

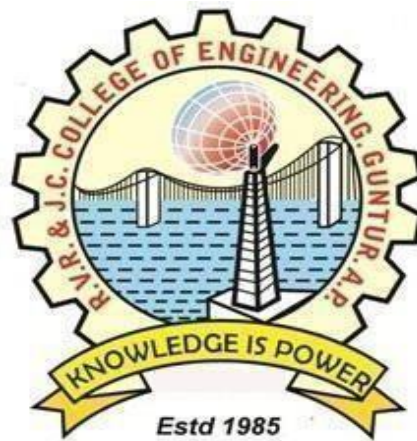
An Internship Report On
LANGUAGE DETECTION USING NATURAL LANGUAGE
PROCESSING

Submitted in partial fulfillment of requirements

For the Internship (IT-353)

Submitted By

I. AVINASH (Y20IT041)



February-2023

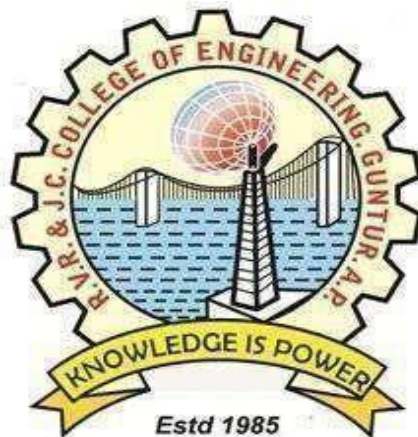
R.V.R&J.C. COLLEGE OF ENGINEERING

(Autonomous)

Approved by AICTE:: Affiliated to Acharya Nagarjuna University

Chowdavaram, GUNTUR – 522019, Andhra Pradesh, India

DEPARTMENT OF INFORMATION TECHNOLOGY
R.V.R&J.C. COLLEGE OF ENGINEERING (Autonomous)
GUNTUR-522019



BONAFIDE CERTIFICATE

This is to certify that this internship report “**LANGUAGE DETECTION USING NATURAL LANGUAGE PROCESSING**” is the bonafide work of “**IMMADISETTY AVINASH (Y20IT041)**” who have carried out the work under my supervision, and submitted in partial fulfillment for the award of **INTERNSHIP(IT-353)** during the year 2022-2023.

Smt. B. Manasa
Assistant Professor, IT

Dr.A.Srikrishna
Prof.& HOD, Department of IT

INTERNSHIP CERTIFICATE

IndianServers - A Premier Software Development Company

www.IndianServers.com

Indian Servers

Vijayawada,
Dt..25-06-2022

Internship Completion Certificate

This is to certify that **Ms. Batchu Yasaswini** Successfully completed "8+ weeks Internship" at **Indian Servers Private Limited**, Vijayawada from 24-Apr-2022 to 25-Jun-2022, as "**Trainee - Machine Learning**".

During the period of Internship Program with us, Batchu Yasaswini has been exposed to various concepts like *Basic Python Programming, Statistics Machine Learning Algorithms (Supervised and Unsupervised), basics of Deep Learning using Keras, OpenCV, basics of transfer learning.*

During this period the contribution of Batchu Yasaswini in development of our company projects using Machine Learning is "Good".

UID: **INDSER2022ML237**

This certificate can be verified on request



With warm regards,

Sai Satish
D. Sai Satish
CEO, Indian Servers

Indian Servers Private Limited
(web for All)
Estd: 2008
Andhra Pradesh
www.IndianServers.com

info@indianservers.com
Mobile : 9618222220

ACKNOWLEDGEMENT

The successful completion of any task would be in complete without a proper suggestions, guidance and environment. The combination of these three factors acts like back bone to our internship “**LANGUAGE DETECTION USING NATURAL LANGUAGE PROCESSING**”.

I would like to express our gratitude to the Management of **R.V.R&J.C.COLLEGE OF ENGINEERING** for providing us with a pleasant environment and excellent lab facility.

I regard my sincere thanks to our Principal, **Dr.Kolla Srinivas** for providing support and stimulating environment.

I am greatly indebted to **Dr.A.Srikrishna**, Professor, Head of the Department Information Technology, for her valuable suggestions during the internship.

I would like to express our special thanks to our guide **Smt. B. Manasa**, Assistant Professor who helped us in doing the internship successfully.

IMMADISETTY AVINASH
(Y20IT041)

ABSTRACT

Language identification (“LI”) is the problem of determining the natural language that a document or part thereof is written in. Automatic LI has been extensively researched for over fifty years. Today, LI is a key part of many text processing pipelines, as text processing techniques generally assume that the language of the input text is known. Research in this area has recently been especially active. This article provides a brief history of LI research, and an extensive survey of the features and methods used in the LI literature. We describe the features and methods using a unified notation, to make the relationships between methods clearer. We discuss evaluation methods, applications of LI, as well as off-the-shelf LI systems that do not require training by the end user. Finally, we identify open issues, survey the work to date on each issue, and propose future directions for research in LI.

CONTENTS

Title	i
Certificate	iii
Acknowledgement	iv
Abstract	v
Contents	vi
List of Figures	vii
List of Abbreviations	viii
1.Introduction	
1.1 Background	1
1.2 Significance of the work	3
2.System Analysis	
2.1 Requirements Specification	5
2.1.1Functional Requirements	5
2.1.2 System Requirements	6
2.1.3 Non-Functional Requirements	6
3.System Design	
3.1 Architecture of the Proposed System	7
3.2 Flow of Work	19
4.Implementation and Testing	20
5.Results	24
6.Conclusion and future work	25
7.References	26

LIST OF FIGURES

Fig.No	Name	PageNo
1.1	Structure of Neuron	2
3.1.1	System Architecture	7
3.1.2	Sample of acquired music dataset	8
3.1.3	Reading the dataset from the CSV file into notebook	9
3.1.4	Dropping unnecessary features from data frame	10
3.1.5	Sample data	11
3.1.6	Process of feature selection	11
3.1.7	Splitting dataset into training data and test data	12
3.1.8	Understanding audio	12
3.1.9	Importing Audio	13
3.1.10	Plot Raw Wave Files	14
3.1.11	Displaying Wave Plot	14
3.1.12	Displaying Specshow	15
3.1.13	Spectral Rolloff	16
3.1.14	Chroma Feature	16
3.1.15	Zero Crossing rate	17
3.1.16	Standardizing dataset	18
4.1	Training of data	20
4.1	Model Summary	21
4.2	Sample Code for building and testing the model	22
4.3	Comparison with other algorithms	23
5.1	Test Accuracy	24

LIST OF ABBREVIATIONS

CNN Convolutional Neural Networks

CSV Comma Separated Value

HTML Hyper Text Markup Language

CSS Cascading Style Sheets

1.INTRODUCTION

1.1 Background

Every Machine Learning enthusiast has a dream of building/working on a cool project, isn't it? Mere understandings of the theory aren't enough, you need to work on projects, try to deploy them, and learn from them. Moreover, working on specific domains like NLP gives you wide opportunities and problem statements to explore. Through this article, I wish to introduce you to an amazing project, the Language Detection model using Natural Language Processing. This will take you through a real-world example of ML(application to say). So, let's not wait anymore.

About the dataset

We are using the [Language Detection dataset](#), which contains text details for 17 different languages.

Languages are:

- English
- Portuguese
- French
- Greek
- Dutch
- Spanish
- Japanese
- Russian

- Danish
- Italian
- Turkish
- Swedish
- Arabic
- Malayalam
- Hindi
- Tamil
- Telugu

Using the text we have to create a model which will be able to predict the given language. This is a solution for many artificial intelligence applications and computational linguists. These kinds of prediction systems are widely used in electronic devices such as mobiles, laptops, etc for machine translation, and also on robots. It helps in tracking and identifying multilingual documents too. The domain of NLP is still a lively area of researchers.

Significance of work

2.SYSTEM ANALYSIS

2.1 RequirementSpecification

Requirements analysis, also called requirements engineering is the process of determining user expectations for a new or modified product. These features called requirements must be quantifiable, relevant and detailed.

In software engineering, such requirements are often called functional specifications. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

2.1.1 Functional Requirements

A Functional Requirement may be a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software, its behaviour, and outputs. It is often a calculation, data manipulation, business process, user interaction, or the other specific functionality which defines what function a system is probably going to perform. Functional Requirements describe the interactions between the system and its environment independent of its application.

- Applying the algorithms on the test data.
- Display the result with the description of which genre it belongs to.
- Execution time should be fast
- More Accurate
- User Friendly

2.1.2 System Requirements

Software Requirements:

Operating System : windows XP, Windows 7, 8.1,10,11

Coding languages: PYTHON , HTML ,CSS

Tools : Google Chrome

Software : Google Colab

HardwareRequirements:

Personal computer with keyboard and mouse maintained with uninterrupted power supply.

- Processor: Intel® core™ i5
- Installed Memory (RAM): 8.00 GB 6
- Hard Disc: 250GB SSD

2.1.3 Non-functional requirements:

Non-Functional Requirements specifies the standard attribute of a software. They judge the software supported Responsiveness, Usability, Security, Portability, and other 34 non-functional standards that are critical to the success of the software.

Non-functional Requirements allow you to impose constraints or restrictions on the planning of the system across the varied agile backlogs

- Accuracy
- Reliability
- Flexibility

3.SYSTEM DESIGN

3.1 Architecture of proposed system

Machine learning has given computer systems the ability to automatically learn without being explicitly programmed. In this, the author has used machine learning algorithm (CNN).

The following is the step by step working architecture of proposed System:

- Music Dataset
- Pre-processing data
- Training data
- Apply Machine Learning Algorithms
- CNN
- Testing Data

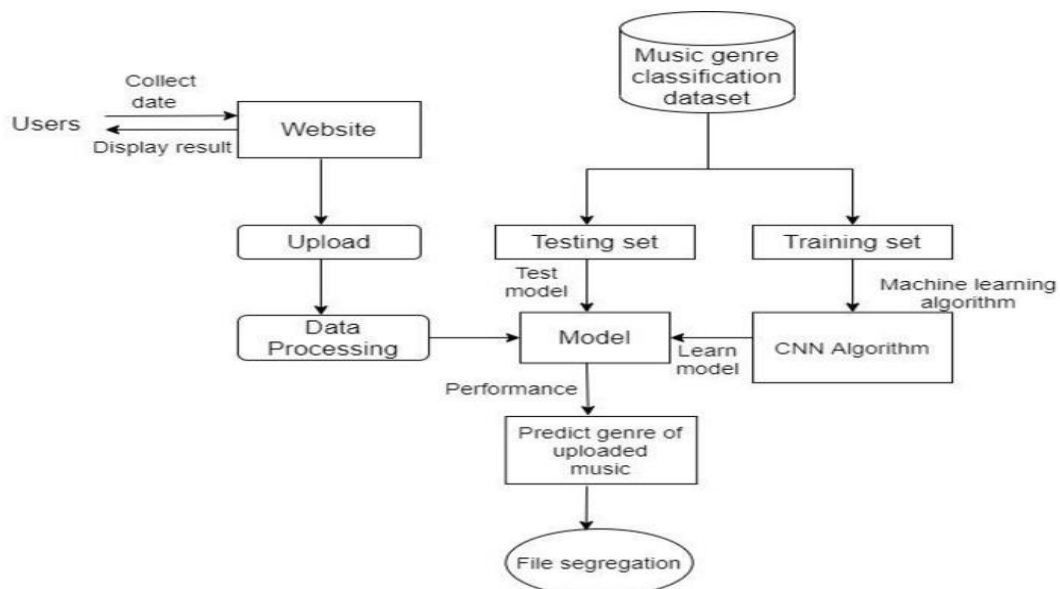


Figure 3.1.1 System Architecture

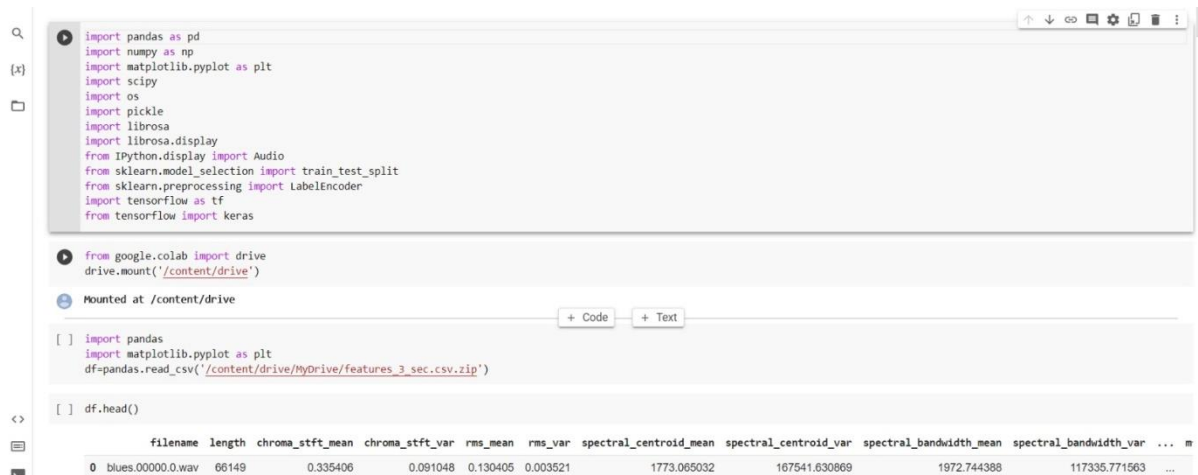
Working of Proposed System:

Music Dataset:

The main aim of this step is to spot and acquire all data-related problems. The number and quality of the collected data will determine the efficiency of the output. The more are going to be the info, the more accurate are going to be the prediction.

This acquired dataset has around 9990 music data and each row has 60 different music features.

Fig-3.1.2 Sample of acquired music dataset



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
import os
import pickle
import librosa
import librosa.display
from IPython.display import Audio
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow import keras

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] import pandas
import matplotlib.pyplot as plt
df=pandas.read_csv('/content/drive/MyDrive/features_3_sec.csv.zip')

[ ] df.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	...	m
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388	117335.771563	...	m

Fig-3.1.3 Reading the dataset from the CSV file into notebook

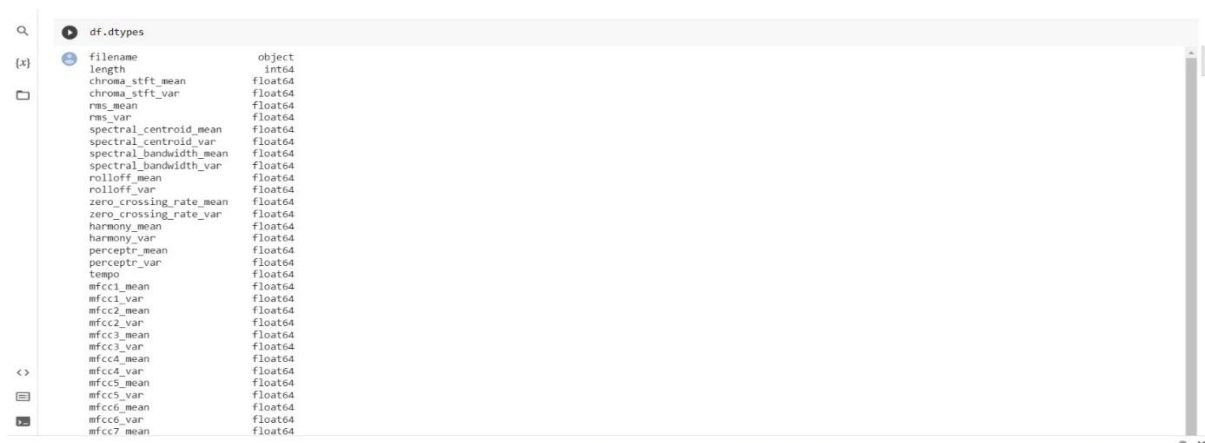
The dataset we chose is in the form of CSV (Comma Separated Value) file. After acquiring the data our next step is to read the data from the CSV file into the Google Colab also called a Python notebook.

Data Pre-Processing:

The main aim of this step is to study and understand the nature of data that was acquired in the previous step and also to know the quality of data. A real-world data generally contains noises, missing values, and maybe in an unusable format that cannot be directly used for machine learning models. Data pre-processing is a required task for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. Identifying duplicates in the dataset and removing them is also done in this step. When we check the correlation of the features, some of them are the same.

For data processing and features extraction, we used one of the very popular python library for audio analysis, Librosa. The Librosa library has all the necessary tools and techniques for the data extraction from any audio sample. The tasks of extracting the data can be achieved using many inbuilt functions in Librosa library. It extracts all the necessary data for training and classification.

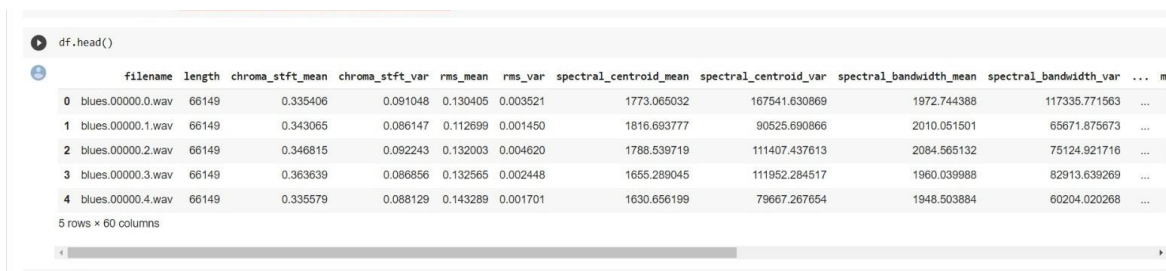
We have to extract relevant characteristics out of many, which are necessary to solve our problem. The process of extraction of features from the audio samples is called feature extraction. Some of the features are studied below. Zero-Crossing Rate: It's the measure of rate of sign changes along the signal. It is the rate at which the graph moves from negative to positive or back. The feature is widely used in speech recognition and now to classify the audio samples. Spectral Centroid: It calculates the weighted mean of the frequencies in the sound and locates the 'center of mass' of the sound. For example, consider a song of genre metal which has more frequencies towards the end compared to a song belonging to the blues genre whose frequencies are same throughout the length. So the spectral centroid for the blues genre will be somewhere in the middle and for the metal, it will be in the end. Spectral Roll-off: It measures the shape of a signal.



Feature	dtype
filename	object
length	int64
chroma_stft_mean	float64
chroma_stft_var	float64
rms_mean	float64
rms_var	float64
spectral_centroid_mean	float64
spectral_centroid_var	float64
spectral_bandwidth_mean	float64
spectral_bandwidth_var	float64
rolloff_mean	float64
rolloff_var	float64
zero_crossing_rate_mean	float64
zero_crossing_rate_var	float64
harmony_mean	float64
harmony_var	float64
perceptr_mean	float64
perceptr_var	float64
tempo	float64
mfcc1_mean	float64
mfcc1_var	float64
mfcc2_mean	float64
mfcc2_var	float64
mfcc3_mean	float64
mfcc3_var	float64
mfcc4_mean	float64
mfcc4_var	float64
mfcc5_mean	float64
mfcc5_var	float64
mfcc6_mean	float64
mfcc6_var	float64
mfcc7_mean	float64

Fig-3.1.4 Dropping unnecessary features from data frame

After pre-processing the acquired data, the next step is to identify the best features. The identified best features should be able to give high efficiency. In Fig 3.1.5, a screenshot of our notebook is shown how to select k best features using sklearn. The classes within the sklearn.feature_selection module are often used for feature selection/dimensionality reduction on sample sets, either to enhance estimators' accuracy scores or to spice up their performance on very high-dimensional datasets.

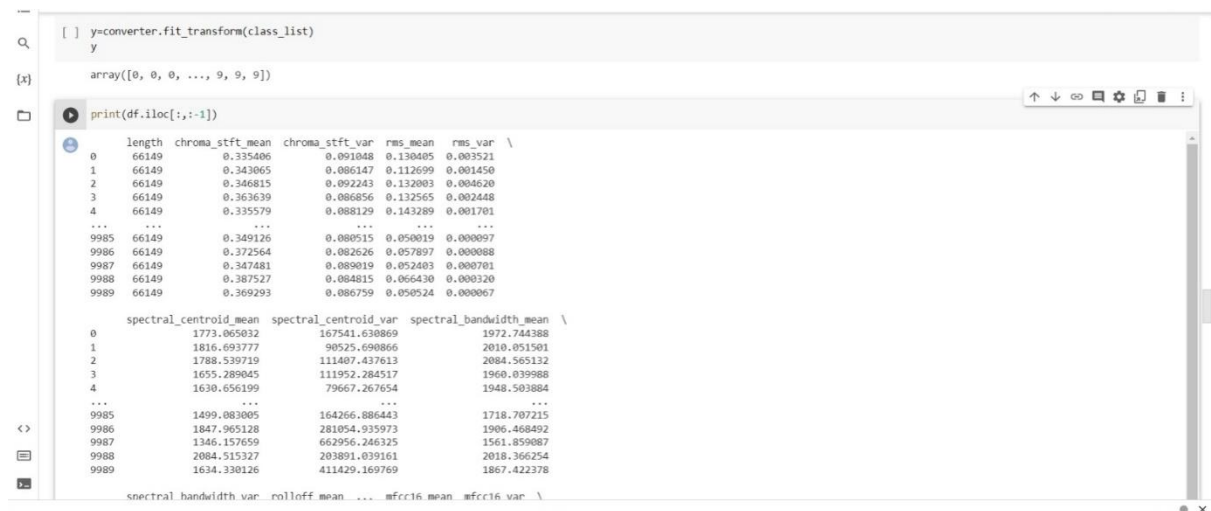


df.head()

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	...	m
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388	117335.771563	...	
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866	2010.051501	65671.875673	...	
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613	2084.565132	75124.921716	...	
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517	1960.039988	82913.639269	...	
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654	1948.503884	60204.020268	...	

5 rows x 60 columns

Fig-3.1.5 Sample data



```
[ ] y=converter.fit_transform(class_list)
y
array([[0, 0, 0, ..., 9, 9, 9]])
```

```
[x] print(df.iloc[:, :-1])
```

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean
0	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388
1	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866	2010.051501
2	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613	2084.565132
3	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517	1960.039988
4	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654	1948.503884
...
9985	66149	0.349126	0.089515	0.050019	0.000097	1499.083005	164266.086443	1718.707215
9986	66149	0.372564	0.082626	0.057897	0.000088	1847.965128	281054.935973	1906.468492
9987	66149	0.347481	0.089019	0.052403	0.000701	1346.157659	662956.246325	1561.859087
9988	66149	0.387527	0.084815	0.066430	0.000320	2084.515327	203891.039161	2018.366254
9989	66149	0.369293	0.086759	0.050524	0.000067	1634.330126	411429.169769	1867.422378

Fig-3.1.6 Process of Feature selection

Training data:

Splitting the dataset into Training set and testing set:

In machine learning data pre-processing, we have to break our dataset into both training set and test set. This is often one among the crucial steps of knowledge pre-processing as by doing this, we will enhance the performance of our machine learning model.

```
[ ] from sklearn.preprocessing import StandardScaler
    fit=StandardScaler()
    x=fit.fit_transform(np.array(df.iloc[:, :-1], dtype=float))

[ ] X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33)

len(y_test)

3297

[ ] len(y_train)

6693
```

Fig-3.1.7 Splitting dataset into training data and test data

Splitting the dataset into train and test in the ratio of 66:32 i.e., 66 percent of data is used for training and 32 percent of data is used for testing the model.

Apply Machine Learning Algorithms:

Now, we've both the train and test data. The subsequent step is to spot the possible training methods and train our models. As this is often a classification problem, we've used three different classification methods CNN. The algorithm has been run over the Training dataset and their performance in terms of accuracy is evaluated alongside the prediction wiped out the testing data set.

Understanding the Audio Files:

```
[x] [ ] df=df.drop(labels="filename",axis=1)

import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile
data,sr = librosa.load('drive/MyDrive/country.aac')
print(type(data),type(sr))

/usr/local/lib/python3.8/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
warnings.warn("PySoundFile failed. Trying audioread instead.")
<class 'numpy.ndarray'> <class 'int'>

[ ] librosa.load('drive/MyDrive/country.aac',sr=45600)

/usr/local/lib/python3.8/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
warnings.warn("PySoundFile failed. Trying audioread instead.")
(array([0., 0., 0., ..., 0.15360294, 0.09140505,
0., ], dtype=float32), 45600)

[ ] import IPython
IPython.display.Audio(data,rate=sr)
```

Fig-3.1.8. Understanding Audio

Through this code:

```
data, sr = librosa.load(audio_recording)
```

It loads and decodes the audio as a time series y . sr = sampling rate of y . It is the number of samples per second. 20 kHz is the audible range for human beings. So it is used as the default value for sr . In this code we are using sr as 45600Hz.

Audio Libraries Used:

1. Librosa

Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. By using Librosa, we can extract certain key features from the audio samples such as Tempo, Chroma Energy Normalized, Mel-Frequency Cepstral Coefficients, Spectral Centroid, Spectral Contrast, Spectral Rolloff, and Zero Crossing Rate.

2. IPython.display.Audio

With the help of **IPython.display.Audio** we can play audio in the notebook. It is a library used for playing the audio in the jupyterlab. The code is below:

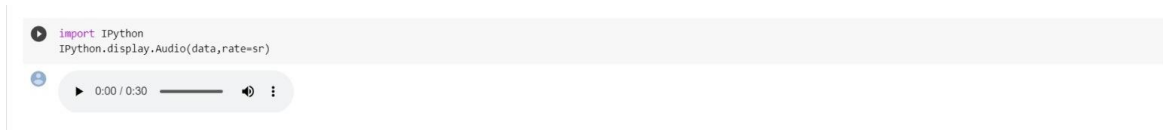


Fig-3.1.9. Importing Audio

Visualizing Audio Files

The following methods are used for visualizing the audio files we have:

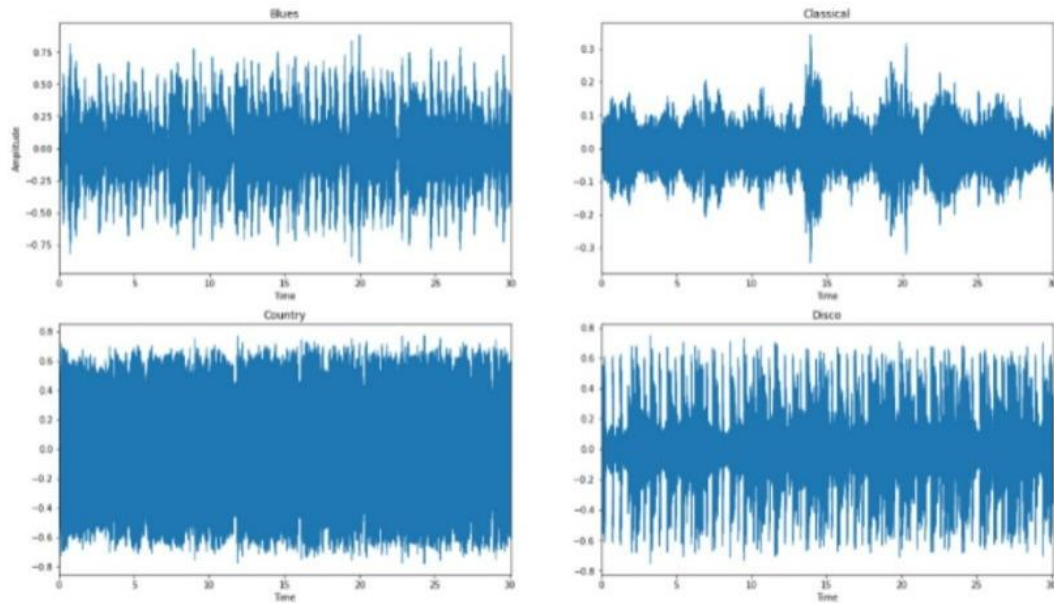


Fig-3.1.10. Plot Raw Wave Files

Waveforms are visual representations of sound as time on the x-axis and amplitude on the y-axis. They are great for allowing us to quickly scan the audio data and visually compare and contrast which genres might be more similar than others.

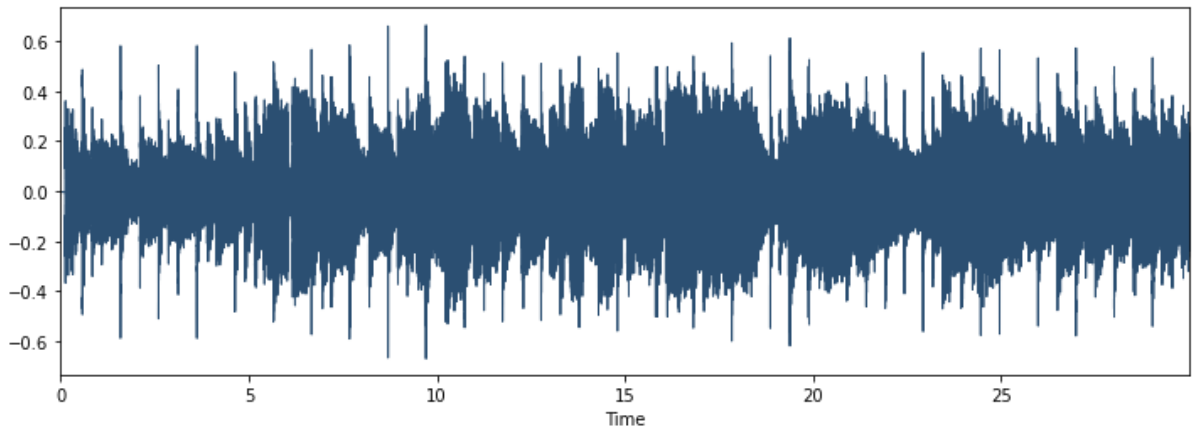


Fig-3.1.11. Displaying WavePlot

Spectograms:

A spectrogram is a visual way of representing the signal loudness of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time.

Spectrograms are sometimes called sonographs, voiceprints, or voicegrams. When the data is represented in a 3D plot, they may be called waterfalls. In 2-dimensional arrays, the first axis is frequency while the second axis is time.

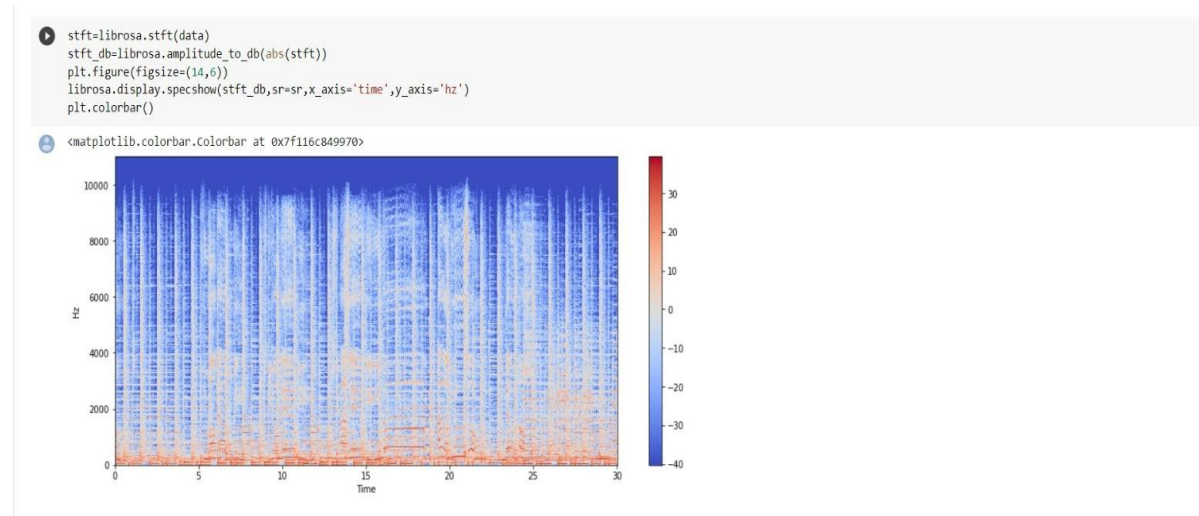


Fig-3.1.12. Displaying Specshow

Spectral Rolloff:

Spectral Rolloff is the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies

librosa.feature.spectral_rolloff computes the rolloff frequency for each frame in a signal.

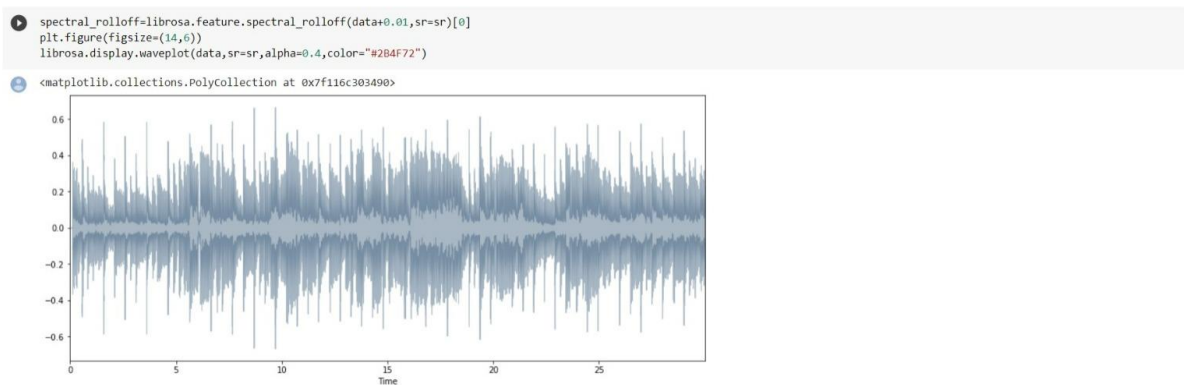


Fig-3.1.13. Spectral Rolloff

Chroma Feature:

It is a powerful tool for analyzing music features whose pitches can be meaningfully categorized and whose tuning approximates to the equal-tempered scale. One main property of chroma features is that they capture harmonic and melodic characteristics of music while being robust to changes in timbre and instrumentation.

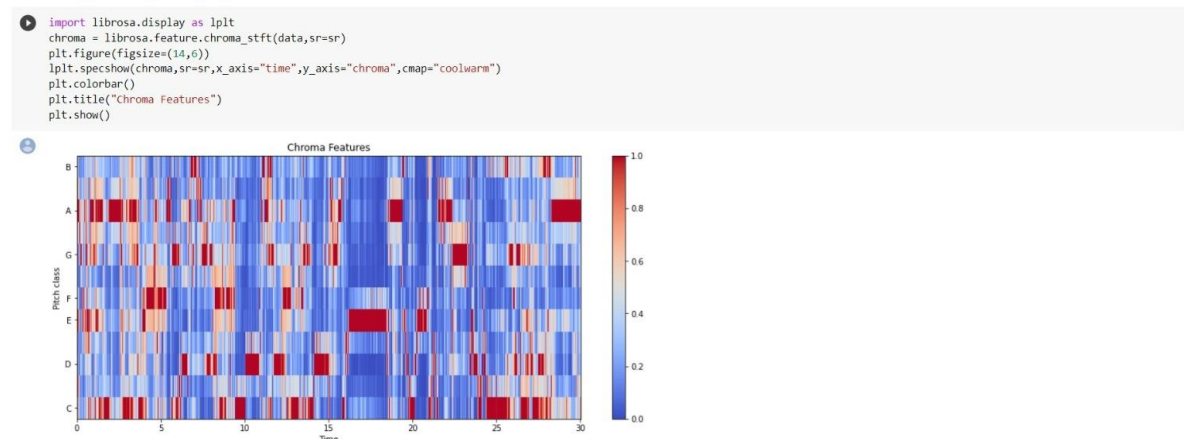


Fig-3.1.14. Chroma Feature

Zero Crossing Rate:

Zero crossing is said to occur if successive samples have different algebraic signs. The rate at which zero-crossings occur is a simple measure of the frequency content of a signal. Zero-crossing rate is a measure of the number of times in a given time interval/frame that the amplitude of the speech signals passes through a value of zero.

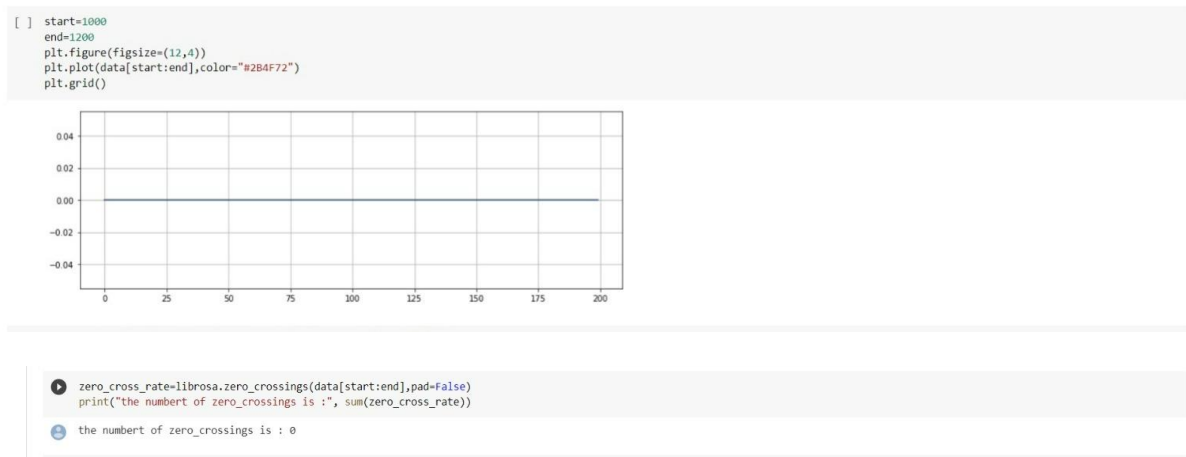


Fig-3.1.15. Zero Crossing Rate

Scaling the Features:

Standard scaler is used to standardize features by removing the mean and scaling to unit variance.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

The standard score of sample x is calculated as:

$$z = (x - \mu) / s$$

```
[ ] from sklearn.preprocessing import StandardScaler  
    fit=StandardScaler()  
    X=fit.fit_transform(np.array(df.iloc[:, :-1], dtype=float))
```

Fig-3.1.16. Standardizing dataset

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

3.2Flow of work:

- 1.Import necessary libraries and packageslike scikit-learn, numpy, pandas, librosa, pickle, tensorflowand scipy, we will build a model using CNN.
2. Read the dataset:refine the raw data and used for classifying the music genre.
- 3.The next step is pre-processing the collected raw data into an understandable format.
4. Then we have to train the data by splitting the dataset into train data and test data.
- 5.The music genre data is evaluated with the application of a machine learning algorithm i.e, CNN and the classification accuracy of this model is found.
- 6.After training the data with this algorithm we have to test on the same algorithm.
7. Finally, the result of this algorithm is compared on the basis of classification accuracy.

4.IMPLEMENTATION & TESTING

This project is implemented in python language. Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. Python has become a staple in data science, allowing data analytics and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyze data, and complete other data-related tasks. Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots. Python also has a number of libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

```
[ ] def trainModel(model,epochs,optimizer):  
    batch_size=128  
    model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',metrics='accuracy')  
    return model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=epochs,batch_size=batch_size)  
  
[ ] def plotValidate(history):  
    print("Validation Accuracy",max(history.history["val_accuracy"]))  
    pd.DataFrame(history.history).plot(figsize=(12,6))  
    plt.show()
```

Fig-4.1. Training of Data

```
[ ] import tensorflow as tf

model=tf.keras.models.Sequential([
    tf.keras.layers.Dense(512,activation='relu',input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(256,activation='relu'),
    keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(64,activation='relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(10,activation='softmax'),
])

print(model.summary())
model_history=trainModel(model=model,epochs=600,optimizer='adam')
```

Total params: 203,338
Trainable params: 203,338
Non-trainable params: 0

None
Epoch 1/600
53/53 [=====] - 2s 18ms/step - loss: 1.6919 - accuracy: 0.3926 - val_loss: 1.1614 - val_accuracy: 0.6109
Epoch 2/600
53/53 [=====] - 1s 14ms/step - loss: 1.1580 - accuracy: 0.5985 - val_loss: 0.8479 - val_accuracy: 0.7213
Epoch 3/600
53/53 [=====] - 1s 14ms/step - loss: 0.9528 - accuracy: 0.6767 - val_loss: 0.7392 - val_accuracy: 0.7537

Fig-4.1. Model Summary

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
import os
import pickle
import librosa.display
from IPython.display import Audio
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow import keras

[ ] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] import pandas
import matplotlib.pyplot as plt
df=pandas.read_csv('/content/drive/MyDrive/features_3_sec.csv.zip')
```

```
[ ] df.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	...
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388	117335.771563	...
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866	2010.051501	65671.875673	...
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613	2084.565132	75124.921716	...
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517	1960.039988	82913.639269	...
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654	1948.503884	60204.020268	...

5 rows x 60 columns

```
df.shape
```

(9990, 60)

```
df.dtypes
```

chroma_stft_var	float64
rms_mean	float64
rms_var	float64
spectral_centroid_mean	float64
spectral_centroid_var	float64
spectral_bandwidth_mean	float64
spectral_bandwidth_var	float64
rolloff_mean	float64
rolloff_var	float64
zero_crossing_rate_mean	float64
zero_crossing_rate_var	float64

```
[ ] df=df.drop(labels="filename",axis=1)

[ ] import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile
data,sr = librosa.load('drive/MyDrive/country.aac')
print(type(data),type(sr))

/usr/local/lib/python3.8/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
warnings.warn("PySoundFile failed. Trying audioread instead.")
<class 'numpy.ndarray'> <class 'int'>

librosa.load('drive/MyDrive/country.aac',sr=45600)

/usr/local/lib/python3.8/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
warnings.warn("PySoundFile failed. Trying audioread instead.")
(array([[0.          , 0.          , ..., 0.15360294, 0.09140505,
         0.          ], dtype=float32), 45600)

[ ] import IPython
IPython.display.Audio(data,rate=sr)

▶ 0:00 / 0:30 🔊 ⋮
```

```
[ ] class_list=df.iloc[:,~1]
converter=LabelEncoder()

Double-click (or enter) to edit

[ ] y=converter.fit_transform(class_list)
y
array([0, 0, 0, ..., 9, 9, 9])

[ ] print(df.iloc[:,~1])

   length  chroma_stft_mean  chroma_stft_var  rms_mean  rms_var \
0    66149      0.335406      0.091048      0.130405      0.003521
1    66149      0.343065      0.086147      0.112699      0.001450
2    66149      0.346815      0.092243      0.132003      0.004620
3    66149      0.363639      0.086856      0.132565      0.002448
4    66149      0.335579      0.088129      0.143289      0.001701
...     ...           ...           ...           ...           ...
9985   66149      0.349126      0.080515      0.050019      0.000097
9986   66149      0.372564      0.082626      0.057897      0.000088
9987   66149      0.347481      0.089019      0.052403      0.000701
9988   66149      0.387527      0.084815      0.066430      0.000320
9989   66149      0.369293      0.086759      0.050524      0.000067

   spectral_centroid_mean  spectral_centroid_var  spectral_bandwidth_mean \
0      1773.065032      167541.630869      1972.744388
1      1816.693777      90525.690866      2010.051501
2      1788.539719      111407.437613      2084.565132
3      1655.289045      111952.284517      1960.039988
4      1630.656199      79667.267654      1948.503884
```

```
[ ] from sklearn.preprocessing import StandardScaler
fit=StandardScaler()
X=fit.fit_transform(np.array(df.iloc[:,~1],dtype=float))

[ ] X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33)

[ ] len(y_test)
3297

[ ] len(y_train)
6693

[ ] from tensorflow.keras.models import Sequential

[ ] def trainModel(model,epochs,optimizer):
    batch_size=128
    model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',metrics='accuracy')
    return model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=epochs,batch_size=batch_size)

[ ] def plotValidate(history):
    print("Validation Accuracy",max(history.history["val_accuracy"]))
    pd.DataFrame(history.history).plot(figsize=(12,6))
    plt.show()
```

Fig-4.2. Sample Code for Building and Testing the model

Comparative Analysis:

The model that suites for our system has been found out through mentioned graph comparison. In this graph we shown the comparative analysis of the three models based on some of the confusion metrics.

	With data processing			Without data processing		
	Train	CV	Test	Train	CV	Test
Support Vector Machine	.97	.60	.60	.75	.32	.28
K-Nearest Neighbors	1.00	.52	.54	1.00	.21	.21
Feed-forward Neural Network	.96	.55	.54	.64	.26	.25
Convolution Neural Network	.95	.84	.82	.85	.59	.53

Fig-4.3. Comparison with other Algorithms

5.Results

To demonstrate the results of the project, the remaining data should be tested and it is tested using the algorithm. After that the trained model is ready to predict the disease is present or not.

First, CNN algorithm is trained with the training dataset and later it was tested with the remaining test data. In Fig 5.1, a screenshot of our notebook is showing that how the process of CNN algorithms is done and the accuracy the model returns and it is of 91.84%.



```
[ ] test_loss,test_acc=model.evaluate(X_test,y_test,batch_size=128)
print("The test loss is ",test_loss)
print("The best accuracy is: ",test_acc*100)
```

26/26 [=====] - 0s 3ms/step - loss: 0.5753 - accuracy: 0.9184
The test loss is 0.5752704739570618
The best accuracy is: 91.84106588363647

Fig-5.1. Test Accuracy

From this technique, we got more accuracy. Finally, it results with a genre with the input music belongs to.

6.CONCLUSION AND FUTURE WORK

For the CNN model, we had used the Adam optimizer for training the model. The epoch that was chosen for the training model is 600.

All of the hidden layers are using the RELU activation function and the output layer uses the softmax function. The loss is calculated using the sparse_categorical_crossentropy function.

Dropout is used to prevent overfitting.

We chose the Adam optimizer because it gave us the best results after evaluating other optimizers.

The model accuracy can be increased by further increasing the epochs but after a certain period, we may achieve a threshold, so the value should be determined accordingly.

The accuracy we achieved for the test set is 92.93 percent which is very decent.

So we come to the conclusion that Neural Networks are very effective in machine learning models. Tensorflow is very useful in implementing Convolutional Neural Network (CNN) that helps in the classifying process.

7.REFERENCES

- [1] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson. FMA: A Dataset For Music Analysis. Sound; Information Retrieval. arXiv:1612.01840v3, 2017.
- [2] Tom LH Li, Antoni B Chan, and A Chun. Automatic musical pattern feature extraction using convolutional neural network. In Proc. Int. Conf. Data Mining and Applications, 2010.
- [3] Hareesh Bahuleyan, Music Genre Classification using Machine Learning Techniques, University of Waterloo, 2018
- [4] Loris Nanni, Yandre MG Costa, Alessandra Lumini, Moo Young Kim, and Seung Ryul Baek. Combining visual and acoustic features for music genre classification. Expert Systems with Applications 45:108–117, 2016.
- [5] Thomas Lidy and Alexander Schindler. Parallel convolutional neural networks for music genre and mood classification. MIREX2016, 2016.
- [6] Chathuranga, Y. M. ., & Jayaratne, K. L. Automatic Music Genre Classification of Audio Signals with Machine Learning Approaches. GSTF International Journal of Computing, 3(2), 2013.
- [7] Fu, Z., Lu, G., Ting, K. M., & Zhang, D. A survey of audio based music classification and annotation. IEEE Transactions on Multimedia, 13(2), 303–319, 2011.