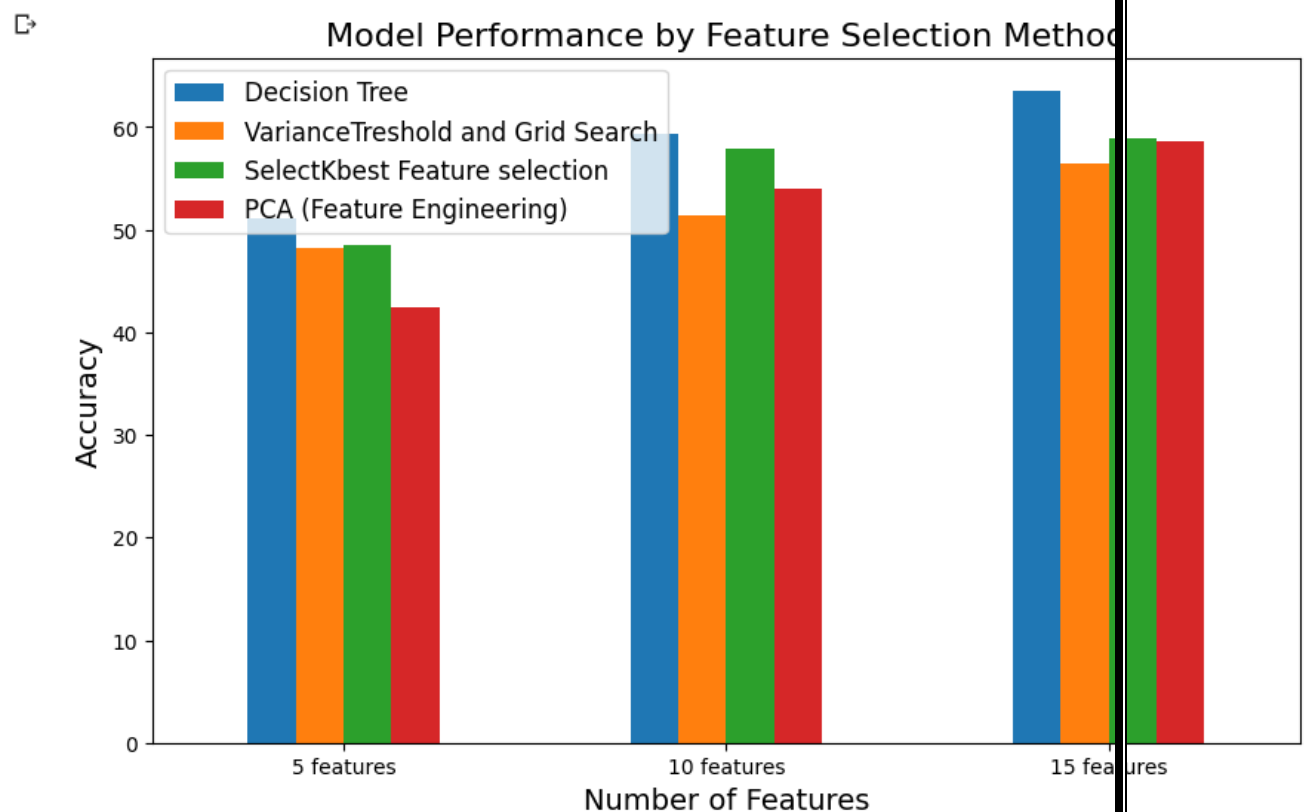


Fig:The comparison of accuracy of the models.

Chapter 5:

Standards Adopted:

5.1 Design Standards:

Our problem statement and objective is to comparisinal analysis among feature selection model on which our dataset is are properly trained. The design standards of this project may vary depending on the project objectives and circumstances.

Data Integrity:

We collected the dataset from Kaggle by checking all important necessities in it like Citations, licence verification, total viewers, total downloads - like criterias are tested so that the authenticity, completeness, reliability of our data remains unchanged. We documented the data sources, data preprocessing steps are also done. We did the Exploratory data analysis and implemented scatter plots on choosing all 57 features by dividing them on a particular range.

Feature Selection:

Now, all of these features are not at all important for our project model performance. So we applied feature selection techniques and feed our data on those 4 techniques named - Decision Tree, SelectKBest, Variance Threshold with Grid-Search, PCA and checked the best top most features coming for different parameters and estimators value in Random Forest, as well as in SelectKBest by changing the K parameter value.

Model Selection:

Now, for Music Genre classification, many ML models can be used like - Decision Tree, SelectKBest, Grid Search etc. After making train-test splitting data, we used those feature selection models and created 3 different pipelines for 5, 10 and 15 features to make a comparision between them.

Model Evaluation:

Now, individual accuracy has been calculated and visualized them in a frame to showcase the performance of training and testing model in different selection models.

5.2 Coding Standard:

```
In [50]: from sklearn.metrics import accuracy_score

clf_final=RandomForestClassifier(random_state=79)
clf_final.fit(df_to_train,y_train)
y_pred=clf_final.predict(df_to_test)
acc = accuracy_score(y_test, y_pred)

print("Accuracy:", acc)
```

Accuracy: 0.6357142857142857

```
In [19]: X_selected = selector.fit_transform(x_train)
indices = selector.get_support(indices=True)[:15]
X_top5 = X_selected[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_train)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

Best parameters: {'bootstrap': False, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Best accuracy score: 0.6069444444444445

```
In [23]: selector = SelectKBest(score_func=mutual_info_classif, k=15)
X_new = selector.fit_transform(x_train, y_train)
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
X_new_test=selector.transform(x_test)
# evaluate performance on test set
accuracy = rfc.score(X_new_test, y_test)
print(f"Accuracy: {accuracy}")
```

Accuracy: 0.5892857142857143


```
In [46]: rf_model = RandomForestClassifier(random_state=79)
rf_model.fit(X_train_pca ,y_train)
accuracy = rf_model.score(X_test_pca, y_test)
print(f"Accuracy: {accuracy}")
```

Accuracy: 0.5857142857142857

5.3 Testing Standard:

```
In [63]: f=["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_mean","mfcc7_var","chroma_stft_mean","rms_mean"]
import librosa
import numpy as np

# Load the WAV file
audio_data, sr = librosa.load('/content/reggae.00005.wav')

# Extract features
perceptr_var = np.var(librosa.feature.spectral_contrast(y=audio_data, sr=sr)[1])
rolloff_var = np.var(librosa.feature.spectral_rolloff(y=audio_data, sr=sr))
spectral_bandwidth_mean = np.mean(librosa.feature.spectral_bandwidth(y=audio_data, sr=sr))
mfcc4_mean = np.mean(librosa.feature.mfcc(y=audio_data, sr=sr)[3])
mfcc7_var = np.var(librosa.feature.mfcc(y=audio_data, sr=sr)[6])
chroma_stft_mean = np.mean(librosa.feature.chroma_stft(y=audio_data, sr=sr))
rms_mean = np.mean(librosa.feature.rms(y=audio_data))
harmony_var = np.var(librosa.effects.harmonic(y=audio_data))
spectral_centroid_mean = np.mean(librosa.feature.spectral_centroid(y=audio_data, sr=sr))
mfcc20_mean = np.mean(librosa.feature.mfcc(y=audio_data, sr=sr)[19])
mfcc5_var = np.var(librosa.feature.mfcc(y=audio_data, sr=sr)[4])
rms_var = np.var(librosa.feature.rms(y=audio_data))
mfcc18_var = np.var(librosa.feature.mfcc(y=audio_data, sr=sr)[17])
chroma_stft_var = np.var(librosa.feature.chroma_stft(y=audio_data, sr=sr))
mfcc12_var = np.var(librosa.feature.mfcc(y=audio_data, sr=sr)[11])
mfcc1_mean = np.mean(librosa.feature.mfcc(y=audio_data, sr=sr)[0])
features = np.array([perceptr_var, rolloff_var, spectral_bandwidth_mean, mfcc4_mean, mfcc7_var, chroma_stft_mean, rms_mean, mfcc1_mean, mfcc12_var, mfcc18_var, chroma_stft_var, mfcc20_mean, mfcc5_var, rms_var, harmony_var])
features = np.reshape(features, (1, -1))
df_features = pd.DataFrame(features, columns=["perceptr_var", "rolloff_var", "spectral_bandwidth_mean", "mfcc4_mean", "mfcc7_var", "chroma_stft_mean", "rms_mean", "mfcc1_mean", "mfcc12_var", "mfcc18_var", "chroma_stft_var", "mfcc20_mean", "mfcc5_var", "rms_var", "harmony_var"])
```

The prediction function uses the model of decision tree classifier with 15 features as it achieved the highest accuracy of 63 % .Here in the prediction function we are using a wav file and through librosa library we are extracting the desired features, Then we are creating a dataframe with the values and passing it to the model for prediction .

```
✓ [40] print(clf_final.predict(df_features))
0s ['reggae']
```

fig:Output of the pred function.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion:

The challenge of categorising music into different genres has always been difficult, and in recent years, a lot of work has been done in this field recently. In this study, musical genre classification results from that accessible dataset were evaluated using a variety of machine learning models. This project suggests using the Random Forest Classifier for modelling with the top three characteristics and proper hyperparameter tweaking. It showcases better performance in model training, as for validation set yielded results that were comparable when using Grid Search. Additionally, the results demonstrate a significant reduction in overall time and memory consumption in comparison of any other machine learning models that have been put into use. Experimenting with deep learning models on bigger datasets is one of the area's future directions. This project can be expanded to include the implementation of a system for making music recommendations based on the user's mood.

6.2 Future Scope:

The future directions and scope for the project "Comparison of Feature Selection Pipelines for Music Genre Classification" can include the following:

Exploration of other feature selection techniques:

The project can be extended to explore other feature selection techniques that are not covered in the project. Some of these techniques include Recursive Feature Elimination (RFE), feature scaling, and principal component analysis (PCA).

Comparison with deep learning models:

Deep learning models such as convolutional neural networks (CNN) and recurrent neural networks (RNN) can be used to compare their performance with the feature selection pipelines. This will provide insights into whether traditional feature selection techniques still have relevance in music genre classification.

Exploration of other music datasets: The project can be extended to explore other music datasets such as GTZAN dataset, MagnaTagATune dataset, and Million Song dataset. This will provide a more comprehensive understanding of the performance of different feature selection pipelines on different music datasets.

Real-time classification: The project can be extended to real-time classification of music genres by developing a web or mobile application that can classify music genres in real-time using the best-performing feature selection pipeline.

Transfer learning: Transfer learning can be applied to this project to improve the performance of the classification models. The pre-trained models from other domains such as speech recognition can be used as a starting point for the music genre classification models. These are just a few potential future directions and scope for the project, and there are many more possibilities depending on the interests and goals of the project.

References

- [1] "Music Genre Classification via Machine Learning", Category: Audio and Music Li Guo(liguo94), Zhiwei Gu(zhiweig), Tianchi Liu(kitliu5)
- [2] Beici Liang, QQ Music, Tencent Music Entertainment Shenzhen, China; Minwei Gu QQ Music, Tencent Music Entertainment Shenzhen, China : "Music Genre Classification Using Transfer Learning"
- [3]2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE 2021) 978-1-6654-1540-8/21/\$31.00 ©2021 IEEE 53 "Combined Transfer and Active Learning for High Accuracy Music Genre Classification Method" by Congyue Chen and Xin Steven
- [4] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR), 2017.
- [5] Raglio, Alfredo, Marcello Imbriani, Chiara Imbriani, Paola Baiardi, Sara Manzoni, Marta Gianotti, Mauro Castelli, Leonardo Vanneschi, Francisco Vico, and Luca Manzoni. "Machine learning techniques to predict the effectiveness of music therapy: A randomized controlled trial." Computer methods and programs in biomedicine 185 (2020): 105160.
- [6] Siddique, Md Abdus Salam, Md Imran Sarker, Robin Ghosh, and Kamal Gosh. "Toxicity Classification on Music Lyrics Using Machine Learning Algorithms." In 2021 24th International Conference on Computer and Information Technology (ICCIT), pp. 1-5. IEEE, 2021.

[7]Kaggle.com. 2022. Prediction of music genre. [online] Available at:
<<https://www.kaggle.com/vicsuperman/prediction-of-music-genre>>

TURNITIN PLAGIARISM REPORT
(This report is mandatory for all the projects and plagiarism must be below 25%)