

MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

HEMALATHA.V (Reg. No. 38290024)

VINOBA.S (Reg. No. 38290104)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

APRIL 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **HEMALATHA. V** (Reg. No. 38290024) and **VINOBA. S** (Reg. No. 38290104) who carried out the project entitled “**MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING**” under my supervision from November 2020 to April 2021.

Internal Guide

Dr. MERCY PAUL SELVAN, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **HEMALATHA.V (Reg. No. 38290024)** hereby declare that the Project Report entitled "**MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING**" done by me under the guidance of **Dr. MERCY PAUL SELVAN** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D.**, and **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. MERCY PAUL SELVAN, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways For the Completion of the project.

ABSTRACT

Music plays a very important role in people's lives. Music brings like-minded people together and is the glue that holds communities together. Communities can be recognized by the type of songs that they compose, or even listen to. The purpose of our project and research is to find a better machine learning algorithm than the pre-existing models that predicts the genre of songs.

Genres can be defined as categorical labels created by humans to identify or characterize the style of music. The concept of automatic music genre classification has become very popular in recent years as a result of the rapid growth of the digital entertainment industry.

This work presents a comprehensive machine learning approach to the problem of automatic musical genre classification using the audio signal. The system is developed using a KNN and SVM to recognize the genre.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 OVERVIEW OF PROJECT	1
2	LITERATURE SURVEY	2
3	AIM AND SCOPE OF PRESENT INVESTIGATION	5
	3.1 AIM OF THE PROJECT	5
	3.2 SCOPE AND OBJECTIVE	5
	3.3 SYSTEM REQUIREMENTS	6
	3.3.1 HARDWARE REQUIREMENTS	6
	3.3.2 SOFTWARE REQUIREMENTS	6
	3.4 SOFTWARE USED	7
	3.4.1 PYTHON LANGUAGE	7
	3.4.2 PYTHON CHARACTERISTICS	7
	3.4.3 APPLICATIONS OF PYTHON	7
	3.4.4 OPENCV PACKAGE	8
	3.5 ANACONDA NAVIGATOR	9
	3.5.1 ANACONDA	9
	3.5.2 APPLICATIONS IN ANACONDA	11
	3.5.3 VS CODE	12
	3.5.4 NEW FEATURES OF ANACONDA 5.3	12

4	EXPERIMENTAL OR MATERIAL METHODS	14
	4.1 DESIGN METHODOLOGY	14
	4.1.1 EXISTING SYSTEM	14
	4.1.2 PROPOSED SYSTEM	14
	4.2 MODULE DESCRIPTION	14
	4.2.1 WORKING OF SVM	15
	4.2.2 WORKING OF KNN ALGORITHM	17
	4.3 ARCHITECTURE DIAGRAM	18
	4.4 MODEL TRAINING	19
	4.4.1 TRAINED CLASSIFIER	19
	4.5 PREDICTION	20
5	RESULTS AND PERFORMANCE ANALYSIS	21
	5.1 UPLOADING THE AUDIO FILE	21
	5.2 TO PREDICT WHAT KIND OF GENRE	22
6	CONCLUSION AND FUTURE ENHANCEMENT	23
	6.1 CONCLUSION	23
	6.2 FUTURE ENHANCEMENT	23
	REFERENCES	24
	APPENDIX	26
	A. SOURCE CODE	26

LIST OF FIGURES

FIGURE NO:	FIGURE NAME	PAGE NO
3.1	Anaconda Distribution	10
3.2	Anaconda Navigator Home Page	11
4.1	SVM Graphical Representation	16
4.2	KNN Graphical Representation	17
4.3	Architecture Diagram	18
5.1	Home Page – Upload Page	21
5.2	After uploading the audio file – Predication Page	22

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW OF PROJECT

Music genres are a set of descriptive keywords that convey high-level information about a music clip (jazz, classical, rock...). Music classification is considered as a very challenging task due to selection and extraction of appropriate audio features. While unlabeled data is readily available music tracks with appropriate genre tags is very less. Genre classification is a task that aims to predict music genre using the audio signal. Being able to automatize the task of detecting musical tags allow to create interesting content for the user like music discovery and playlist creations, and for the content provider like music labeling and ordering.

Building this system requires extracting acoustic features that are good estimators of the type of genres we are interested, followed by a single or multi label classification or in some cases, regression stage. Conventionally, feature extraction relies on a signal processing front-end in order to compute relevant features from time or frequency domain audio representation. The features are then used as input to the machine learning stage.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of computer programs that can access data and use it learn for themselves. Machine learning algorithms are used to classify and predict the genre of a music.

CHAPTER 2

2. LITERATURE SURVEY

1. Music Genre Classification and Recommendation by Using Machine Learning Techniques

Authors: Ahmet Elbir ; Hilmi Bilal Çam ; Mehmet Emre Iyican ; Berkay Öztürk ; Nizamettin Aydın, 2018 Innovations in Intelligent Systems and Applications Conference (ASYU)

Music genre prediction is one of the topics that digital music processing is interested in. In this study, acoustic features of music have been extracted by using digital signal processing techniques and then music genre classification and music recommendations have been made by using machine learning methods. In addition, convolutional neural networks, which are deep learning methods, were used for genre classification and music recommendation and performance comparison of the obtained results has been. In the study, GTZAN database has been used and the highest success was obtained with the SVM algorithm.

2. Music Genre Classification: A N-Gram Based Musicological Approach

Authors: Eve Zheng ; Melody Moh ; Teng-Sheng Moh, 2017 IEEE 7th International Advance Computing Conference (IACC)

Digitalization of music has grown deep into people's daily life. Derived services of digital music, such as recommendation systems and similarity test, then become essential for online services and marketing essentials. As a building block of these systems, music genre classification is necessary to support all these services. Previously, researchers mostly focused on low-level features, few of them viewed this problem from a more interpretable way, i.e., a musicological approach. This creates the problem that intermediate stages of the classification process are hardly interpretable, not much of music professionals' domain knowledge was therefore useful in the process. This paper approaches genre classification in a musicological way. The proposed method takes into consideration the high-level features that have clear musical meanings, so that music professionals would find the classification results interpretable.

To examine more musicological elements other than additional statistical information, we use a dataset of only symbolic piano works, including more than 200 records of classical, jazz, and ragtime music. Feature extraction and n-gram text classification algorithm are performed. The proposed method proves its concept with experimental results achieving the prediction accuracy averaged above 90%, and with a peak of 98%.

3. Long short-term memory recurrent neural network-based segment features for music genre classification

Authors: Jia Dai ; Shan Liang ; Wei Xue ; Chongjia Ni ; Wenju Liu, 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)

In the conventional frame feature based music genre classification methods, the audio data is represented by independent frames and the sequential nature of audio is totally ignored. If the sequential knowledge is well modeled and combined, the classification performance can be significantly improved. The long short-term memory (LSTM) recurrent neural network (RNN) which uses a set of special memory cells to model for long-range feature sequence, has been successfully used for many sequence labeling and sequence prediction tasks. In this paper, we propose the LSTM RNN based segment features for music genre classification. The LSTM RNN is used to learn the representation of LSTM frame feature. The segment features are the statistics of frame features in each segment. Furthermore, the LSTM segment feature is combined with the segment representation of initial frame feature to obtain the fusional segment feature. The evaluation on ISMIR database show that the LSTM segment feature performs better than the frame feature. Overall, the fusional segment feature achieves 89.71% classification accuracy, about 4.19% improvement over the baseline model using deep neural network (DNN). This significant improvement shows the effectiveness of the proposed segment feature.

4. Music Genre Recognition Using Residual Neural Networks

Authors; Dipjyoti Bisharad ; Rabul Hussain Laskar

TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)

Genre is an abstract, yet a characteristic feature of music. Existing works for

automatic genre classification compute a set of features from the audio and design a classifier on top of it. Such models, in general, compute these features over a relatively long duration of the audio.

In this paper, a residual neural network-based model is proposed for genre classification which is trained on short clips of just 3 seconds duration. Also, traditional genre classification algorithms will assign a single genre to an audio clip. However, it well established that different genres have overlapping characteristics.

Considering this ambiguous nature of the genre, the model proposed in this work can assign three genre labels to a music clip, with each genre associated with some probability. The proposed model has an error rate of 18%, 9%, and 5.5% while predicting into top-1, top-2 and top-3 genres for a music clip respectively. We demonstrate in this work that the predictions made by the classifier align with the broader understood meaning of genre in a realistic setting.

5. On Combining Diverse Models for Lyrics-Based Music Genre Classification
Authors: Caio Luiggy Riyoichi Sawada Ueno ; Diego Furtado Silva, 2019 8th Brazilian Conference on Intelligent Systems (BRACIS)

Automatic music organization and retrieval is a highly required task nowadays. Labeling songs with summarized but descriptive information have implications in a wide range of tasks in this scenario. The genre is one of the most common labels used for music recordings. Using this piece of information, music platforms can organize collections by, for instance, associating songs and artists with similar characteristics. Lyrics represent an alternative source of data for genre recognition. While "traditional" bag-of-words-based text mining techniques represent a considerable part of the literature, recent papers shown an advance on this task applying deep learning algorithms. However, there is no research on how these distinct strategies contribute to each other. In this paper, we explore different strategies for music genre classification from lyrics and show that even simple combinations of these strategies allow improving accuracy on the lyrics-based music genre identification.

CHAPTER 3

3. AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 AIM OF THE PROJECT

Genre classification is an important task with many real world applications. to develop a deep learning project to automatically classify different musical genres from audio files. We will classify these audio files using their low-level features of frequency and time domain. we need a dataset of audio tracks having similar size and similar frequency range. GTZAN genre classification dataset is the most recommended dataset for the music genre classification

3.2 SCOPE AND OBJECTIVE

- Classification of genre can be very valuable to explain some interesting problems such as creating song references, tracking down related songs, discovering societies that will like that specific song, sometimes it can also be used for survey purposes.
 - Automatic musical genre classification can assist humans or even replace them in this process and would be of a very valuable addition to music information retrieval systems. In addition to this, automatic classification of music into genres can provide a framework for development and evaluation of features for any type of content-based analysis of musical signals
1. Developing a machine learning model that classifies music into genres shows that there exists a solution which automatically classifies music into its genres based on various different features, instead of manually entering the genre.
 2. Another objective is to reach a good accuracy so that the model classifies new music into its genre correctly
 3. This model should be better than at least a few preexisting models.

3.3 SYSTEM REQUIREMENTS

3.3.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. The minimal hardware requirements are as follows,

1. Processor : Pentium IV
2. RAM : 8 GB
3. Processor : 2.4 GHz
4. Main Memory : 8GB RAM
5. Hard Disk Drive : 1tb
6. Keyboard : 104 Keys

3.3.2 Software Requirements

Software requirements deals with defining resource requirements and prerequisites that needs to be installed on a computer to provide functioning of an application. The minimal software requirements are as follows,

1. Front end : python
2. Dataset : csv
3. IDE : anaconda
4. Operating System : Windows 10

3.4 SOFTWARE USED:

3.4.1 Python Language

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability & this class will teach you python from basics.

3.4.2 Python Characteristics

- It provides rich data types and easier to read syntax than any other programming languages
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Python are cross-platform compatible with Linux, Macintosh, and Windows
- For building large applications, Python can be compiled to byte-code
- Python supports functional and structured programming as well as OOP
- It supports interactive mode that allows interacting Testing and debugging of snippets of code
- In Python, since there is no compilation step, editing, debugging and testing is fast.

3.4.3 Applications of Python Programming

Web Applications:

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for

creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

Scientific and Numeric Computing:

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: Earthy for earth science, Astray for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

Creating software Prototypes:

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is a great language for creating prototypes. For example: You can use Pygmy (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

Good Language to Teach Programming:

Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest languages to learn because of its simple easy-to-use syntax.

3.4.4 OpenCV Package

Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python

wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of NumPy makes the task easier. NumPy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from NumPy arrays. So whatever operations you can do in NumPy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

3.5 ANACONDA NAVIGATOR:

3.5.1 Anaconda:

Anaconda is a free and open source, easy to install distribution of Python and R programming languages. Anaconda provides a working environment which is used for scientific computing, data science, statistical analysis and machine learning.

The latest distribution of Anaconda is Anaconda 5.3 and is released in October, 2018. It has the conda package, environment manager and a collection of 1000+ open source packages long with free community support.

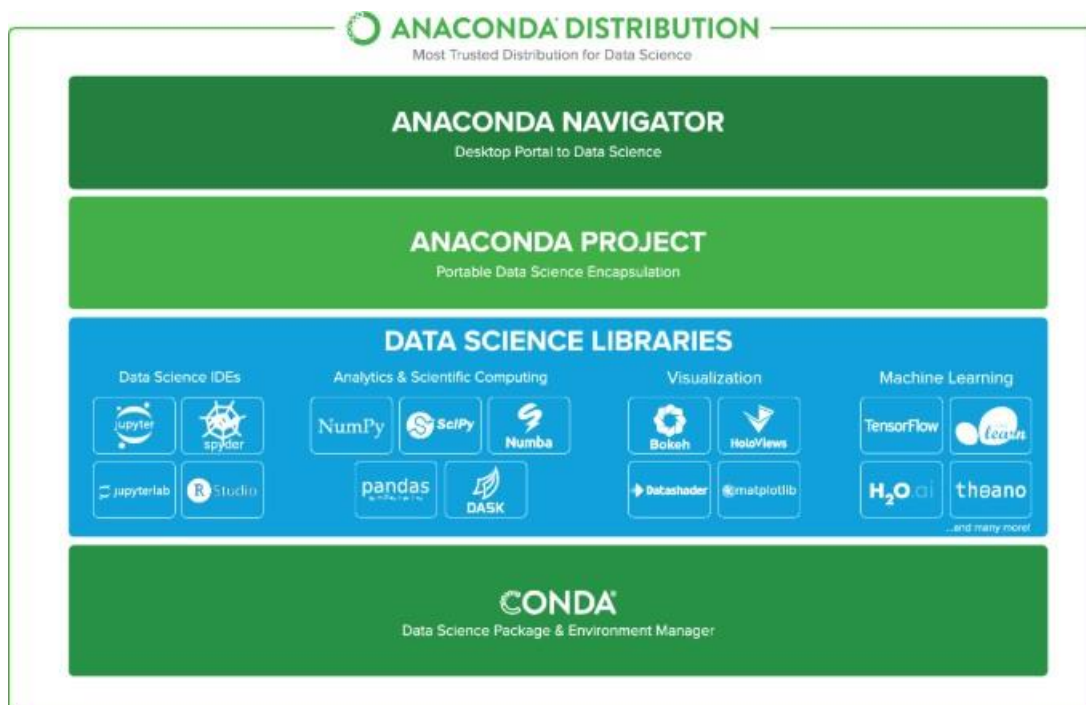


Fig 3.1: Anaconda Distribution

What is Anaconda Navigator?

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda distribution. It allows us to launch applications provided in the Anaconda distribution and easily manage conda packages, environments and channels without the use of command-line commands. It is available for Windows, macOS and Linux.

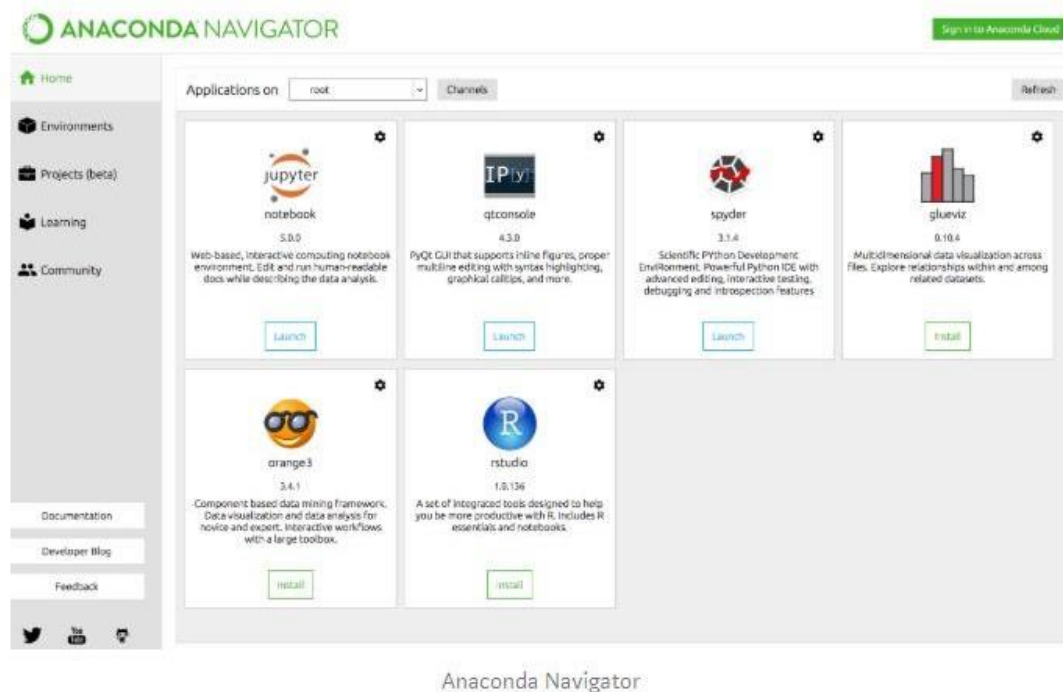


Fig 3.2: Anaconda Navigator Home Page

3.5.2 Applications Of Anaconda

The Anaconda distribution comes with the following applications along with Anaconda Navigator.

1. JupyterLab
2. Jupyter Notebook
3. Qt Console
4. Spyder
5. Glueviz
6. Orange3
7. RStudio

8. Visual Studio Code

3.5.3 VS Code:

VS Code is a free, open source streamlined cross-platform code editor with excellent support for Python code editing, IntelliSense, debugging, linting, version control, and more. Additionally, the Python Extension for Visual Studio Code tailors VS Code into a Python IDE. It is a streamlined code editor with support for development operations like debugging, task running and version control.

VS Code is free for both private and commercial use, runs on Windows, macOS, and Linux, and includes support for linting, debugging, task running, version control and Git integration, IntelliSense code completion, and conda environments. VS Code is openly extensible and many extensions are available.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

3.5.4 New Features of Anaconda 5.3

Compiled with Latest Python release: Anaconda 5.3 is compiled with Python 3.7, taking advantage of Python's speed and feature improvements.

- **Better Reliability:**

The reliability of Anaconda has been improved in the latest release by capturing and storing the package metadata for installed packages.

- **Enhanced CPU Performance:**

The Intel Math Kernel Library 2019 for Deep Neural Networks(MKL 2019) has been introduced in Anaconda 5.3 distribution. Users deploying

TensorFlow can make use of MKL 2019 for Deep Neural Networks. These Python binary packages are provided to achieve high CPU performance.

- **New packages are added:**

There are over 230 packages which has been updated and added in the new release.

- **Work in Progress:**

There is a casting bug in Numpy with Python 3.7 but the team is currently working on patching it until Numpy is updated

CHAPTER 4

4. EXPERIMENTAL OR MATERIAL METHODS

4.1 DESIGN METHODOLOGY

4.1.1 Existing System:

In existing system, we used k-nearest neighbor (k-NN) to classify the genres. This doesn't give an absolute reasonable correlation between learning strategies for classification of music genre. It uses filter modeling before Piecewise Gaussian Modeling. However, these improvements are not statistically significant. This procedure does not increase classification accuracy and it doesn't achieve the efficiency prediction.

4.1.2 Proposed System:

Support Vector Machine (SVMs) have been actively used for various music classification tasks such as music tag- ging , genre classification and user-item latent feature prediction for recommendation. SVMs and KNNs assume features that are in different levels of hierarchy and can be extracted by convolutional kernels. The hierarchical features are learned to achieve a given task during supervised training.

4.2 MODULE DESCRIPTION

Module 1: Data Preprocessing

Data Preprocessing is a process of transforming the raw, complex data into systematic understandable knowledge. It involves the process of finding out missing and redundant data in the dataset. Thus, this brings uniformity in the dataset. However, in our dataset, there was no missing values found meaning that every record was constituted its corresponding feature values.

Module 2: Feature Extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

Module 3: Model Creation:

Algorithm's Used:

- Support Vector Machine (SVM) algorithm
- K-nearest neighbors (KNN) algorithm

4.2.1 Working of SVM:

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

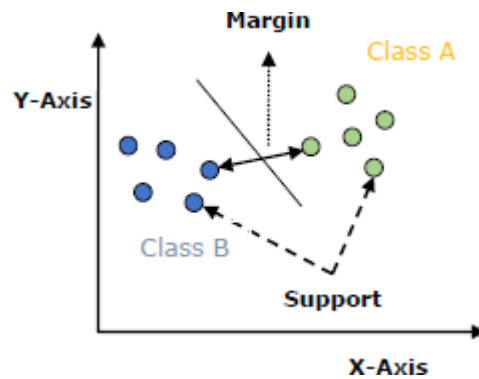


Fig 4.1: SVM Graphical Representation

The followings are important concepts in SVM :

Support Vectors:

Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

Hyperplane:

As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

Margin:

It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps:

First, SVM will generate hyperplanes iteratively that segregates the classes in best way.

Then, it will choose the hyperplane that separates the classes correctly.

4.2.2 Working of KNN Algorithm:

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps:

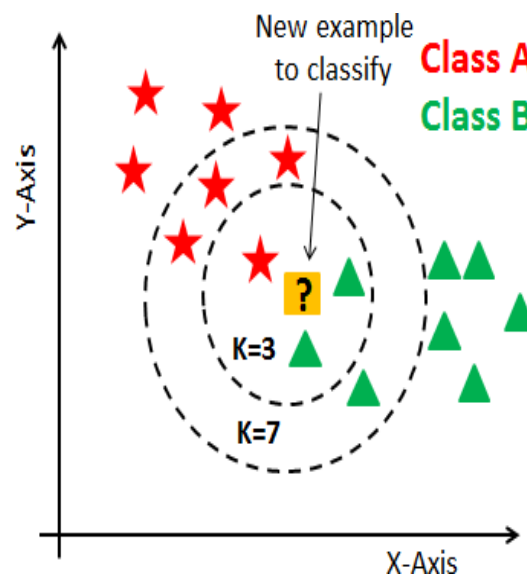


Fig 4.2: KNN Graphical Representation

Step 1:

For implementing any algorithm, we need dataset. So, during the first step of KNN, we must load the training as well as test data.

Step 2:

Next, we need to choose the value of K i.e., the nearest data points. K can be any integer.

Step 3:

For each point in the test data do the following

3.1 – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 – Now, based on the distance value, sort them in ascending order.

3.3 – Next, it will choose the top K rows from the sorted array.

3.4 – Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 – End

4.3 ARCHITECTURE DIAGRAM

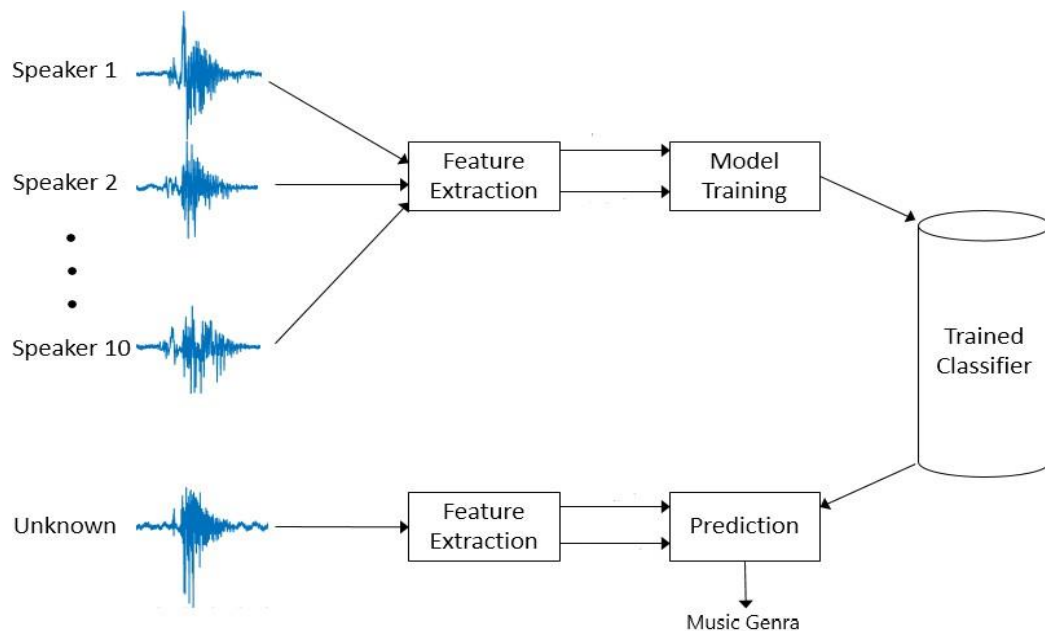


Fig 4.3: Architecture Diagram

Feature Extraction:

The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

4.4 Model Training:

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.

This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model.

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model.

4.4.1 Trained Classifier:

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

4.5 Prediction:

The song genre is predicted from the input we provided.

CHAPTER 5

5. RESULTS AND PERFORMANCE ANALYSIS

5.1. Uploading the audio file:

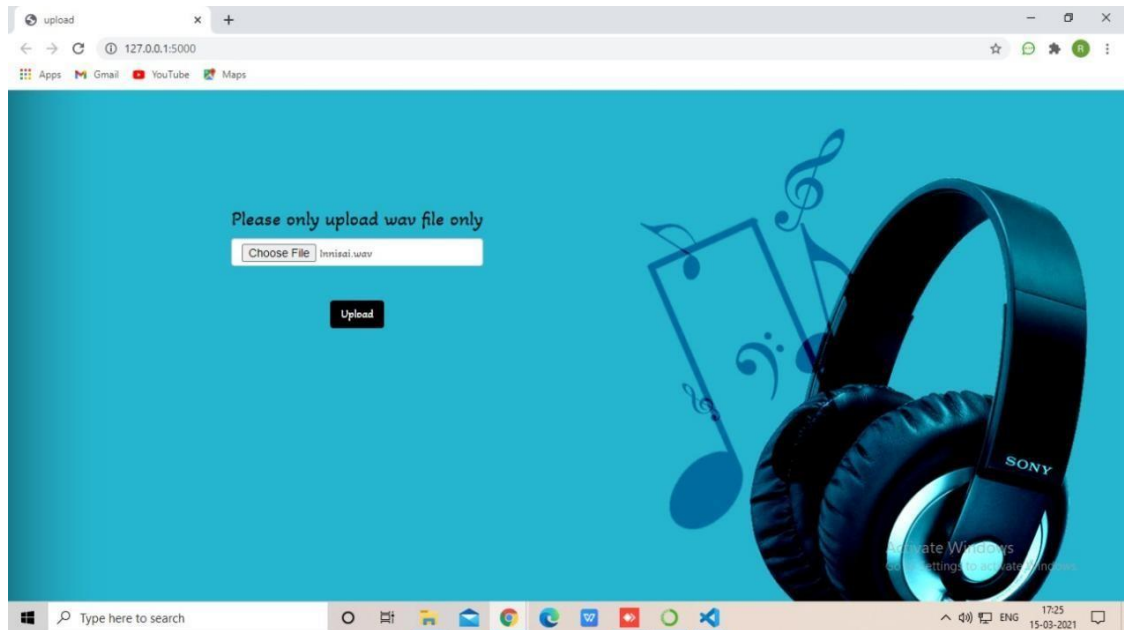


Fig 5.1: Home page -Upload Page

This is the home page or the main page of our project. It is the web page created using html. Here it provides an option to upload a audio file. The audio file should be in wav format because of its accurate and lossless format. The user can choose a file and easily upload using the upload button.

5.2. To predict what kind of genre:

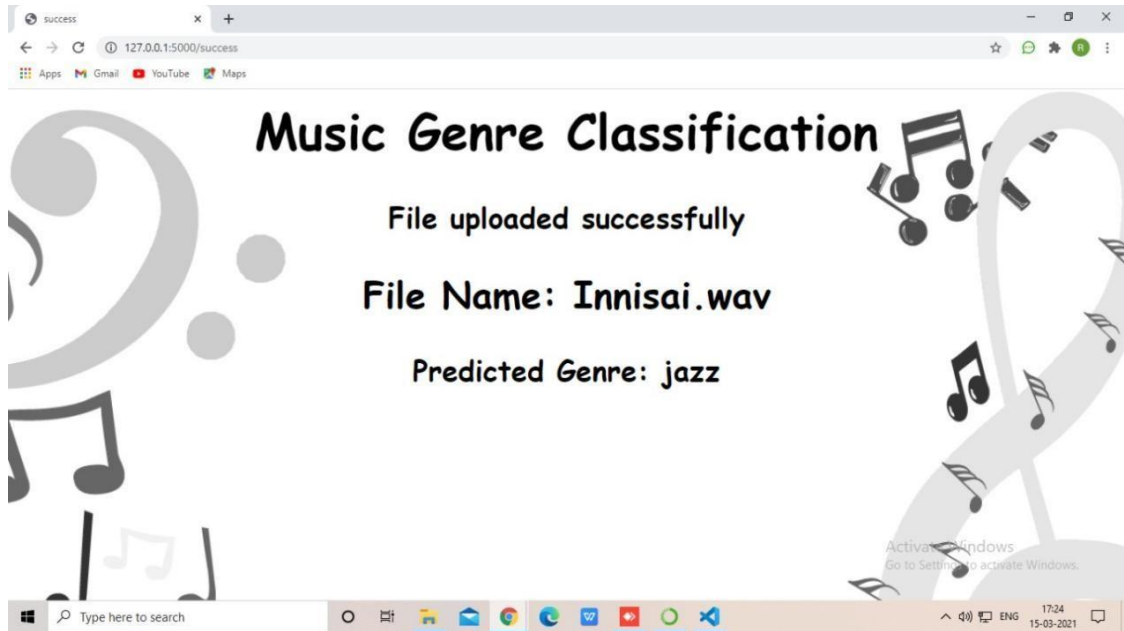


Fig 5.2: After uploading an audio file -Prediction Page

After the audio file is uploaded, within a few seconds it will display message that the file is uploaded successfully. It classifies the uploaded song file using machine learning techniques and predicts its genre with high accuracy. The Page provides the file name and its predicted genre.

CHAPTER 6

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION:

We would like to address the multi-class problem with an ensemble of Machine Learning methods that could combine to create a confidence vector for each input, such as using our SVM and KNN models together. This would expand our findings to be used for a streaming service that can input several different genres and have some interesting results. In addition, we would like to experiment with different types like svm and knn this expose different implicit features in the dataset

6.2 FUTURE ENHANCEMENTS:

In the future, we hope to experiment with other types of deep learning methods, given they performed the best. Given that this is time series data, some sort of RNN model may work well (GRU, LSTM, for example). We are also curious about generative aspects of this project, including some sort of genre conversion (in the same vein as generative adversarial networks which repaint photos in the style of Van Gogh, but for specifically for music). Additionally, we suspect that we may have opportunities for transfer learning, for example in classifying music by artist or by decade.

REFERENCES

- [1] Music Genre Classification Using Independent Recurrent Neural Network, Wenli Wu ; Fang Han ; Guangxiao Song ; Zhijie Wang, 2018 Chinese Automation Congress (CAC)
- [2] Music Genre Classification and Recommendation by Using Machine Learning Techniques, Ahmet Elbir ; Hilmi Bilal Çam ; Mehmet Emre Iyican ; Berkay Öztürk ; Nizamettin Aydin, 2018 Innovations in Intelligent Systems and Applications Conference (ASYU)
- [3] Improve Music Genre Classification with Conv Nets, Rafael L. Aguiar ; Yandre M.G. Costa ; Carlos N. Silla, 2018 International Joint Conference on Neural Networks (IJCNN)
- [4] Genre Classification using Word Embeddings and Deep Learning, Akshi Kumar ; Arjun Rajpal ; Dushyant Rathore, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)
- [5] Music Genre Classification: A N-Gram Based Musicological Approach, Eve Zheng ; Melody Moh ; Teng-Sheng Moh, 2017 IEEE 7th International Advance Computing Conference (IACC)
- [6] Long short-term memory recurrent neural network based segment features for music genre classification, Jia Dai ; Shan Liang ; Wei Xue ; Chongjia Ni ; Wenju Liu, 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)
- [7] Music Genre Recognition Using Residual Neural Networks, Dipjyoti Bisharad ; Rabul Hussain Laskar
TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)
- [8] Parallel Recurrent Convolutional Neural Networks Based Music Genre Classification Method for Mobile Devices, Rui Yang ; Lin Feng ; Huibing Wang ; Jianing Yao ; Sen Luo IEEE Access Year: 2020

- [9] On Combining Diverse Models for Lyrics-Based Music Genre Classification ,Caio Luiggy Riyoichi Sawada Ueno ; Diego Furtado Silva, 2019 8th Brazilian Conference on Intelligent Systems (BRACIS)
- [10] Automatic Classification of Music Genre Using Masked Conditional Neural Networks, Fady Medhat ; David Chesmore ; John Robinson, 2017 IEEE International Conference on Data Mining (ICDM)

APPENDIX

A. SOURCE CODE

MACHINE TRAINING CODE:

```
import pickle
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from matplotlib import cm
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVC

df = pd.read_csv('data.csv')
df['class_name'] = df['class_name'].astype('category')
df['class_label'] = df['class_name'].cat.codes

lookup_genre_name = dict(zip(df.class_label.unique(),
df.class_name.unique()))

cols = list(df.columns)
cols.remove('label')
cols.remove('class_label')
cols.remove('class_name')

X = df.iloc[:,1:28]
y = df['class_label']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

```

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
# we must apply the scaling to the test set that we computed for the training
set
X_test_scaled = scaler.transform(X_test)

pickle.dump(scaler,open('scaler.pkl','wb'))

knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(X_train_scaled, y_train)
#print(knn.score(X_test_scaled, y_test))
pickle.dump(knn, open('model_knn.pkl','wb'))

svm = SVC(kernel='linear', C = 10).fit(X_train_scaled, y_train)
pickle.dump(knn, open('model_cnn.pkl','wb'))

```

CODE FOR PREDICTION:

```

from flask import *
import pickle
import librosa
import numpy as np
from sklearn.preprocessing import MinMaxScaler

app = Flask(__name__,template_folder='template')

model = pickle.load(open('model_cnn.pkl','rb'))
scaler = pickle.load(open('scaler.pkl','rb'))

def getmetadata(filename):
    y, sr = librosa.load(filename)
    onset_env = librosa.onset.onset_strength(y, sr)

```

```

tempo = librosa.beat.tempo(onset_envelope=onset_env, sr=sr)

y_harmonic, y_percussive = librosa.effects.hpss(y)
tempo, beat_frames = librosa.beat.beat_track(y=y_percussive, sr=sr)

#chroma_stf
chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
#rmse
rmse = librosa.feature.rms(y=y)
#fetching spectral centroid
spec_centroid = librosa.feature.spectral_centroid(y, sr=sr)[0]
#spectral bandwidth
spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
#fetching spectral rolloff
spec_rolloff = librosa.feature.spectral_rolloff(y+0.01, sr=sr)[0]
#zero crossing rate
zero_crossing = librosa.feature.zero_crossing_rate(y)
#mfcc
mfcc = librosa.feature.mfcc(y=y, sr=sr)
#metadata dictionary
metadata_dict =
{'tempo':tempo,'chroma_stft':np.mean(chroma_stft),'rmse':np.mean(rmse),
'spectral_centroid':np.mean(spec_centroid),'spectral_bandwidth':np.mean(spe
c_bw),
        'rolloff':np.mean(spec_rolloff),
'zero_crossing_rates':np.mean(zero_crossing)}

for i in range(1,21):
    metadata_dict.update({'mfcc'+str(i):np.mean(mfcc[i-1])})

return (metadata_dict)

@app.route('/')
def upload():

```

```

return render_template("file_upload_form.html")

@app.route('/success', methods = ['POST'])
def success():
    if request.method == 'POST':
        f = request.files['file']
        f.save(f.filename)
        f_name = f.filename.split('.')

        if(f_name[-1] != 'wav'):
            file_name = 'Upload only wav file'
        else:
            file_name = f.filename

        genre = {0: 'blues', 1: 'classical', 2: 'country', 3: 'disco', 4: 'hiphop', 5: 'jazz',
6: 'metal', 7: 'pop', 8: 'reggae',9: 'rock'}

        mtdt = getmetadata(f.filename)
        mtdt = np.array(list(mtdt.values()))
        mtdt.reshape(-1,1)
        mtdt_scaled = scaler.transform([mtdt])
        pred_genre = model.predict(mtdt_scaled)
        genre_name = genre[pred_genre[0]]
        return render_template("success.html", name = file_name, genre =
genre_name)

if __name__ == '__main__':
    app.run(debug = True)

```

HTML CODE:

FILE UPLOAD PAGE:

```

<html>
<head>
<title>upload</title>

```

```

<link rel="preconnect" href="https://fonts.gstatic.com">
<link
href="https://fonts.googleapis.com/css2?family=Akaya+Telivigala&display=swap"
rel="stylesheet">
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script
>
  <style>
    .music{
      position: absolute;
      top:20%;
      left:20%;
    }
    body{
      /* background:green; */
      background-image: url('/static/1.png');
      background-size: cover;
      background-repeat: no-repeat;
      color:black;
      font-family: 'Akaya Telivigala', cursive;
      text-align:center;
      font-size: 30px;
    }
    .btn:hover{
      background-color: black;
      color:white;

```

```

</style>
</head>
<body>
  <div class="music">
    <div class="form-group">
      <h3> Please only upload wav file only </h3>
      <form action = "/success" method = "post" enctype="multipart/form-data">
        <input type="file" name="file" class="form-control"/><br>
        <input type = "submit" class="btn" value="Upload">
      </form>
    </div>
  </div>
</body>
</html>

```

PREDICTION PAGE:

```

<html>
<head>
<title>success</title>
<link rel="preconnect" href="https://fonts.gstatic.com">
<link
href="https://fonts.googleapis.com/css2?family=Josefin+Sans:wght@100&display=
swap" rel="stylesheet">
<style>
body{
  /* background:green; */
  background-image: url('/static/2.png');
  background-size: cover;
  background-repeat: no-repeat;
  color:black;

```

```
font-family: 'Akaya Telivigala', cursive;
text-align:center;
font-size: 30px;
}
</style>

</head>
<body>
    <center><h1>Music Genre Classification</h1></center>
    <center><h3>File uploaded successfully</h3></center>
    <center><h2>File Name: {{name}}</h2></center>
    <center><h3>Predicted Genre: {{genre}}</h3></center>
</body>
</html>
```