VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Project Report

CSE3047 – Predictive Analytics

Music Classification & Recommendation System

Project by

Sudeep Roy K	20BDS0113	B2+TB2
Prithvi Navneet Gupta	20BDS0179	B2+TB2
Mohd. Misbah	20BDS0284	B2+TB2
Shri R	20BDS0320	B2+TB2
Vishnu Ramkumar	20BDS0250	B1+TB1

Under Guidance of

Prof. Gunavathi C

Fall Semester 2022-23

DECLARATION

We hereby declare that the project report submitted by our team titled "**Music Classification & Recommendation System**", for the award of the degree of Bachelor of Technology in Computer Science to Vellore Institute of technology, VIT, is a record of the work carried out by our team under the guidance of Prof. Gunavathi C.

We further declare that the work reported in this project report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore, Tamil Nadu

Date: 12th Nov, 2022

CERTIFICATION

This is to certify that the project report entitled "**Music Classification & Recommendation System**" submitted by the team Sudeep Roy K(20BDS0113), Prithvi Navneet Gupta (20BDS0179), Mohd. Misbah (20BDS0284), Shri R (20BDS0320), Vishnu Ramkumar (20BDS0250), SCOPE, VIT University, for the award of the degree of Bachelor of Technology in Computer Science, is a record of work carried out by them under my supervision during the period 20.07.2022 to 30.11.2022, as per the VIT code of academics and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Place: Vellore, Tamil Nadu

Date: 9th Nov, 2022

ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all those who gave me the possibility to complete this project. Special thanks are due to my supervisor **Prof. Gunavathi C**, whose help, stimulating suggestions, and encouragement helped us at all time of the fabrication process and in making this project. We also sincerely thank you for the time spent proofreading.

We would also like to acknowledge with much appreciation the crucial role of our HOD, for providing us with this wonderful opportunity to work on this project. This project would not have been accomplished without their help and insights.

Many thanks go to the whole lecturer and supervisors who have given their full effort in guiding the team in achieving the goal as well as their encouragement to maintain our progress in track. Our profound thank goes to all my classmates, especially to my teammates for spending their time in helping and giving support whenever we need it in fabricating our project.

Finally, we wish to thank our parents for their support and encouragement throughout our study.

ABSTRACT

In recent years, online music streaming services have rapidly increased. Searching for new songs which fall under user's similar tastes has become a challenging issue.

In this project, we develop a music genre classification model using a convolutional neural network which takes user input songs and outputs its genre.

We also developed a personalized music recommendation system using Euclidean Distance, Cosine Similarity and Correlation Distance.

TABLE of CONTENTS

1. INTRODUCTION

Online music streaming services such as Spotify, iTunes, Saavn, etc. all provide users with millions of songs from a wide variety of artists, genres and decades to choose from, along with complex recommendation algorithms to suggest new music tailored to fit the tastes of any user.

For example, Spotify provides users a web API to extract audio features and metadata like popularity, tempo, loudness, keys and the year of release to create music recommendation systems based on both collaborative and content-based filtering.

In this project we build two models:

1) Music Genre Classification System
2) Music Recommendation System

Music Genre Classification System

Genre classification is the base of any strong music recommendation system which helps recommend songs based on the core style or category of songs, namely its genre.

We develop a model which can take an input song from the user, extract its features and output the genre under which it falls under with a high degree of accuracy.

Music Recommendation System

Searching for songs we like manually is a difficult and time-consuming process. This also requires having access to a huge dataset of songs and finding songs which match a given taste.

The goal of any good music recommendation system is to provide users with a selection of songs matching their tastes (based on their previously heard history of songs). Recommendation systems automate finding songs by analyzing trends, popular genres and artists.

2. LITERATURE SURVEY

1. **Title**: A survey of music recommendations systems and future perspectives

Author: Marcus Pearce, Simon Dixon, Yading Song
Year published: June 2012
Abstract: Three essential components of a music recommender system—user modeling, item profiling, and match algorithms—are covered in this article. Four possible concerns with the user experience and six suggestion models are outlined.

2. **Title:** Research on Music Content Recognition and Recommendation Technology Based on Deep Learning

Author: Gao Yang, Muhammad Arif
Year published: March 2022
Abstract: With the use of user data for deep learning, a potential matrix compression approach for bettering recommendation quality, accuracy, recall rate, and other metrics as evaluation criteria, this research intends to develop music algorithms.

3. **Title:** A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge

Author: Fabio Aiolli
Year published: January 2013
Abstract: The early research we did on the Million Songs Dataset (MSD) challenge is discussed in this publication. The goal of the competition was to choose a group of music to recommend to a user based on their listening history and the listening histories of 1 million other users. The paper focussed more on

defining similarity functions; effect of “locality” of collaborative scoring function; aggregating multiple ranking strategies.

4. **Title:** Hybrid Music Recommendation using K-Means Clustering

Author: Gurpreet Singh, Vishal Kashyap, Jaskirat Singh, Bishal,Pratham Verma
Year published: July 2018
Abstract: The demand for effective music recommendation systems is increasing as the digital music market grows. Collaborative filtering is a well-liked technique that has been around for a long and is often employed. Collaborative filtering is a technique, although it is quite slow. K-Means clustering is applied on the user-features modeled using the user's listening history and MFCC features of the songs to address this.

5. **Title**: A music recommendation system based on music data grouping and user interests

Author: Hungchen Chen, Arbee L.P. Chen
Year published: October 2001
Abstract: With the expansion of the World Wide Web, a lot of music data is now accessible online. It becomes required to provide a suggestion service in addition to looking for expected music items for customers. In this study, we develop the Music Recommendation System (MRS), which offers a customized music recommendation service. On the basis of the users' favorite music groups, content-based, collaborative, and statistics-based recommendation approaches are offered.

3. METHODOLOGY

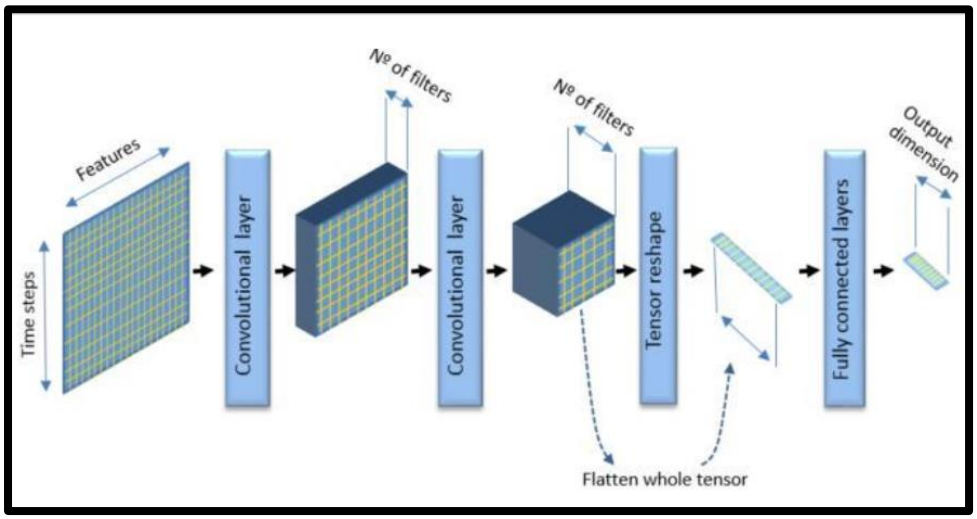
Recommendation systems generally use collaborative-based filtering or content-based filtering to classify data tuples to given labels or make novel labels from scratch with no prior knowledge given.

For ‘Genre classification’ we are using convolutional neural network (CNN or convent).

Convolutional Neural Networks

A convolutional neural network (CNN or convnet) is a machine learning subset. It is one of several types of artificial neural networks used for various applications and data types. A CNN is a type of network architecture for deep learning algorithms that is specifically used for image recognition and pixel data processing tasks.

There are other types of neural networks in deep learning, but CNNs are the network architecture of choice for identifying and recognizing objects. As a result, they are ideal for computer vision (CV) tasks and applications requiring object recognition, such as self-driving cars and facial recognition.



Deep learning algorithms rely heavily on artificial neural networks (ANNs). A recurrent neural network (RNN) is one type of ANN that takes sequential or time series data as input. It is appropriate for natural language processing (NLP), language translation, speech recognition, and image captioning applications. Another type of neural network that can uncover important information in both time series and image data is CNN.

A deep learning CNN is composed of three layers: convolutional, pooling, and fully connected (FC). The first layer is the convolutional layer, and the last layer is the FC layer.

Convolutional layer: The majority of computations take place in the convolutional layer, which is the foundation of a CNN. A second convolutional layer can be added after the first. Convolution involves a kernel or filter within this layer moving across the image's receptive fields, checking for the presence of a feature.

Pooling layer: The pooling layer, like the convolutional layer, sweeps a kernel or filter across the input image. However, in contrast to the convolutional layer, the pooling layer reduces the number of parameters in the input while also causing some information loss. On the plus side, this layer reduces complexity and improves CNN's efficiency.

Fully connected layer: The FC layer in the CNN is where image classification occurs based on the features extracted in the previous layers. Fully connected in this context means that all of the inputs or nodes from one layer are linked to every activation unit or node in the next layer.

As it would result in an overly dense network, all of the layers in the CNN are not fully connected. It would also increase losses, reduce output quality, and be computationally expensive.

We used python to create various graphs such as, Raw_Audio_Waveplot, Audio_Spectrogram, Audio_Spectral_Rolloff_Plot, Zero_Crossing_Rate Plot, etc , to classify the genre of our music input file.

- Audio_Spectrogram - A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. Spectrograms are used extensively in the fields of music, linguistics, sonar, radar, speech

processing , seismology, and others. We are using them to compare the genres wavelength, tempo

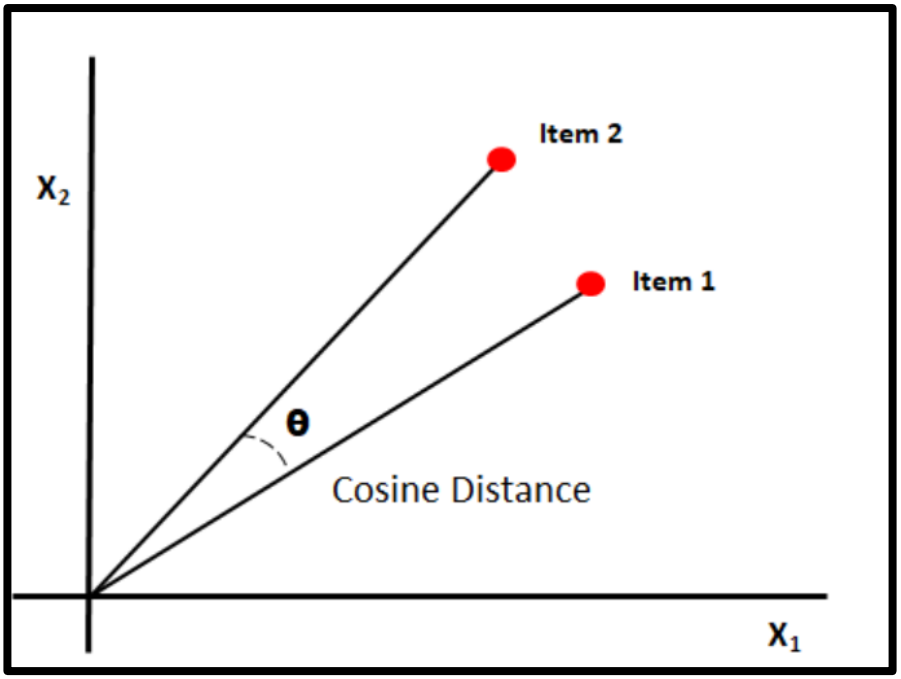
- Zero-Crossing Rate: The zero-crossing rate (ZCR) is the rate at which a signal transitions from positive to zero to negative or negative to zero to positive.

For ‘Music Recommendation’ we are using Cosine similarity, Euclidean Distance and Correlation distance.

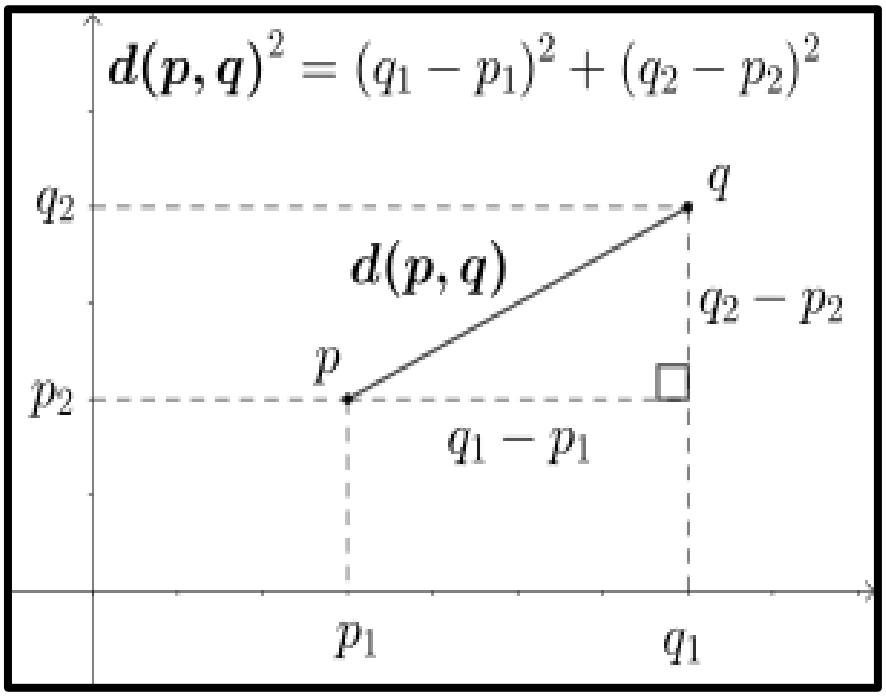
Cosine Similarity - Cosine similarity is the cosine of the angle between two n-dimensional vectors in an n-dimensional feature space. It is the dot product of the two vectors divided by the product of the magnitude of two vectors.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

To understand it better, let us take an example of two items, item 1 and item 2, and two features ‘x_1’ and ‘x_2’ , which define an item. The plot below represents item 1 and item 2 as vectors in a feature space. The lesser the angle between vectors, the more the cosine similarity.



Euclidean Distance – In mathematics, the Euclidean distance between two points in Euclidean space is the length of a line segment between the two points. It can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem



Correlation distance – In statistics and in probability theory, distance correlation or distance covariance is a measure of dependence between two paired random vectors of arbitrary, not necessarily equal, dimension. The population distance correlation coefficient is zero if and only if the random vectors are independent. Thus, distance correlation measures both linear and nonlinear association between two random variables or random vectors. Distance correlation can be used to perform a statistical test of dependence with a permutation test. One first computes the distance correlation (involving the re-centering of Euclidean distance matrices) between two random vectors, and then compares this value to the distance correlations of many shuffles of the data.

The distance correlation of two random variables is obtained by dividing their distance covariance by the product of their distance standard deviations. The distance correlation is the square root of the formula given below.

$$\operatorname{dCor}^2(X,Y) = \frac{\operatorname{dCov}^2(X,Y)}{\sqrt{\operatorname{dVar}^2(X)\,\operatorname{dVar}^2(Y)}},$$

The *sample distance variance* is the square root of

$$\operatorname{dVar}_n^2(X) := \operatorname{dCov}_n^2(X,X) = \frac{1}{n^2}\sum_{k,\ell} A_{k,\ell}^2,$$

$$\operatorname{dCov}_n^2(X,Y) := \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}\,B_{j,k}.$$

4. IMPLEMENTATION

A. Genre Classification Model

I. Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
import sys
import os
import csv
import librosa
import librosa.display
from IPython.display import Audio
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import tensorflow as tf
from tensorflow import keras
import plotly.express as px
from pydub import AudioSegment
```

II. Dataset Pre-processing

```
df = df.drop(labels='filename',axis=1)
df = df.drop(labels='length',axis=1)
df.head()
class_list = df.iloc[:, -1]
print(np.unique(np.array(class_list)))
convertor=LabelEncoder()
y = convertor.fit_transform(class_list)
print(np.unique(np.array(y)))
from sklearn.preprocessing import StandardScaler
fit = StandardScaler()
z= fit.fit(np.array(df.iloc[:, :-1], dtype= float))
X=z.transform(np.array(df.iloc[:, :-1], dtype= float))
X
```

III. Building & Training Model

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



from keras.models import Sequential
#from keras.wrappers.scikit learn import KerasClassifier
def trainModel (model, epochs, optimizer):
  batch_size=128
  #callback myCallback()
  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)

model=keras.models.Sequential([
    keras.layers.Dense(512, activation='relu',
input_shape=(X_train.shape[1],)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout (0.2),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax'),])
print(model.summary())
model_history=trainModel(model=model, epochs=600, optimizer='adam')
```

IV. Audio File Feature Extraction

```
header = 'name chroma_stft_mean chroma_stft_var rms_mean rms_var
spectral_centroid_mean spectral_centroid_var spectral_bandwidth_mean
spectral_bandwidth_var rolloff_mean rolloff_var zero_crossing_rate_mean
zero_crossing_rate_var harmony_mean harmony_var perceptr_mean
perceptr_var tempo'
for i in range(1, 21):
    header += f' mfcc{i}_mean mfcc{i}_var'
header = header.split()
print(header)
```

```
file = open('data.csv', 'a', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
directory = "C:/Users/sudee/OneDrive/Desktop/projects/PA/Genre
Classification/Music/"
output_file = "result.wav"
for file in os.listdir(directory):
    my_file = os.path.join(directory, file)
    sound = AudioSegment.from_mp3(my_file)
    sound.export(output_file, format="wav")
    songname = f'result.wav'
    y, sr = librosa.load(songname, mono=True, duration=30)
    chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
    rmse = librosa.feature.rms(y=y)[0]
    spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
    spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
    zcr = librosa.feature.zero_crossing_rate(y)
    harmonic = librosa.effects.harmonic(y=y)
    perceptr = librosa.effects.percussive(y=y)
    mfcc = librosa.feature.mfcc(y=y, sr=sr)
    tempo = librosa.beat.tempo(y=y, sr=sr)
    to_append = f'{file} {np.mean(chroma_stft)} {np.var(chroma_stft)}
{np.mean(rmse)} {np.var(rmse)} {np.mean(spec_cent)} {np.var(spec_cent)}
{np.mean(spec_bw)} {np.var(spec_bw)} {np.mean(rolloff)}
{np.var(rolloff)} {np.mean(zcr)} {np.var(zcr)} {np.mean(harmonic)}
{np.var(harmonic)} {np.mean(perceptr)} {np.var(perceptr)}
{np.mean(tempo)}'
    for e in mfcc:
        to_append += f' {np.mean(e)} {np.var(e)}'

    print(to_append)
    file = open('data.csv', 'a', newline='')
    with file:
        writer = csv.writer(file)
        writer.writerow(to_append.split())
```

V. Predicting Genre from Audio File

```
df = df.reset_index()
df_name=df[['name']]
df = df.drop(labels='name',axis=1)
predict_list = []

for i in range(len(df)):
    predict_input = (df.loc[i]).values.tolist()
    predict_list.append(predict_input)
predict_list
B= z.transform(np.array(predict_list, dtype= float))
B
output_label = model.predict(B)
print(output_label)
```

VI. Audio File Plots

```
audio_recording = "Data/genres_original/country/country.00050.wav"
data, sr = librosa.load(audio_recording)
print(type(data), type(sr))
import IPython
IPython.display.Audio(data, rate=sr)

plt.figure(figsize = (12, 4))
librosa.display.waveshow(data, color = "#284F72")
plt.show()
stft = librosa.stft(data)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize=(14, 6))
librosa.display.specshow(stft, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
plt.figure(figsize=(14, 6))
librosa.display.specshow(stft_db, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()

from sklearn.preprocessing import normalize
spectral_rolloff = librosa.feature.spectral_rolloff(data+0.01,
sr=sr)[0]
plt.figure(figsize=(12, 4))
librosa.display.waveshow(data, sr=sr, alpha=0.4, color="#284F72")

import librosa.display as lplt
```

```
chroma = librosa.feature.chroma_stft(data, sr=sr)
plt.figure(figsize=(16,6))
lplt.specshow(chroma, sr=sr, x_axis='time', y_axis= 'chroma',
cmap='coolwarm')
plt.colorbar()
plt.title("Chroma Features")
plt.show()

start=1000
end=1200
plt.figure( figsize= (14, 5))
plt.plot(data[start:end], color="#2B4F72")
plt.grid()
```

VII. Trained Model Evaluation Plots

```
test_loss, test_acc=model.evaluate(X_test, y_test, batch_size=128)
print("The test Loss is :", test_loss)
print("\nThe Best test Accuracy is ", test_acc*100)

print("Validation Accuracy",max(model_history.history["val_accuracy"]))
df = pd.DataFrame(model_history.history)
print(df)
fig1 = px.line(df, title="CNN Model Evaluation Plot")
fig1.show()
```

VIII. Prediction Probability Plots

```
fig2 = px.line(df1, title="Feature Extraction Output")
fig2.show()

genre = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz',
'metal', 'pop', 'reggae', 'rock']
df = pd.DataFrame(output_label,columns =genre)
df
df.insert(loc = 0,
          column = 'name',
          value = df_name['name'])
# show the dataframe
df
df1 = df
df1.set_index("name", inplace = True)
fig3 = px.bar(df.T, title="Genre Prediction Output")
fig3.show()
```

B. Music Recommendation Model

I. Import Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import plotly
from sklearn.preprocessing import StandardScaler
from scipy.spatial import distance
import copy
import warnings
warnings.filterwarnings("ignore")
plotly.offline.init_notebook_mode (connected = True)
```

II. Dataset Pre-processing

```
cols=list(data.columns[11:])
del cols[7]
df=copy.deepcopy(data)
df.drop(columns=cols,inplace=True)
data.drop('Unnamed: 0',axis=1,inplace=True)

data=data.dropna(subset=['song_name'])
df=data[data.columns[:11]]
df['genre']=data['genre']
df['time_signature']=data['time_signature']
df['duration_ms']=data['duration_ms']
df['song_name']=data['song_name']

x=df[df.drop(columns=['song_name','genre']).columns].values
scaler = StandardScaler().fit(x)
X_scaled = scaler.transform(x)
df[df.drop(columns=['song_name','genre']).columns]=X_scaled
```

III. Song Name Lookup

```
def find_word(word,words):
    t=[]
    count=0
    if word[-1]==' ':
        word=word[:-1]
    for i in words:
        if word.lower() in i.lower():
            t.append([len(word)/len(i),count])
        else:
            t.append([0,count])
        count+=1
    t.sort(reverse=True)
    return words[t[0][1]]
```

IV. Recommendation using Euclidean Distance

```
def make_matrix(data,song,number):
    df=pd.DataFrame()
    data.drop_duplicates(inplace=True)
    songs=data['song_name'].values
    best=find_word(song,songs)
    print('The song closest to your search is :',best)
    genre=data[data['song_name']==best]['genre'].values[0]
    df=data[data['genre']==genre]

x=df[df['song_name']==best].drop(columns=['genre','song_name']).values
    if len(x)>1:
        x=x[1]
    song_names=df['song_name'].values
    df.drop(columns=['genre','song_name'],inplace=True)
    df=df.fillna(df.mean())
    p=[]
    r=[]
    count=0
    for i in df.values:
        r.append(distance.euclidean(x,i))
        p.append([distance.euclidean(x,i),count])
        count+=1
    p.sort()
    r.sort()
    ls=[]
```

```
    for i in range(1,number+1):
        ls.append(song_names[p[i][1]])
    df1 = {'song':ls,'euclidean_dist':r[1:number+1]}
    return df1

a=input('Please enter The name of the song :')
b=int(input('Please enter the number of recommendations you want: '))
df1 = make_matrix(df,a,b)
df1 = pd.DataFrame(df1)
```

V. Recommendation using Cosine Similarity Distance

```
def make_matrix_cosine(data,song,number):
    df=pd.DataFrame()
    data.drop_duplicates(inplace=True)
    songs=data['song_name'].values
    best=find_word(song,songs)
    print('The song closest to your search is :',best)
    genre=data[data['song_name']==best]['genre'].values[0]
    df=data[data['genre']==genre]

x=df[df['song_name']==best].drop(columns=['genre','song_name']).values
    if len(x)>1:
        x=x[1]
    song_names=df['song_name'].values
    df.drop(columns=['genre','song_name'],inplace=True)
    df=df.fillna(df.mean())
    p=[]
    r=[]
    count=0
    for i in df.values:
        r.append(distance.cosine(x,i))
        p.append([distance.cosine(x,i),count])
        count+=1
    p.sort()
    r.sort()
    ls=[]
    for i in range(1,number+1):
        ls.append(song_names[p[i][1]])
    df1 = {'song':ls,'cosine_dist':r[1:number+1]}
    return df1

c=input('Please enter The name of the song :')
```

```
d=int(input('Please enter the number of recommendations you want: '))
df1 = make_matrix_cosine(df,c,d)
df1 = pd.DataFrame(df1)
```

VI. Recommendation using Correlation Distance

```
def make_matrix_correlation(data,song,number):
    df=pd.DataFrame()
    data.drop_duplicates(inplace=True)
    songs=data['song_name'].values
    best=find_word(song,songs)
    print('The song closest to your search is :',best)
    genre=data[data['song_name']==best]['genre'].values[0]
    df=data[data['genre']==genre]

x=df[df['song_name']==best].drop(columns=['genre','song_name']).values
    if len(x)>1:
        x=x[1]
    song_names=df['song_name'].values
    df.drop(columns=['genre','song_name'],inplace=True)
    df=df.fillna(df.mean())
    p=[]
    r=[]
    count=0
    for i in df.values:
        r.append(distance.correlation(x,i))
        p.append([distance.correlation(x,i),count])
        count+=1
    p.sort()
    r.sort()
    ls=[]
    for i in range(1,number+1):
        ls.append(song_names[p[i][1]])
    df1 = {'song':ls,'correlation_dist':r[1:number+1]}
    return df1

e=input('Please enter The name of the song :')
f=int(input('Please enter the number of recommendations you want: '))
df1 = make_matrix_correlation(df,e,f)
df1 = pd.DataFrame(df1)
```

VII. Dataset Visualization Plots

```
fig4 = px.line(data[data.columns[:11]],title='Features of Dataset')
fig4.show()

fig5 = px.pie(data,names=data[data.columns[-4]],title="Genre
Distribution of Dataset")
fig5.show()

x=list(data.corr().columns)
y=list(data.corr().index)
values=np.array(data.corr().values)
fig6 = go.Figure(data=go.Heatmap(
    z=values,
    x=x,
    y=y,
    hoverongaps = False, colorscale='Aggrnyl'))
fig6.update_layout(title='Correlation Heatmap of Dataset')
fig6.show()
```

VIII. Recommendation Result Plots

```
fig7 =
px.line(df1,x=df1['song'],y=df1['euclidean_dist'],title='Recommendation
Using Euclidean Distance',markers=True)
fig7.show()

fig8 =
px.line(df1,x=df1['song'],y=df1['cosine_dist'],title='Recommendation
Using Cosine Similarity Distance',markers=True)
fig8.show()

fig9 =
px.line(df1,x=df1['song'],y=df1['correlation_dist'],title='Recommendato
nn Using Correlation Distance',markers=True)
fig9.show()
```

5. RESULT

A. Genre Classification Model

I. Dataset Pre-processing

```
array([[-0.48780784,  0.64052047, -0.00662408, ..., -0.51356204,
         0.12841417, -0.29178072],
       [-0.40314187,  0.13183473, -0.26494432, ...,  1.01138445,
         1.27578001,  0.05642464],
       [-0.36169428,  0.7644909 ,  0.01669533, ..., -0.04624405,
         0.65390663, -0.52145798],
       ...,
       [-0.35433044,  0.42997426, -1.14464442, ..., -0.15370124,
         0.11765485, -0.33882395],
       [ 0.0883611 , -0.00630133, -0.93999575, ..., -0.72456977,
         0.30333409, -0.95893743],
       [-0.11321002,  0.19536324, -1.17205474, ..., -0.37245283,
        -0.47495901, -0.55112155]])
```

Output Dataset after Pre-processing

II. Building & Training Model

```
=================================================================
Total params: 202,826
Trainable params: 202,826
...
Epoch 599/600
53/53 [==============================] - 0s 9ms/step - loss: 0.0087 - accuracy:
0.9973 - val_loss: 0.5761 - val_accuracy: 0.9163
Epoch 600/600
53/53 [==============================] - 0s 9ms/step - loss: 0.0057 - accuracy:
0.9979 - val_loss: 0.6318 - val_accuracy: 0.9154
```

Epoch Result

III. Audio File Feature Extraction

	name	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean
0	24k_magic.mp3	0.447259	0.089772	0.268961	0.006627	2661.691806	852536.987703	2620.133834
1	angel_of_death.mp3	0.620491	0.051823	0.232580	0.001422	3129.300162	211912.659662	2607.662831
2	blinding.mp3	0.413992	0.082264	0.303647	0.005714	2403.434187	655346.404259	2502.174420
3	eminem_rap_god.mp3	0.391850	0.091883	0.269059	0.008262	2354.570962	897297.161218	2322.468995
4	mozart_classic.mp3	0.211787	0.085305	0.080697	0.002716	925.554084	43083.609919	1087.242415
5	nanchak.mp3	0.506156	0.087415	0.253803	0.017939	2698.331707	838991.482867	2856.727618
6	night_fever_disco.mp3	0.373885	0.084838	0.112127	0.001268	2370.107749	508828.998109	2619.519098
7	purple_haze.mp3	0.435694	0.084136	0.187512	0.003197	2355.993475	237681.948386	2109.503381
8	reggae_bob.mp3	0.393482	0.095247	0.175453	0.006026	2080.100221	860930.932633	2478.889915
9	rock_and_roll.mp3	0.435333	0.078401	0.176033	0.000775	2624.394391	146829.043621	2621.686063
10	starboy.mp3	0.505851	0.076774	0.356599	0.007405	1736.307739	489128.518929	2213.359855
11	take_five.mp3	0.249689	0.095684	0.093074	0.002174	2384.867922	246719.895051	2723.811875
12	take_me_home.mp3	0.315989	0.084407	0.113494	0.000918	2196.802866	363360.533096	2113.370228
13	thrill_is_gone.mp3	0.374929	0.094397	0.143870	0.002147	1225.869025	305665.177241	1650.770386

Subset of Feature Extracted Data

```
array([[ 7.48663954e-01,  5.08116853e-01,  2.01485101e+00,
         1.10184032e+00,  6.15134663e-01,  1.00211895e+00,
         6.96448901e-01,  6.40026346e-01,  6.81831415e-01,
         8.65525257e-01,  2.53673825e-01,  8.99609422e-01,
         2.28943618e-01,  1.71624129e+00,  2.46834129e-01,
         2.80100991e+00, -5.23296223e-01,  1.10792800e+00,
         3.48482557e-01, -6.39690694e-01,  1.42728963e+00,
         4.17815491e-01,  8.02424501e-01, -6.49758924e-01,
         1.30313714e+00,  2.19054751e-01,  8.02290152e-01,
         2.26762524e-01,  3.04104040e-01,  1.21166536e-01,
         8.62164518e-01,  2.30597927e-01,  7.50577679e-01,
         6.56840662e-01,  5.62221753e-01,  3.55956650e-01,
         4.92997618e-01,  5.47447794e-01,  5.72257904e-01,
         7.71778658e-01,  4.47608632e-01,  8.23492087e-02,
         4.19633433e-01,  7.51262006e-01,  3.39351274e-01,
        -2.56810840e-01,  4.44033293e-01,  9.99972317e-01,
         3.05158657e-01, -5.48048050e-01,  5.74386677e-01,
         1.84565179e+00,  8.89959122e-01, -2.65299402e-01,
         7.45975480e-01,  1.66255894e+00,  4.50321057e-01],
```

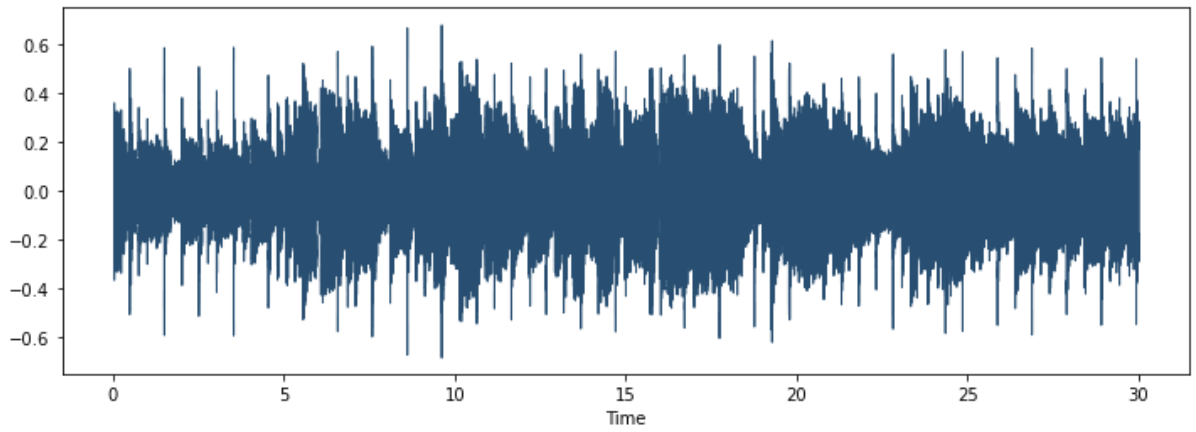
Transformed Feature Data

IV. Predicting Genre from Audio File

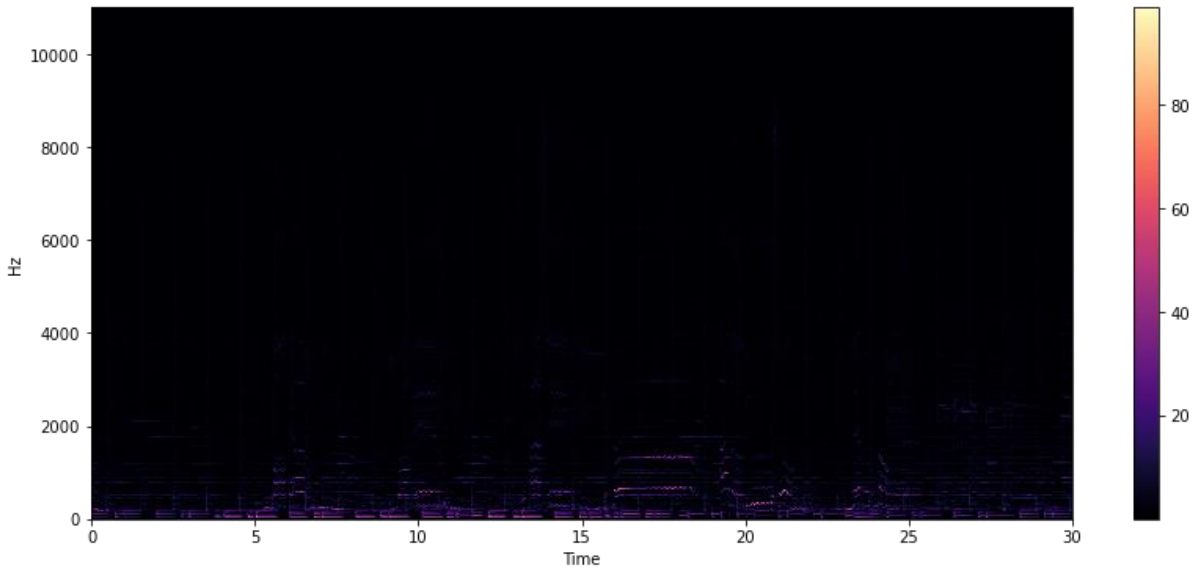
```
[[2.45499328e-15 3.81786345e-14 8.83967032e-12 1.81118820e-09
  9.99999285e-01 1.22743807e-14 5.20618028e-11 5.51438347e-07
  6.42823323e-08 7.39978044e-12]
 [3.41070066e-29 5.45270320e-34 4.77867901e-28 6.43823464e-24
  5.76567273e-24 4.84132227e-35 1.00000000e+00 0.00000000e+00
  1.68095208e-36 1.60451960e-22]
 [6.34525786e-05 1.29941822e-04 1.92657746e-02 9.35927927e-01
  3.79917361e-02 1.70646969e-03 3.94318136e-04 6.05276786e-04
  1.39216520e-03 2.52283108e-03]
 [7.99042397e-08 1.88808205e-08 1.06584366e-05 4.54307592e-05
  9.99758542e-01 6.91009845e-08 4.45004709e-08 2.73438927e-05
  1.57688395e-04 7.85540308e-08]
 [4.90368922e-13 1.00000000e+00 1.90504486e-13 8.26021516e-14
  7.16067678e-16 4.05334930e-08 9.49606350e-15 2.80898351e-13
  1.66333845e-16 3.18230143e-14]
 [0.00000000e+00 1.12518120e-35 2.12757312e-30 1.97868206e-21
  1.00000000e+00 0.00000000e+00 6.04040194e-28 2.35827766e-18
  1.76081554e-22 2.81280985e-29]
 [6.39991597e-08 4.41057146e-09 1.96838710e-06 9.99940872e-01
  8.61188073e-06 2.65075442e-07 8.75213755e-06 6.20802439e-06
  1.96728070e-05 1.36114222e-05]
 [6.75014704e-02 1.35801911e-05 1.11090427e-03 3.95409705e-04
  6.26314955e-04 1.20778801e-04 1.52832326e-02 1.01764446e-04
  2.02028304e-02 8.94643724e-01]
...
  8.87647067e-10 2.31823098e-04]
 [9.99145269e-01 4.20470853e-10 5.45440528e-08 2.67760425e-10
  1.81789486e-10 1.43887027e-08 2.79810042e-09 3.73890804e-15
  8.54682236e-04 1.20389032e-10]]
```

Predicted Probability Data

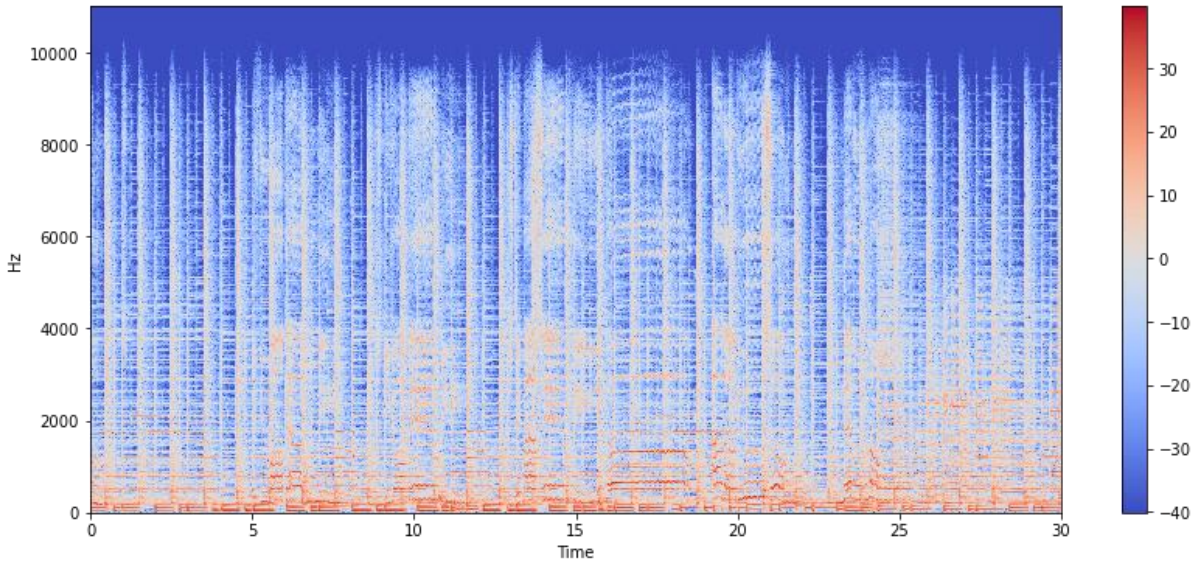
V. Audio File Plots



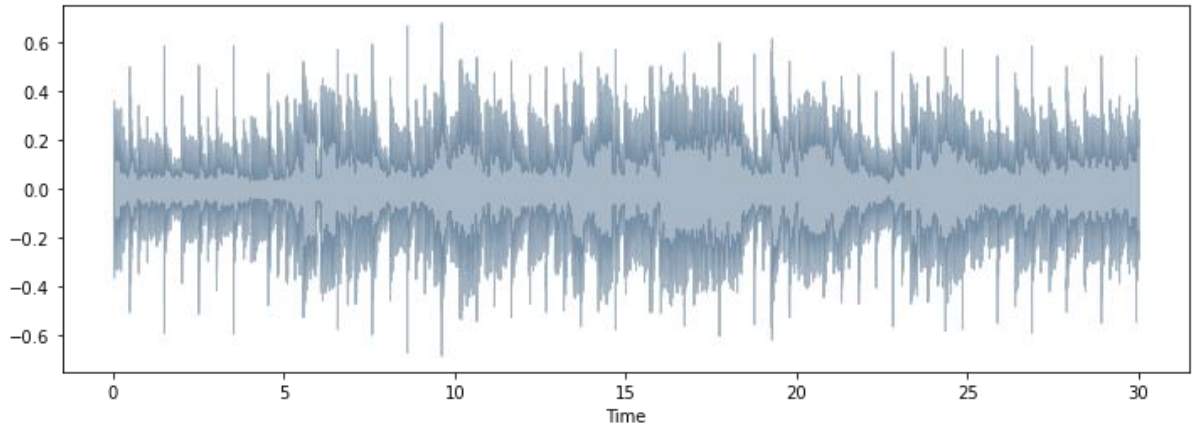
Raw Audio File Plot



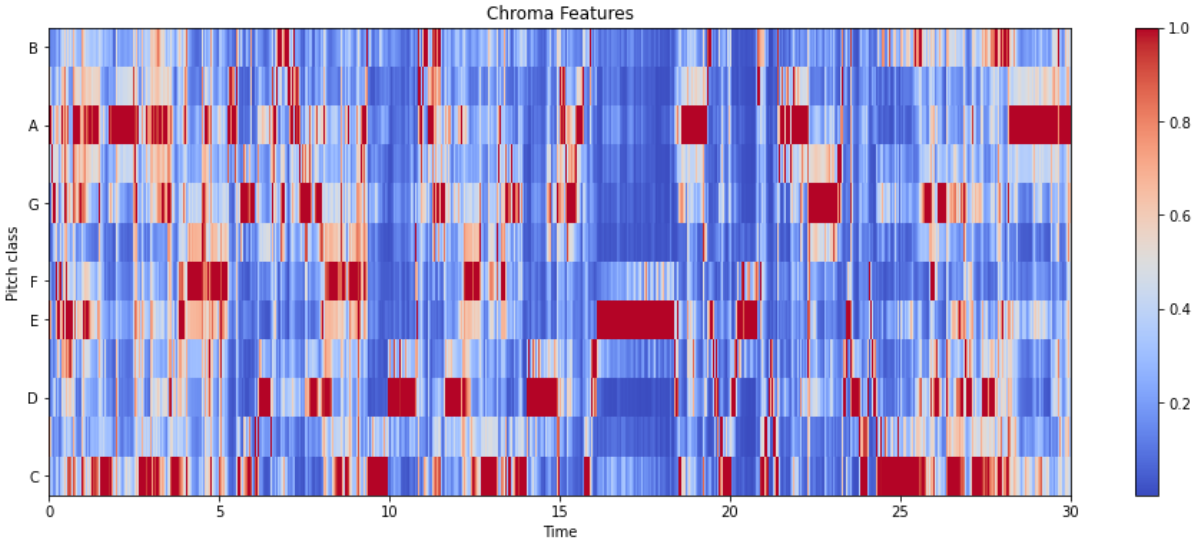
Spectrogram Plot



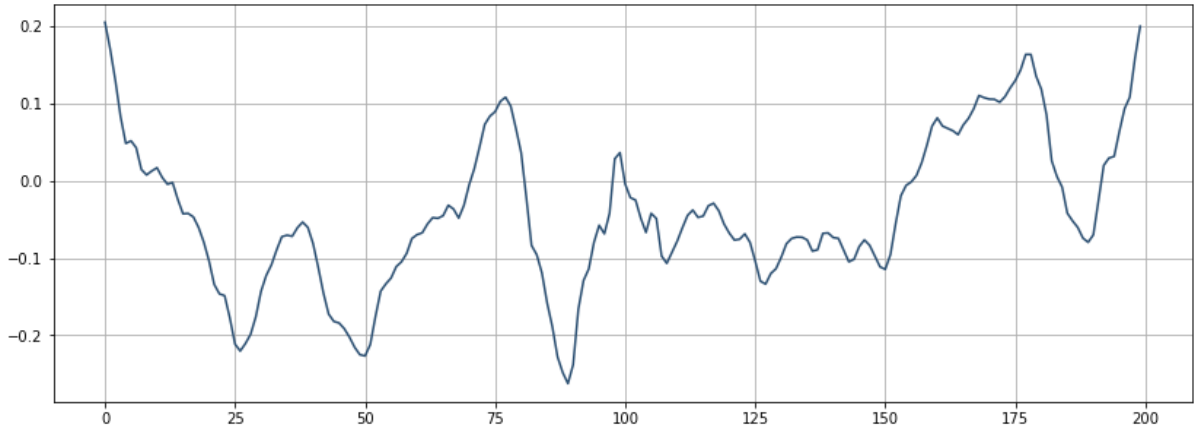
Spectrogram Plot Recolor



Audio Frequency Roll off Plot

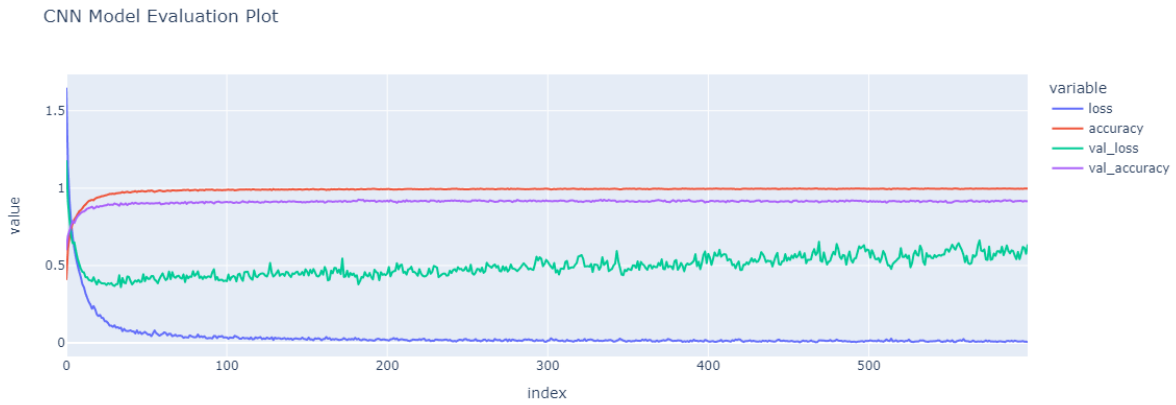


Chroma Features Plot

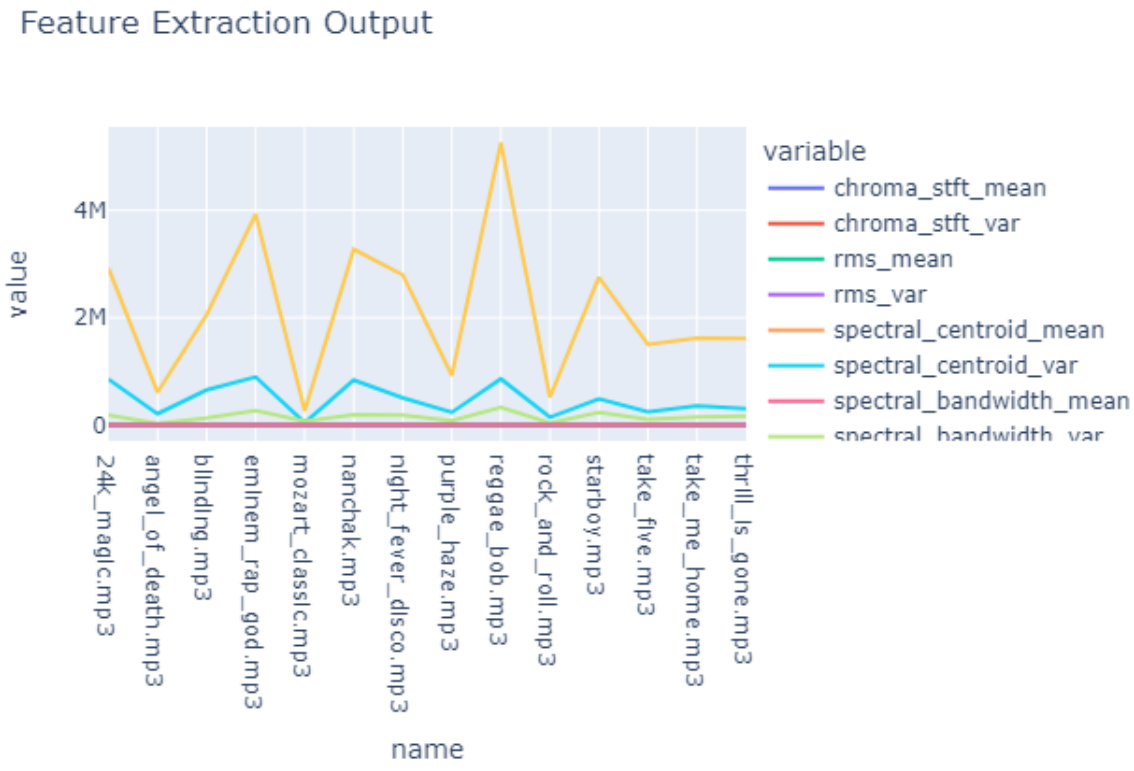


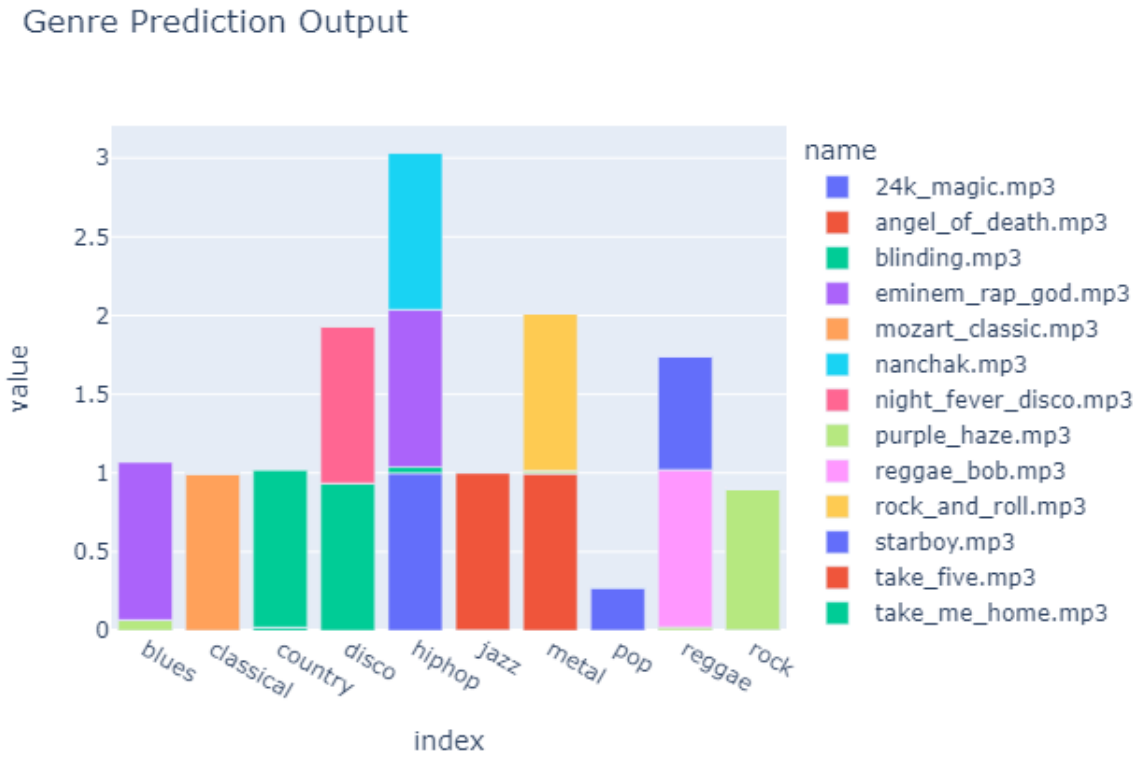
Zero Crossing Plot

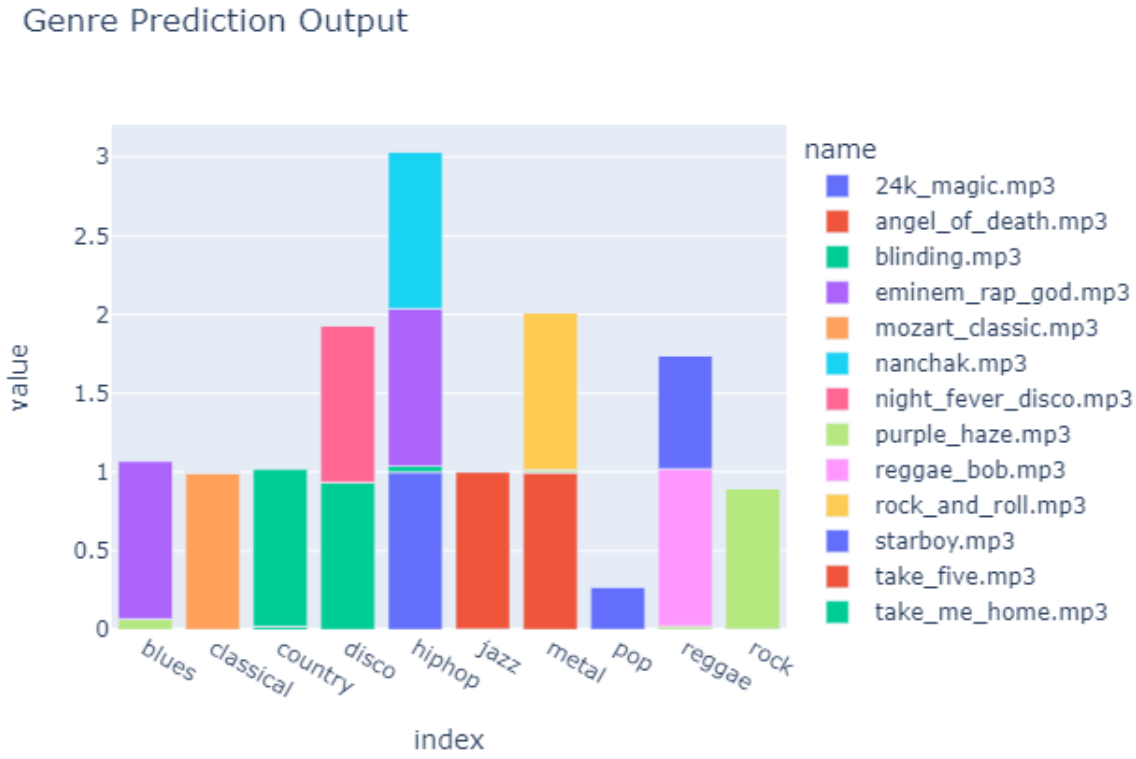
VI. Trained Model Evaluation Plots



VII. Prediction Probability Plots

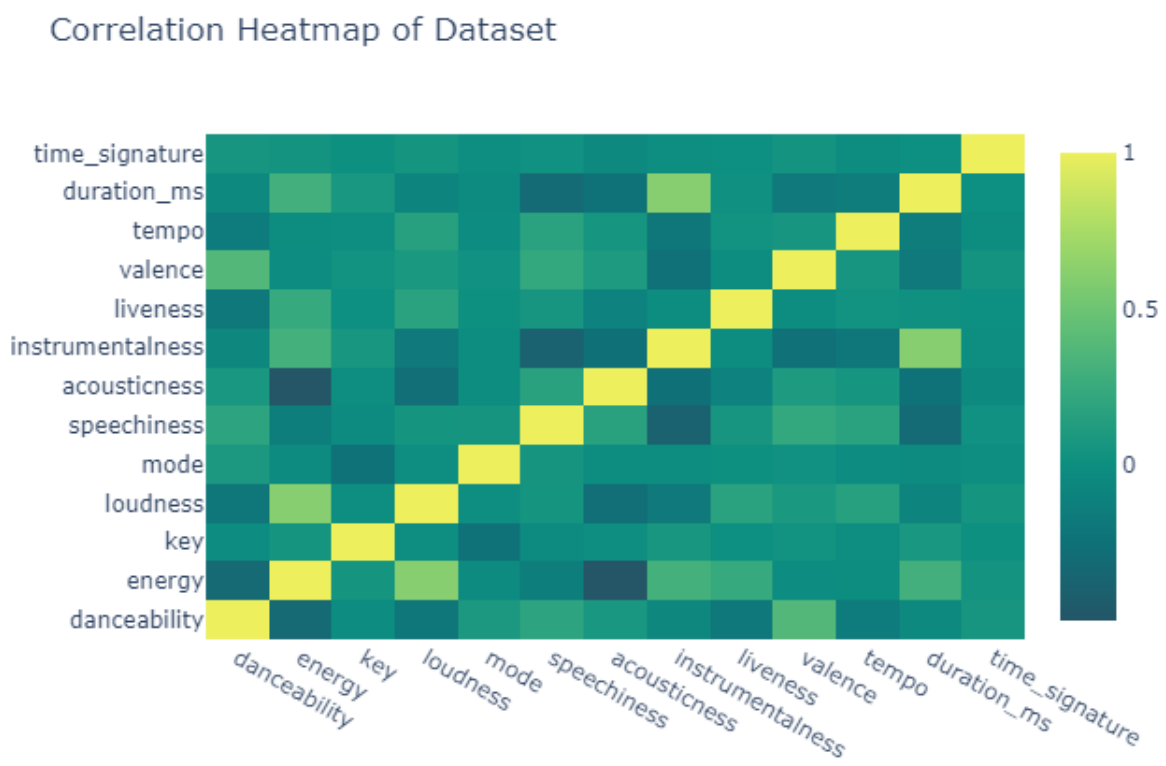



Genre Prediction Output
value
index
blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
name
24k_magic.mp3
angel_of_death.mp3
blinding.mp3
eminem_rap_god.mp3
mozart_classic.mp3
nanchak.mp3
night_fever_disco.mp3
purple_haze.mp3
reggae_bob.mp3
rock_and_roll.mp3
starboy.mp3
take_five.mp3
take_me_home.mp3

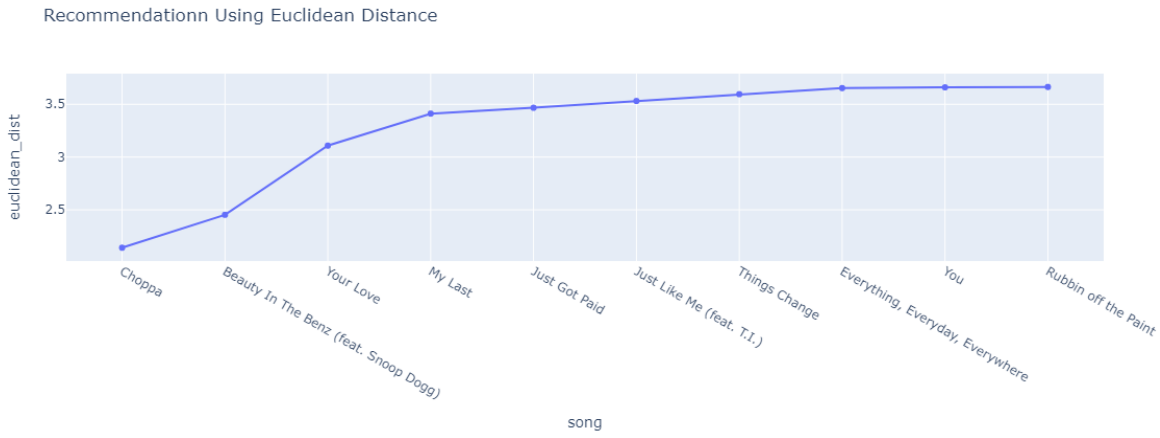

Genre Prediction Output
value
index
blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
name
24k_magic.mp3
angel_of_death.mp3
blinding.mp3
eminem_rap_god.mp3
mozart_classic.mp3
nanchak.mp3
night_fever_disco.mp3
purple_haze.mp3
reggae_bob.mp3
rock_and_roll.mp3
starboy.mp3
take_five.mp3
take_me_home.mp3

B. Music Recommendation Model

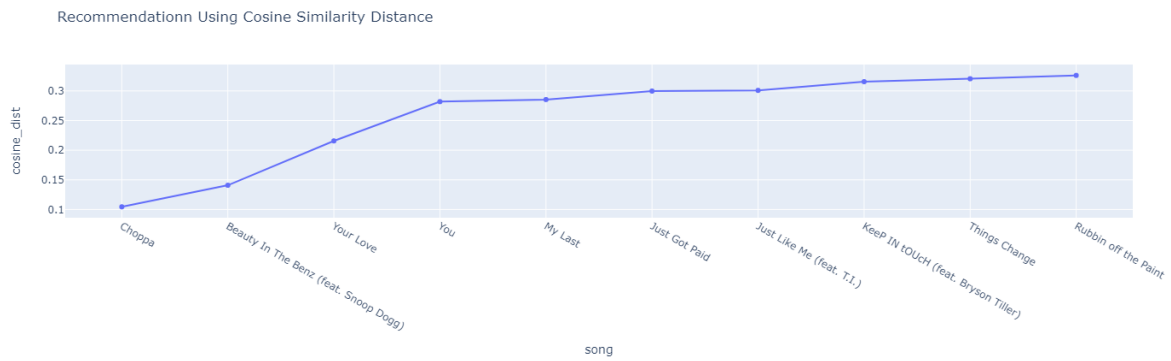
I. Dataset Pre-processing



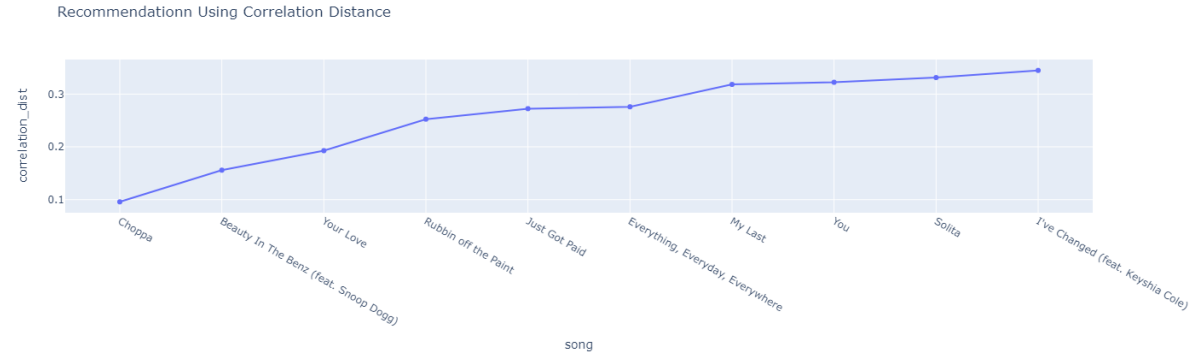
II. Recommendation using Euclidean Distance



III. Recommendation using Cosine Similarity Distance



IV. Recommendation using Correlation Distance



V. Dataset Visualization Plots

6. CONCLUSION

In the end we successfully predict the genre of our music and also suggest recommendations based around our input audio file. The results obtained were effectively visualized with the help of graphs.

7. REFERENCES

[1] Song, Yading & Dixon, Simon & Pearce, Marcus. (2012). A Survey of Music Recommendation Systems and Future Perspectives.

[2] Yang, G. (2022) 'Research on Music Content Recognition and Recommendation Technology Based on Deep Learning', *Security and Communication Networks*. Hindawi, 2022, σ. 7696840. doi: 10.1155/2022/7696840.

[3] Aiolli, Fabio. (2013). A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge. 964.

[4] Hung-Chen Chen and Arbee L. P. Chen. 2001. A music recommendation system based on music data grouping and user interests. In Proceedings of the tenth international conference on Information and knowledge management (CIKM '01). Association for Computing Machinery, New York, NY, USA, 231–238. https://doi.org/10.1145/502585.502625

[5] GTZAN data set (for Genre Classification)