

# **EMOTION AWARE MUSIC RECOMMENDATION SYSTEM**

A

*Project Report*

*Submitted in partial fulfilment of the  
Requirements for the award of the Degree of*

## **BACHELOR OF ENGINEERING**

IN

## **COMPUTER SCIENCE & ENGINEERING**

By

**SAI SUMAN CHITTURI**

**1602-18-733-097**

**PRANEETH KAPILA**

**1602-18-733-116**

*Under the guidance of*

**Dr. T. ADILAKSHMI**

**HEAD OF THE DEPARTMENT**



**Department of Computer Science & Engineering  
Vasavi College of Engineering (Autonomous)  
(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2022**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Hyderabad-500 031**

**Department of Computer Science & Engineering**



### **DECLARATION BY THE CANDIDATE**

We, **SAI SUMAN CHITTURI** and **PRANEETH KAPILA**, bearing hall ticket numbers **1602-18-733-097** and **1602-18-733-116**, hereby declare that the project report entitled "**EMOTION AWARE MUSIC RECOMMENDATION SYSTEM**" under the guidance of **Dr .T. ADILAKSHMI**, HOD, Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF ENGINEERING** in **COMPUTER SCIENCE & ENGINEERING**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**SAI SUMAN CHITTURI,**

**1602-18-733-097.**

**PRANEETH KAPILA,**

**1602-18-733-116.**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Hyderabad-500 031**

**Department of Computer Science & Engineering**



**BONAFIDE CERTIFICATE**

This is to certify that the project entitled "**EMOTION AWARE MUSIC RECOMMENDATION SYSTEM**" being submitted by **SAI SUMAN CHITTURI** and **PRANEETH KAPILA**, bearing hall ticket numbers **1602-18-733-097** and **1602-18-733-116**, in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING** is a record of bonafide work carried out by them under my guidance.

**Dr. T. Adilakshmi,**  
**Professor & HOD,**  
**Internal guide**

**Dr. T. Adilakshmi,**  
**Professor & HOD,**  
**Dept. of CSE**

## **Acknowledgement**

We are thankful to the College Management for Encouraging us to do our Project titled **EMOTION AWARE MUSIC RECOMMENDATION SYSTEM**. We extend our heart-felt gratitude to our Professor, **Dr. M. SUNITHA REDDY** and our Internal Guide and Head of the Dept., **Dr. T. ADILAKSHMI** for their Invaluable Guidance and Support throughout our project. We extend our heartfelt gratitude to the Principal, **Dr. S. V. RAMANA**, Vasavi College of Engineering, Ibrahimbagh, for permitting us to undertake this Project. Their guidance was unforgettable, and their constructive suggestions helped us in finishing the project Effectively.

**SAI SUMAN CHITTURI,**

**1602-18-733-097**

**PRANEETH KAPILA,**

**1602-18-733-116.**

## **Abstract**

Music is an essential part of our regular life. It cheers us up and makes us feel better. Not all forms of music are appropriate for every mood. Furthermore, ever-growing digital music catalogues make it virtually impossible to recollect a specific tune that fits the present emotion. Besides that, due to the enormous number of songs accessible, people are frequently perplexed when selecting a track. This necessitates the development of a context-sensitive music recommendation system.

Therefore, we present a context-aware music recommendation system that assists in identifying the user's current emotion and suggesting music which is relevant to that emotion. We have come up with a comprehensive strategy to improve user preference prediction; our technique integrates context and emotion elements and strives to give users a more convenient, intuitive, and pleasurable listening experience. Finally, we discuss the evaluation and performance metrics and results of our research.

## 7. Table of Contents

<b>Abstract</b>	v
<b>List of Tables</b>	viii
<b>List of Figures</b>	ix
<b>1. Introduction</b>	1
1.1. Facial Emotion Identification	2
1.2. Music Recommendation	5
1.2.1. Collaborative Filtering	5
1.2.2. Content based Filtering	6
<b>2. Overview of proposed system</b>	7
2.1. Emotion Identification	7
2.2. Music Recommendation	8
<b>3. System design</b>	9
3.1. Emotion Identification Module	9
3.2. Music Tagging & Recommendation Module	10
3.3. User-Interface Module	11
<b>4. Pseudo Algorithm / Implementation</b>	12
4.1. Emotion Identification	12
4.1.1. Modules used	12
4.1.2. CNN for facial emotion identification	13
4.1.3. Deep Neural Network	17
4.1.4. Multi-task Cascaded Neural Network	21
4.2. Music Recommendation	23
4.3. The Backend	27
4.4. The Frontend	30
<b>5. Results</b>	32
5.1. Accurate Predictions	32
5.2. Inaccurate Predictions	39
5.3. No-face-detected	41

<b>6. Testing</b>	<b>42</b>
6.1. Accuracy of CNN Model	42
6.2. Accuracy for Deep Face Model	43
6.2.1. Dataset: AffectNet Test	43
6.2.2. Dataset: AffectNet Train	44
6.3. Accuracy of MTCNN based FER Model	47
6.4. Accuracy for K-MEANS clustering	48
6.5. Compare & Contrast Emotion Detection Models	48
6.6. Emotion-wise Accuracies of all models	49
<b>7. Conclusion &amp; Future work</b>	<b>50</b>
<b>8. References</b>	<b>51</b>

## List of Tables

<b>Table 2.1.1</b> : Compare & Contrast of Emotion Detection Models	<b>7</b>
<b>Table 4.1.2.1</b> : Arguments passed to Keras Model	<b>13</b>

## List of Figures

<b>Figure 1.1.1</b> : Basic model of how Neurons in ANN work	<b>2</b>
<b>Figure 1.1.2</b> : Processing of a digital image by Neural Network	<b>3</b>
<b>Figure 1.1.3</b> : Emotion Identification by a Neural Network	<b>4</b>
<b>Figure 1.1.4</b> : Neural Network involving more hidden layers	<b>5</b>
<b>Figure 1.2.1</b> : Collaborative Filtering	<b>6</b>
<b>Figure 1.2.2</b> : Content-based Filtering	<b>6</b>
<b>Figure 3.1.1</b> : Emotion Identification module	<b>9</b>
<b>Figure 3.2.1</b> : Music Recommendation module	<b>10</b>
<b>Figure 3.3.1</b> : User-Interface Module	<b>11</b>
<b>Figure 4.1.2.1</b> : Script to find the accuracy of CNN model (1)	<b>14</b>
<b>Figure 4.1.2.2</b> : Script to find the accuracy of CNN model (2)	<b>15</b>
<b>Figure 4.1.2.3</b> : Script to find the accuracy of CNN model (3)	<b>16</b>
<b>Figure 4.1.3.1</b> : Deep Neural Network	<b>17</b>
<b>Figure 4.1.3.2</b> : Sample Snippet demonstrating Deep Face Module	<b>18</b>
<b>Figure 4.1.3.3</b> : Script for finding the accuracy of Deep Face module (1)	<b>19</b>
<b>Figure 4.1.3.4</b> : Script for finding the accuracy of Deep Face module (2)	<b>20</b>
<b>Figure 4.1.4.1</b> : Sample snippet demonstrating MTCNN model	<b>21</b>
<b>Figure 4.1.4.2</b> : Script to find accuracy of MTCNN model	<b>22</b>
<b>Figure 4.2.1</b> : Script to Cluster using K-MEANS (1)	<b>23</b>
<b>Figure 4.2.2</b> : Script to Cluster using K-MEANS (2)	<b>24</b>
<b>Figure 4.2.3</b> : Script to Cluster using K-MEANS (3)	<b>25</b>
<b>Figure 4.2.4</b> : Script to Cluster using K-MEANS (4)	<b>26</b>
<b>Figure 4.3.1</b> : Backend script to pre-process (1)	<b>27</b>

<b>Figure 4.3.2</b>	: Backend script to pre-process (2)	<b>28</b>
<b>Figure 4.3.3</b>	: Backend script to pre-process (3)	<b>29</b>
<b>Figure 4.4.1</b>	: Script to create UI using Streamlit (1)	<b>30</b>
<b>Figure 4.4.2</b>	: Script to create UI using Streamlit (2)	<b>31</b>
<b>Figure 5.1.1</b>	: Correctly identifying Angry Emotion	<b>32</b>
<b>Figure 5.1.2</b>	: Correctly identifying Disgust Emotion	<b>33</b>
<b>Figure 5.1.3</b>	: Correctly identifying Fear Emotion	<b>34</b>
<b>Figure 5.1.4</b>	: Correctly identifying Happy Emotion	<b>35</b>
<b>Figure 5.1.5</b>	: Correctly identifying Neutral Emotion	<b>36</b>
<b>Figure 5.1.6</b>	: Correctly identifying Sad Emotion	<b>37</b>
<b>Figure 5.1.7</b>	: Correctly identifying Surprise Emotion	<b>38</b>
<b>Figure 5.2.1</b>	: Wrongly classifying Disgust as Neutral	<b>39</b>
<b>Figure 5.2.2</b>	: Wrongly classifying Neutral as Sad	<b>40</b>
<b>Figure 5.3.1</b>	: Application Failing to detect face	<b>41</b>
<b>Figure 6.1.1</b>	: Accuracy of CNN model against AffectNet Test	<b>42</b>
<b>Figure 6.2.1</b>	: Accuracy of Deep Face model against AffectNet Test	<b>43</b>
<b>Figure 6.2.2</b>	: Accuracy of Deep Face model with AffectNet Train (1)	<b>44</b>
<b>Figure 6.2.3</b>	: Accuracy of Deep Face model with AffectNet Train (2)	<b>45</b>
<b>Figure 6.2.4</b>	: Accuracy of Deep Face model with AffectNet Train (3)	<b>46</b>
<b>Figure 6.3.1</b>	: Accuracy of MTCNN based FER Model	<b>47</b>
<b>Figure 6.4.1</b>	: Accuracy of K-MEANS Clustering	<b>48</b>
<b>Figure 6.5.1</b>	: Compare & Contrast of Emotion Detection Models	<b>48</b>
<b>Figure 6.6.1</b>	: Emotion-wise Accuracies of all models	<b>49</b>

## 1. Introduction

Music is universal and easily accessible in our everyday lives, thanks to the tremendous growth of digital music technologies. People listen to music on various sources to improve their mood and enliven the environment. However, due to the enormous number of songs accessible, people are frequently perplexed when selecting a track.

People enjoy listening to different genres of music according to their mood. So, Naive recommendations do not work as intended. This necessitates the development of a context-sensitive music recommendation system. Context-aware recommendation systems also need accumulation of additional data, which can be difficult to come by but has shown to be worthwhile in some cases.

Context-aware music recommendation systems build on the principle that various personality traits connect with distinct item attributes (for example, acoustic qualities or musical genres) and that users in different emotional states and moods prefer various sorts of things.

Emotion-aware Recommendation systems have yielded encouraging results, demonstrating that using emotional states and reactions, recommendation algorithms may improve prediction accuracy and refine personalisation.

The emotions of users may be identified in a variety of ways. However, they all fall into one of two types. Face expressions, keystrokes, and mouse-click patterns are all examples of implicit ways of detecting emotion. Explicit methods, on the other hand, take direct input from the user. For the identification of emotion, we choose to utilise an approach based on facial expressions.

In the next sections, several Facial-Emotion Identification techniques, machine-learning models, and recommendation algorithms will be reviewed and contrasted. We further demonstrate that the emotion-identification method used has no effect on the recommendation algorithms.

## 1.1. Facial Emotion Identification

The task of recognising emotion from a user's face data is known as facial emotion identification. A Neural Network is typically used to accomplish the task. A Neural Network is a network of Interconnected perceptrons.

### Working of an Artificial Neural Network:

A single neuron can either fire or not, so its activity level can be represented as a one or a zero. One neuron receives its information from a bunch of other neurons, but because the strength of these connections varies, each one might be assigned a different weight. Some connections are excitatory, so they have positive weights, while others are inhibitory, so they have negative weights.

The approach to find out whether a certain neuron fires is to multiply the activity of each input neuron by its weight, then add them all together. The neuron fires if their sum is larger than a value called the bias; if it is less than that, the neuron does not fire.

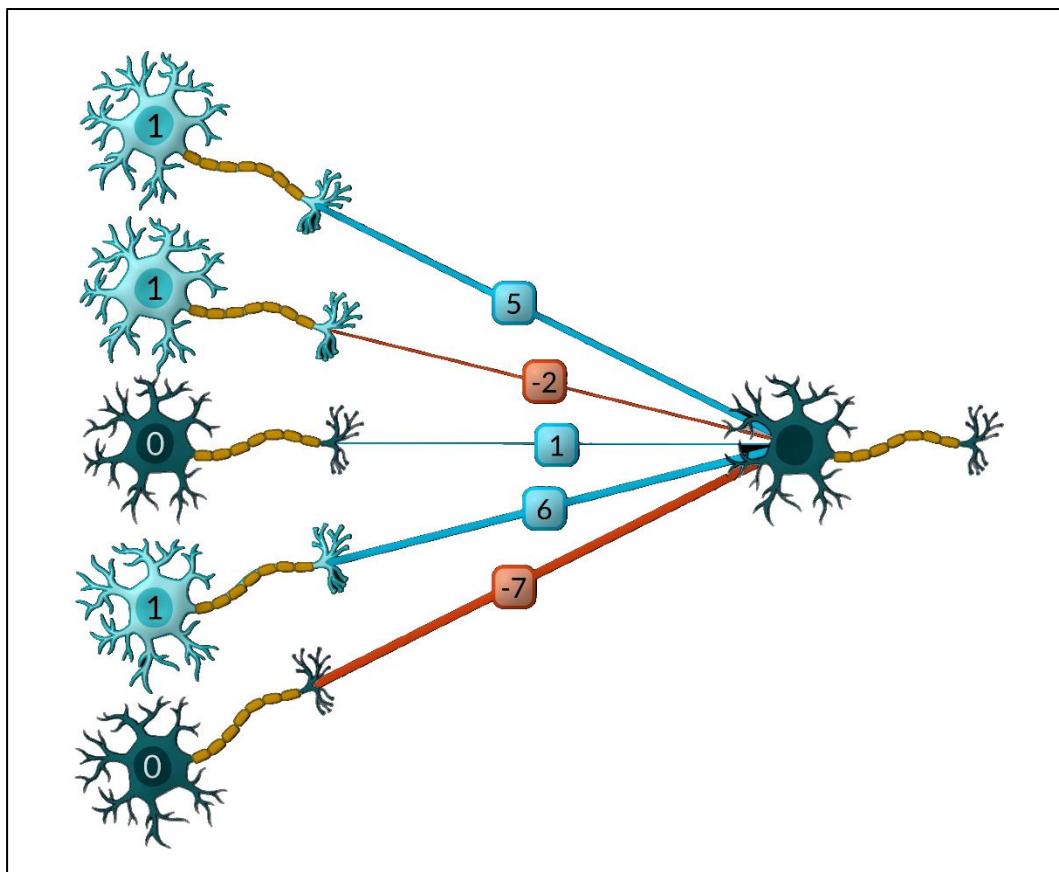


Figure 1.1.1: Basic model of how Neurons in Artificial Neural Networks work.

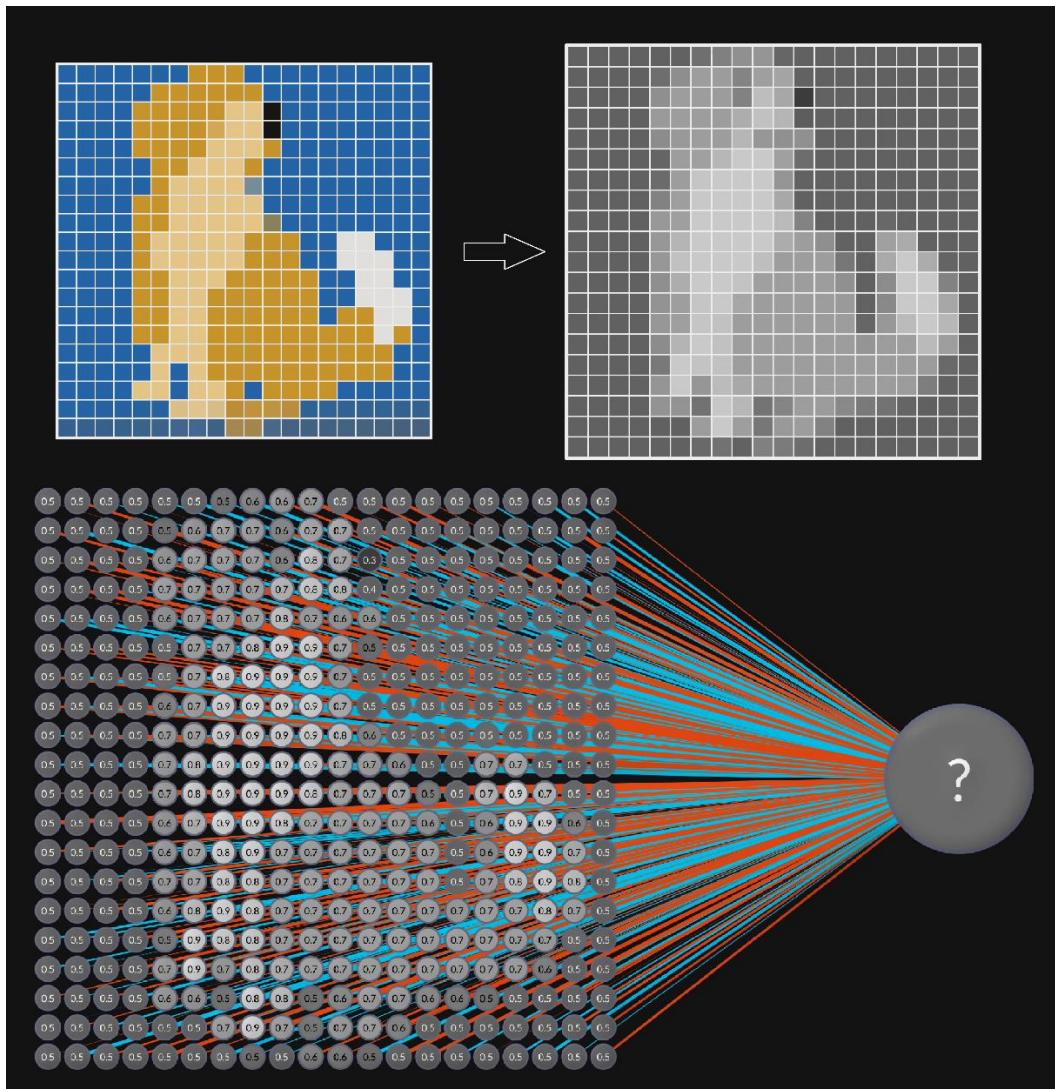


Figure 1.1.2: Processing of a digital image by a Neural network.

The Neural Network accepts an  $M \times N$  matrix as input, which represents a picture with  $M \times N$  pixels. Each pixel functions like a neuron, with the brightness of the pixel determining its activation. In most cases, the activation is normalised to a number between zero and one. Each of these neurons is linked to the next collection of neurons via their own adjustable weights, which is referred to as a layer.

Depending on the bias and the summation of the product of weights and activation, each of the neurons in the following layer will either fire or not. At the end of the process, the output layer will have a number of neurons, generally only one of which will fire.

The outcome of the Neural Network is represented by the output layer. Depending on what the Neural Network interprets the image as, just one neuron in the output layer will fire. The training algorithm will alter the weights associated with each neuron whenever an output neuron fires when it shouldn't. Similarly, when an output neuron does not fire when it should, the weights are modified.

The weights associated with the neurons in the hidden layer are continuously adjusted by the algorithm until the neural network properly recognises all of the training pictures. The process of adjusting the weights is called Back Propagation.

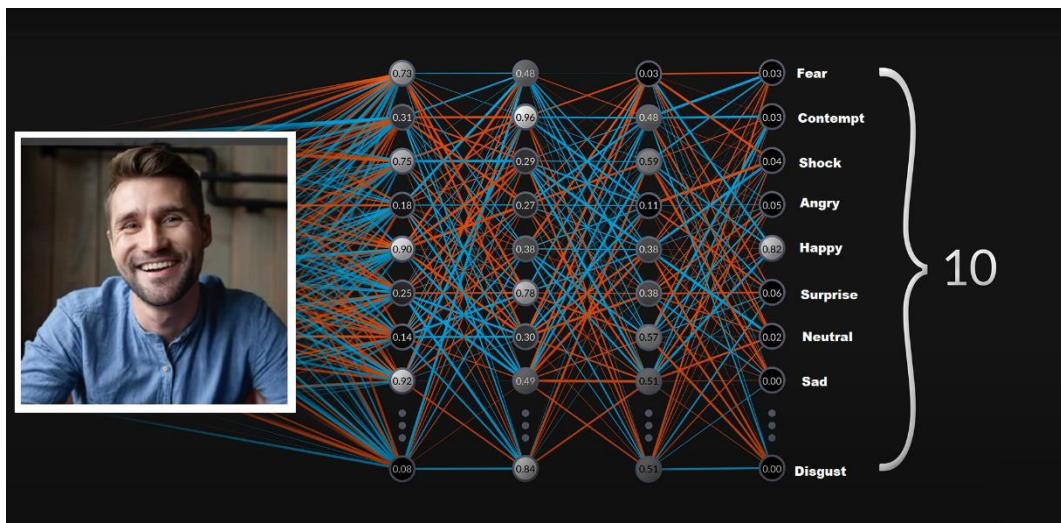


Figure 1.1.3: Emotion Identification by a Neural Network.

For more precise training, the Neural Network becomes denser. Processing a single image takes roughly 700,000,000 mathematical operations because of all the large Matrix Multiplications. This creates a huge need for faster processing equipment, particularly GPUs (graphics processing units) that are designed for rapid, simultaneous computations.

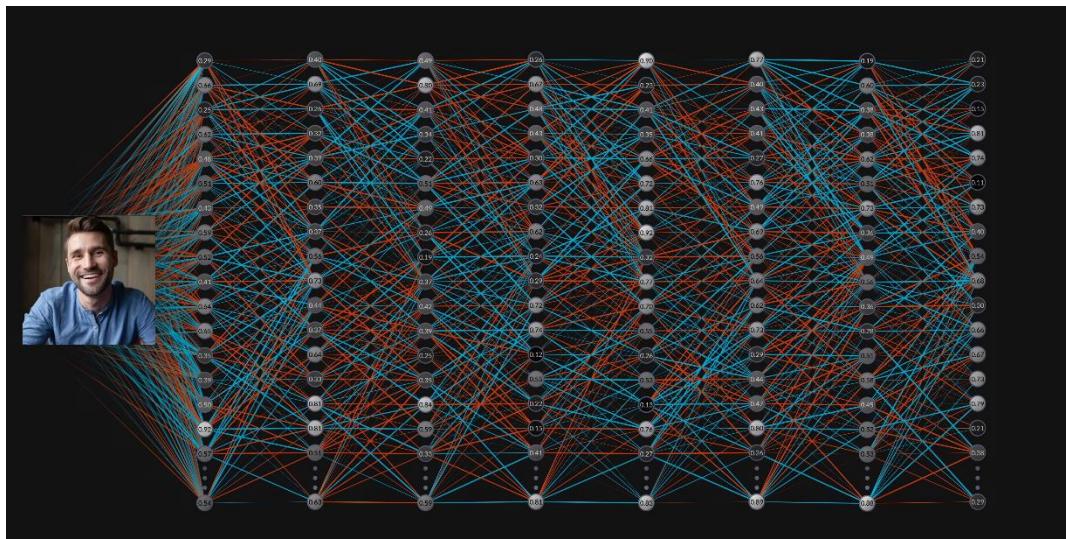


Figure 1.1.4: A Neural Network involving more hidden layers.

The contrasts, benefits, and limitations of the top three Neural Networks, **Convolutional Neural Networks**, **Deep Neural Networks**, and **Multi-task Cascaded Convolutional Neural Networks** are the topic of discussion of our project.

## 1.2. Music Recommendation

The ordinary individual makes a range of purposeful media consumption selections on any given day. As we manoeuvre these omnipresent options in our increasingly diversified environment, recommendation system algorithms that have become pervasive in our lives point us in the right direction.

The fundamental goal of recommendation algorithms is to assess user data so that personalised recommendations may be made.

### 1.2.1. Collaborative Filtering

To provide suggestions, collaborative filtering looks for similarities between users and products at the same time. Collaborative filtering is often used with very big data sets. The collaborative filtering strategy has the benefit of not relying on machine comprehensible material, thus it can properly recommend complicated items without requiring a "knowledge" of the item itself.

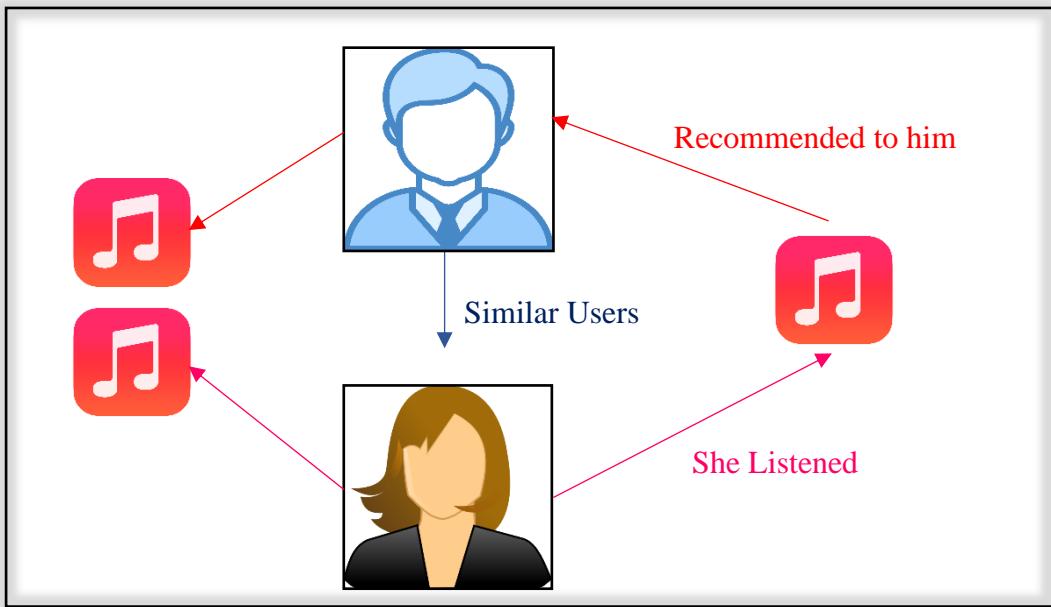


Figure 1.2.1: Collaborative Filtering

### 1.2.2. Content-based Filtering

Another prominent way for constructing recommender systems is content-based filtering. Based on the user's prior activities or explicit input, content-based filtering uses item attributes to recommend additional items that are similar to what they like.

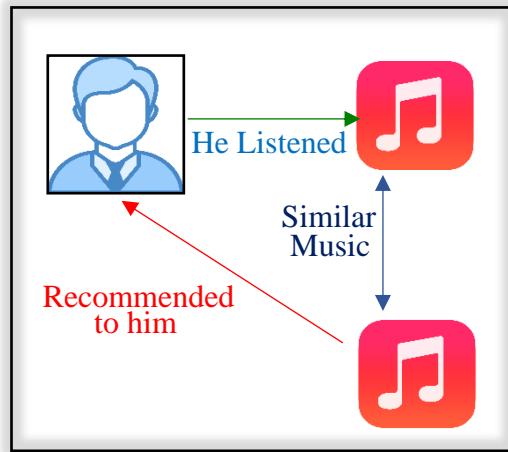


Figure 1.2.2: Content-based Filtering

## 2. Overview of Proposed System

The Proposed system consists of two phases:

2.1. Emotion Identification

2.2. Music Recommendation

### 2.1. Emotion Identification

Neural Networks are typically used for face classification and emotion identification. After extensive research and experimentation on different neural network models such as the CNN model, MTCNN model and DNN model, the following observations were made:

Parameter	CNN	Deep Face (DNN)	FER (MTCNN)
Accuracy	High	Low	High
Train Time	Low	High	High
Validation Time	Low	High	High
Advantages	Speed	Light Weight	Self-alignment of Face
Disadvantages	Large Training Data	Low Accuracy	High Train time
Common	Unable to detect rare facial expressions like Disgust		

Table 2.1.1: Compare & Contrast of Different Emotion Detection Models.

The MTCNN model has the highest accuracy compared to other models. Although the train time of MTCNN is high, it can be reduced using GPUs. Hence, we propose the MTCNN model for emotion identification.

In the proposed system, the MTCNN model takes the user's face as input and outputs the dominant emotion detected.

## 2.2. Music Recommendation

Content-based filtering is used to recommend music. Since there are no publicly available datasets on users' emotions while listening to songs, collaborative filtering cannot be used. Hence content-based filtering approach is proposed for music recommendation.

The proposed approach involves identifying the emotion of each song from the MuSe dataset. Each track in the dataset has valence, arousal, and dominance values, which represent the overall emotion of the song. K-means clustering is done after determining the initial centroids based on the VAD range of the dataset.

After clustering, we get 7 clusters, each representing one of the following seven emotions: happy, sad, angry, neutral, surprise, disgust, and fear. All the tracks with the same emotion get grouped into the corresponding cluster. Based on this, the songs in the dataset get annotated with their corresponding emotion.

Based on the emotion identified from the previous phase and using the tagged music dataset, top k tracks with the same emotion get recommended to the user. These get displayed as Spotify widgets, which can be played by clicking the widget.

### 3. System Design

The System consists of the following modules

1. Emotion Identification module
2. Music Tagging and Recommendation module
3. User Interface module

#### 3.1. Emotion Identification Module

In this phase, we detect and identify the user's emotion. A Multi-task Cascaded Convolutional Network (MTCNN) model, trained on the FER Dataset, is used to identify the emotion. MTCNN has an added advantage of self-alignment of face compared to other models.

The MTCNN model takes the face as input and outputs the dominant emotion identified. The neural network is composed of numerous layers of weighted nodes. Each layer processes the input and feeds it on to the next layer. The node weights are adjusted during propagation. The node with the largest weight in the output layer correlates to the dominant emotion.

The model performs on par with the current state-of-the-art systems and has reported accuracy of 92.19%.

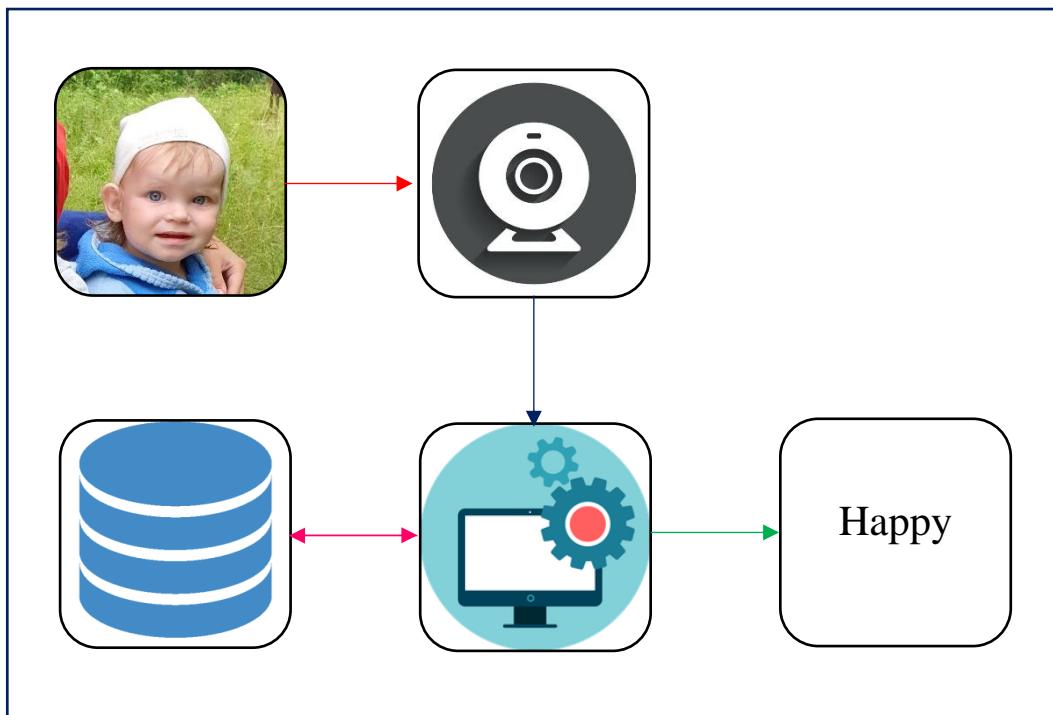


Figure 3.1.1: Emotion Identification Module.

### 3.2. Music Tagging and Recommendation Module

The next phase involves tagging music with its corresponding emotion. The MuSe dataset consists of over 90,000 tracks along with their valence, arousal, and dominance values. Valence represents the pleasantness dimension, Arousal represents the intensity dimension, and dominance represents the control dimension. However, the dataset doesn't consist of the emotion of the track. Hence each track needs to be tagged with its corresponding emotion.

Tagging involves clustering based on VAD values. VAD values are floating-point coordinates in 3-dimensional space, which represent the overall emotion of the song. VAD values are relative and can vary based on the range of the dataset. Based on the range of the MuSe dataset, VAD values for the seven emotions are determined.

K-Means clustering is used to group songs having same emotion. Initial centroids of the 7 clusters are obtained based on the VAD range. K-Means Clustering is then performed with the initial centroids. 7 Clusters are obtained at the end, each identifying an emotion.

Based on the emotion identified from module 1 and using the tagged music dataset, top k tracks with the same emotion get recommended to the user.

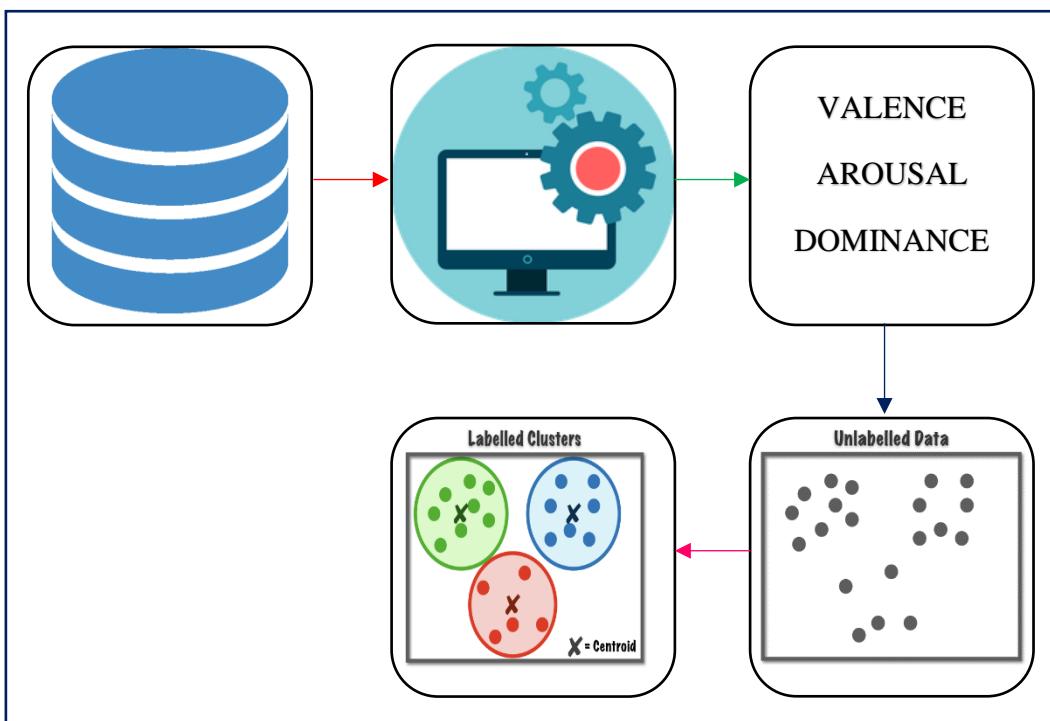


Figure 3.2.1: Music Recommendation Module.

### 3.3. User Interface Module

Through a web-based interface, the user interface module coordinates all interactions between users and our system. Users can capture their emotions using the camera module. Based on the user's emotion top 15 tracks are displayed as Spotify widgets, which get played on clicking the widget.

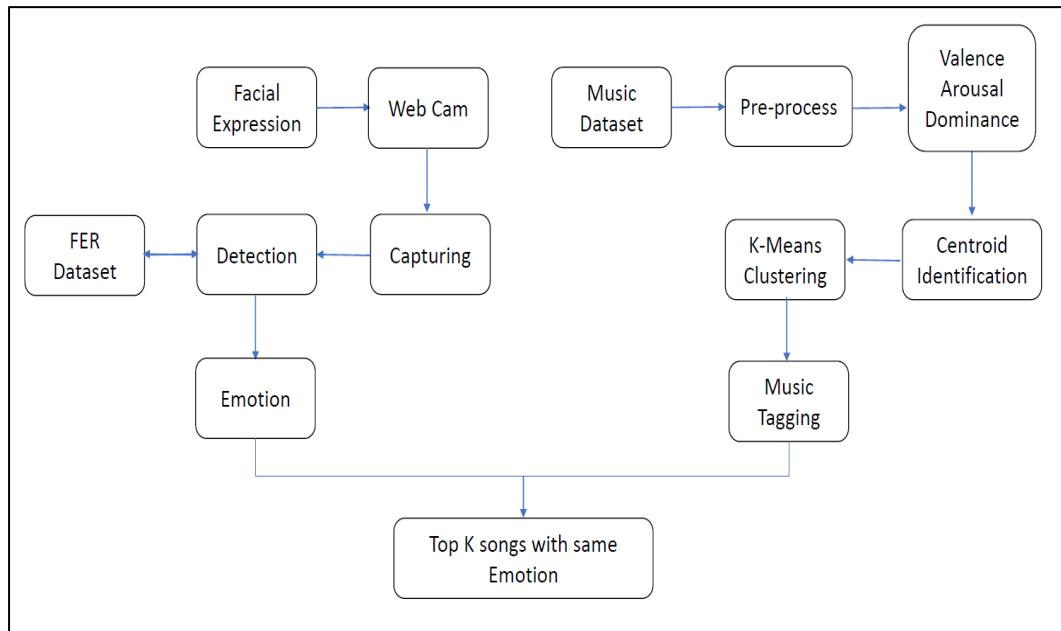


Figure 3.3.1 User-Interface Module.

## **4. Pseudo Algorithm/Implementation**

### **4.1. Emotion Identification**

This stage comprises training a neural network on a large number of visual pictures consisting of facial data to determine the emotion associated with each facial expression.

Rather than having to develop all of the algorithms from scratch, Python allows us to simply import the relevant modules and train models by giving the required inputs. As a result, we decided to use Python, a high-level programming language, to easily create and train Neural Networks.

#### **4.1.1. Modules Used**

- **Programming Language:** Python 3.10.4
- **Python Modules**
  1. FER
  2. DeepFace
  3. NumPy
  4. TensorFlow
  5. Sci-kit learn
  6. OpenCV
  7. Keras: Python deep learning API
  8. OS
  9. Time
  10. Sys Module
  11. Shutil
- **Operating System:** Ubuntu 20.04.4 LTS (Focal Fossa)
- **Source-code Editor:** Microsoft Visual Studio Code

#### 4.1.2. CNN For Facial Emotion Identification

Convolutional Neural Networks use the hierarchical pattern in data to build extremely sophisticated patterns from smaller and simpler patterns imprinted in its filters. As a corollary, CNNs reside on the lower end of the interconnectedness and complexity spectrum.

Compiling a model, which is effectively a neural network, is the first step in implementing a Convolutional Neural Network. The following are the parameters that have to be defined when compiling a model.

- **Optimizers** : Define Weights and Learning Rate
- **Losses** : Choose what to minimize when training
- **Metrics** : Function used to Judge the performance of model
- **Train Set** : Set of Labelled Images used to train the Neural Network
- **Validation Set** : Set of Labelled Images used to validate the Neural Network

We chose the following settings to train our Neural Network.

Parameter	Value
Optimizer	'adam'
Losses	'categorical_crossentropy'
Metrics	'accuracy'
Train Set	FER 2013 Dataset
Validation Set	AffectNet Dataset

Table 4.1.2.1: Arguments passed to Keras Model.

Post Training, the model was tested against the AffectNet Dataset, consisting of around 49K labelled, coloured images of size 224 x 224. The script used for testing the accuracy of the Trained model follows.

```

1 import numpy as np
2 import tensorflow as tf
3 from tensorflow.keras.preprocessing.image import (
4     ImageDataGenerator,
5 )
6
7 from sklearn.metrics import (
8     confusion_matrix,
9     classification_report,
10 )
11 from sklearn.preprocessing import LabelBinarizer
12 from sklearn.metrics import roc_curve, auc, roc_auc_score
13
14 from IPython.display import clear_output
15 import warnings
16
17 warnings.filterwarnings("ignore")
18 import cv2
19 from os import listdir
20 import shutil
21
22 SEED = 12
23 IMG_HEIGHT = 64
24 IMG_WIDTH = 64
25 BATCH_SIZE = 64
26 EPOCHS = 30
27 FINE_TUNING_EPOCHS = 20
28 LR = 0.01
29 NUM_CLASSES = 7
30 EARLY_STOPPING_CRITERIA = 3
31 CLASS_LABELS = [
32     "Anger",
33     "Disgust",
34     "Fear",
35     "Happy",
36     "Neutral",
37     "Sad",
38     "Surprise",
39 ]
40
41 path = (
42     "../Datasets/Facial Emotion Recognition/AffectNet/Annotated/images/"
43 )
44
45 model = tf.keras.models.load_model("best_model.h5")
46
47
48 def get_emotion(path: str) -> str:
49     preprocess_fun = tf.keras.applications.densenet.preprocess_input
50     test_datagen = ImageDataGenerator(
51         rescale=1.0 / 255,
52         validation_split=0.2,
53         preprocessing_function=preprocess_fun,
54     )
55     test_generator = test_datagen.flow_from_directory(
56         directory=path,
57         target_size=(IMG_HEIGHT, IMG_WIDTH),
58         batch_size=BATCH_SIZE,
59         shuffle=False,
60         color_mode="rgb",
61         class_mode="categorical",
62         seed=12,
63     )

```

Figure 4.1.2.1: Script to find the Accuracy of Trained CNN Model (1).

```

63     )
64     results = model.predict(test_generator)
65     freq = [0] * 7
66     for result in results:
67         freq[np.argmax(result)] += 1
68
69     return CLASS_LABELS[np.argmax(freq)]
70
71
72 def main():
73     labels = listdir(path)
74
75     total_correct = 0
76     total_present = 0
77
78     for label in labels:
79         if label == "Contempt":
80             continue
81         else:
82             destination = "Stream/Stream/"
83
84             label_correct = 0
85             label_present = 0
86             print(
87                 "Testing:",
88                 label,
89                 flush=True,
90                 file=open("LOG.txt", "a"),
91             )
92             for image in listdir(path + label):
93                 shutil.copy(
94                     path + label + "/" + image,
95                     destination + "cap.jpg",
96                 )
97             got = get_emotion("Stream")
98             expected = label
99             if got.lower() == expected.lower():
100                 label_correct += 1
101                 total_correct += 1
102             label_present += 1
103             total_present += 1
104             print(
105                 "Accuracy for",
106                 label,
107                 ":",
108                 label_correct / label_present,
109                 flush=True,
110                 file=open("LOG.txt", "a"),
111             )

```

Figure 4.1.2.2: Script to find the Accuracy of Trained CNN Model (2).

```

111         )
112         print(
113             "Label Present:",
114             label_present,
115             flush=True,
116             file=open("LOG.txt", "a"),
117         )
118         print(
119             "Label Correct:",
120             label_correct,
121             flush=True,
122             file=open("LOG.txt", "a"),
123         )
124         print("", flush=True, file=open("LOG.txt", "a"))
125     print(
126         "Total Accuracy:",
127         total_correct / total_present,
128         flush=True,
129         file=open("LOG.txt", "a"),
130     )
131     print(
132         "Total Present:",
133         total_present,
134         flush=True,
135         file=open("LOG.txt", "a"),
136     )
137     print(
138         "Total Correct:",
139         total_correct,
140         flush=True,
141         file=open("LOG.txt", "a"),
142     )
143
144
145 main()
146

```

Figure 4.1.2.3: Script to find the Accuracy of Trained CNN Model (3).

#### 4.1.3. Deep Neural Network

An artificial neural network (ANN) having numerous layers between the input and output layers is called a deep neural network (DNN). Deep Neural Networks don't loop back on themselves. As a result, they are a form of feedforward network, in which data flows from input to output.

Fortunately, a module for Facial Emotion Identification is included in Deep Face, a lightweight face recognition and facial attribute analysis framework. This module contains Deep Neural Network models that have been trained on several facial images.

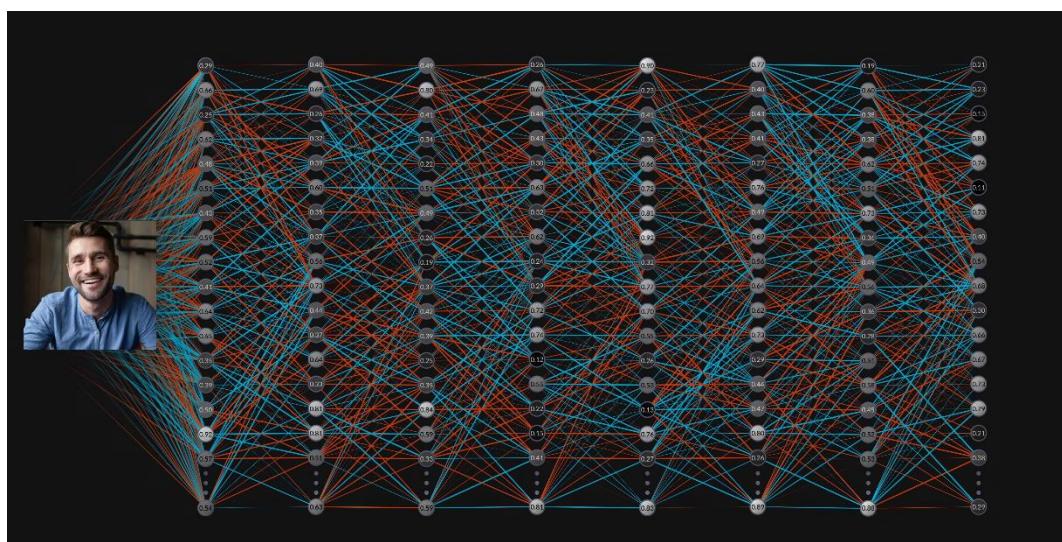


Figure 4.1.3.1: Deep Neural Network.

The Deep Face module makes detecting emotion through facial expression quite simple. The sample script that follows will probably be sufficient to identify the emotion.

```
● ● ●

1 import cv2
2 from deepface import DeepFace
3
4 def get_emotion(img) -> str:
5     try:
6         result = DeepFace.analyze(img, actions = ["emotion"])
7         result = result["dominant emotion"]
8     except:
9         return "No face detected"
10
11 if __name__ == '__main__':
12     img_path = '/pictures/Test.jpg'
13     img = cv2.imread(img_path)
14     emotion = get_emotion(img)
15     print("Identified Emotion: ", emotion)
16
```

Figure 4.1.3.2: Sample Snippet demonstrating the Deep Face Module.

## Script used for Testing the accuracy of the DNN based Deep Face Model:

```
 1 import cv2
 2 from deepface import DeepFace
 3 import os
 4
 5 path = (
 6     "../Datasets/Facial Emotion Recognition/AffectNet/Annotated/images/"
 7 )
 8
 9
10 def get_emotion(img) -> str:
11     try:
12         result = DeepFace.analyze(img, actions=["emotion"])
13         return result["dominant_emotion"]
14     except:
15         return "No face detected"
16
17
18 def from_file(img_path: str) -> str:
19     array = cv2.imread(img_path)
20     emotion = get_emotion(array)
21     return emotion
22
23
24 def validate():
25     labels = os.listdir(path)
26     total_correct = 0
27     total_present = 0
28     for label in labels:
29         emotion_correct = 0
30         emotion_total = 0
31         if label == "Contempt":
32             continue
33
34         actual_emotion = label.lower()
35         if actual_emotion == "anger":
36             actual_emotion = "angry"
37
38         print("Currently Testing " + str(label), flush=True)
39         print(flush=True)
40
41         for img in os.listdir(path + label):
42             emotion = from_file(path + label + "/" + img)
43             if emotion.lower() == actual_emotion:
44                 emotion_correct += 1
45                 total_correct += 1
46             emotion_total += 1
47             total_present += 1
```

Figure 4.1.3.3: Script used for Testing the accuracy of the Deep Face Model (1).

```

47         total_present += 1
48
49         if total_present % 1000 == 0:
50             print(
51                 "Processed "
52                 + str(total_present)
53                 + " images so far",
54                 flush=True,
55             )
56             print(
57                 "Current Accuracy: "
58                 + str(total_correct / total_present),
59                 flush=True,
60             )
61             print(flush=True)
62
63             print(
64                 "Accuracy for "
65                 + label
66                 + ": "
67                 + str(emotion_correct / emotion_total),
68                 flush=True,
69             )
70             print(
71                 "Details: "
72                 + str(emotion_correct)
73                 + "/"
74                 + str(emotion_total),
75                 flush=True,
76             )
77             print(flush=True)
78
79             print(
80                 "Total Accuracy: " + str(total_correct / total_present),
81                 flush=True,
82             )
83             print("Total Correct: " + str(total_correct), flush=True)
84             print("Total Present: " + str(total_present), flush=True)
85             print(flush=True)
86
87
88     if __name__ == "__main__":
89         validate()
90

```

Figure 4.1.3.4: Script used for Testing the accuracy of the Deep Face Model (2).

#### 4.1.4. Multi-Task Cascaded Neural Network

MTCNN stands for Multi-task Cascaded Convolutional Networks, which was created as a solution for both face identification and face alignment. The method includes three levels of convolutional networks that can detect faces and landmarks such as the eyes, nose, and mouth.

MTCNN has the benefit of automatically aligning poorly aligned faces. The precision may be enhanced even further this way. All of the slightly misaligned photos may now be fully aligned, and a rudimentary model can be used to identify face emotions.

A model had been trained and compiled for the Facial Emotion Recognition challenge (FER). The FER module, like the deep face module, is publicly available through Pip, Python's package installer. The sample below will probably be sufficient to identify the emotion through the FER's Emotion Detector.



```
1 from fer import FER
2 import cv2
3
4 img = cv2.imread("Sample.jpg")
5 detector = FER()
6 detector.detect_emotions(img)
```

Figure 4.1.4.1: Sample Snippet demonstrating MTCNN Model.

Script used for Testing the accuracy of the MTCNN based FER model follows. The emotion ‘Contempt’ is considered as a rare emotion; hence it is skipped when evaluation the Accuracy of the model.

```

● ○ ●

1  from fer import FER
2  import cv2
3  from sys import argv
4  from os import listdir
5
6  path = (
7      "../Datasets/Facial Emotion Recognition/AffectNet/Annotated/images/"
8  )
9
10
11 def get_emotion(img_path: str) -> str:
12     try:
13         img = cv2.imread(img_path)
14         detector = FER()
15         emotions = detector.detect_emotions(img)
16         # print(emotions)
17         weights = emotions[0]["emotions"]
18         weights = [(weights[emotion], emotion) for emotion in weights]
19         weights.sort()
20         return weights[-1][1]
21     except:
22         return "No face"
23
24
25 def main():
26     labels = listdir(path)
27
28     total_correct = 0
29     total_present = 0
30
31     for label in labels:
32         if label == "Contempt":
33             continue
34         else:
35             label_correct = 0
36             label_present = 0
37             print("Testing:", label, flush=True)
38             for image in listdir(path + label):
39                 got = get_emotion(path + label + "/" + image)
40                 expected = label.lower()
41                 if expected == "anger":
42                     expected = "angry"
43                 if got == expected:
44                     label_correct += 1
45                     total_correct += 1
46                     label_present += 1
47                     total_present += 1
48             accuracy = label_correct / label_present
49             print("Accuracy for", label, ":", accuracy, flush=True)
50             print("Label Present:", label_present, flush=True)
51             print("Label Correct:", label_correct, flush=True)
52             print("", flush=True)
53             print(
54                 "Total Accuracy:",
55                 total_correct / total_present,
56                 flush=True,
57             )
58             print("Total Present:", total_present, flush=True)
59             print("Total Correct:", total_correct, flush=True)
60
61
62 main()
63

```

Figure 4.1.4.2: Script to find Accuracy of MTCNN Model.

## 4.2. Music Recommendation

Content-based filtering is used to recommend music. Since there are no publicly available datasets on users' emotions while listening to songs, collaborative filtering cannot be used. Hence content-based filtering approach is proposed for music recommendation.

K-Means clustering was done on the MuSe Dataset, as mentioned in the System Design portion of this publication. The Silhouette Score for Clustering was calculated using the snippets below. It is almost impossible to determine the accuracy since it requires real-time input from users' recommendations.

```
● ○ ●

1 import os
2
3 os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
4
5 import numpy, pandas
6 from sklearn.cluster import KMeans
7 from sklearn.metrics import silhouette_score
8 import UTILS
9
10 constants = UTILS.constants
11
12
13 def get_data_frame() -> pandas.DataFrame:
14     print("Parsing Dataset...")
15     path = constants["Music_Dataset"]
16     dataframe = pandas.read_csv(path)
17     dataframe = dataframe[dataframe["spotify_id"].notna()]
18     return dataframe
19
20
21 def get_static_centroids() -> tuple:
22     coordinates = constants["coordinates"]
23     coordinates = [(key, value) for key, value in coordinates.items()]
24     centroids = [list(item[1]) for item in coordinates]
25     centroids = numpy.array(centroids)
26     labels = [item[0] for item in coordinates]
27     return (labels, centroids)
28
29
30 def split_dataframe(dataframe: pandas.DataFrame, train_size=0.80):
31     train_percent = int(train_size * 100)
32     test_percent = 100 - train_percent
33     info = "Splitting Dataframe into Train({}\%) and Test({}\%) Set..."
34     info = info.format(train_percent, test_percent)
35     print(info)
36     train_dataframe = dataframe.sample(frac=train_size, random_state=50)
37     test_dataframe = dataframe.drop(train_dataframe.index)
38     return (train_dataframe, test_dataframe)
39
```

Figure 4.2.1: Script to cluster using K-MEANS (1).

```

39
40
41 def KMeansAll(dataframe: pandas.DataFrame):
42     print("Performing K-Means Clustering without initial centroids...")
43     x = dataframe.iloc[:, 5:8].values
44     k_means_optimum = KMeans(
45         n_clusters=7,
46         n_init=1,
47         init="k-means++",
48         random_state=50,
49         tol=1e-8,
50     )
51     y = k_means_optimum.fit_predict(x)
52     # dataframe['cluster'] = y
53     score = silhouette_score(x, y)
54     print("Silhouette score: ", score)
55
56
57 def KMeans_given_Initial_Centroids(
58     dataframe: pandas.DataFrame, centroids: numpy.array
59 ):
60     print("Performing K-Means Clustering with initial centroids...")
61     x = dataframe.iloc[:, 5:8].values
62     k_means_optimum = KMeans(
63         n_clusters=7,
64         n_init=1,
65         init=centroids,
66         random_state=50,
67         tol=1e-8,
68     )
69     y = k_means_optimum.fit_predict(x)
70     # dataframe['cluster'] = y
71     score = silhouette_score(x, y)
72     print("Silhouette score: ", score)
73
74
75 def KMeans_divided_dataset(
76     whole: pandas.DataFrame,
77     train: pandas.DataFrame,
78     test: pandas.DataFrame,
79 ):
80     print("Performing K-Means Clustering on Train and Test Dataset...")
81     x = whole.iloc[:, 5:8].values
82     k_means_optimum = KMeans(
83         n_clusters=7,
84         n_init=1,
85         init="k-means++",
86         random_state=50,
87         tol=1e-8,
88     )
89     y = k_means_optimum.fit_predict(x)
90     whole["cluster"] = y
91     original_centroids = k_means_optimum.cluster_centers_
92     original_centroids = sorted(
93         [list(triple) for triple in original_centroids]
94     )
95

```

Figure 4.2.2: Script to cluster using K-MEANS (2).

```

95
96     train_x = train.iloc[:, 5:8].values
97     k_means_train = KMeans(
98         n_clusters=7,
99         init="k-means++",
100        random_state=50,
101        tol=1e-8,
102    )
103    y_train = k_means_train.fit_predict(train_x)
104    train["cluster"] = y_train
105    train_centroids = k_means_train.cluster_centers_
106    train_centroids = sorted(
107        [list(triple) for triple in train_centroids]
108    )
109
110    total_correct = 0
111    total_present = 0
112
113    for row in test.itertuples():
114        valence, arousal, dominance = map(
115            float,
116            [
117                row.valence_tags,
118                row.arousal_tags,
119                row.dominance_tags,
120            ],
121        )
122        nearest1 = None
123        nearest_distance = 10**10
124        for i in range(len(original_centroids)):
125            x1, y1, z1 = original_centroids[i]
126            x2, y2, z2 = valence, arousal, dominance
127            distance_ = (x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2
128            if distance_ < nearest_distance:
129                nearest_distance = distance_
130                nearest1 = i
131
132        nearest2 = None
133        nearest_distance = 10**10
134        for i in range(len(train_centroids)):
135            x1, y1, z1 = train_centroids[i]
136            x2, y2, z2 = valence, arousal, dominance
137            distance_ = (x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2
138            if distance_ < nearest_distance:
139                nearest_distance = distance_
140                nearest2 = i
141        if nearest1 == nearest2:
142            total_correct += 1
143    total_present += 1
144
145    print("Total Rows in Test Set: ", total_present)
146    print("Total Correctly Classified: ", total_correct)
147    print("Accuracy: %.3f" % (100 * total_correct / total_present))
148
149

```

Figure 4.2.3: Script to cluster using K-MEANS (3).

```
148
149
150 if __name__ == "__main__":
151     dataframe = get_data_frame()
152
153     KMeansAll(dataframe.copy(deep=True))
154     print()
155
156     labels, centroids = get_static_centroids()
157     KMeans_given_Initial_Centroids(dataframe.copy(deep=True), centroids)
158     print()
159
160     train_df, test_df = split_dataframe(dataframe.copy(deep=True))
161     KMeans_divided_dataset(dataframe.copy(deep=True), train_df, test_df)
162
```

Figure 4.2.4: Script to cluster using K-MEANS (4).

### 4.3. The Backend

Python was used to create the application's backend. It primarily comprises of MuSe Dataset pre-processing algorithms, Emotion recognition tools, and K-Means Clustering utilities. The script also includes a list of Constants to assist reviewers.

```
● ● ●
1 import os
2
3 os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
4
5 import numpy, pandas, collections, fer, random
6 from sklearn.cluster import KMeans
7
8 frame = '<iframe src="https://open.spotify.com/embed/track/{}" '
9 frame += (
10     'width="260" height="380" frameborder="0" allowtransparency="true" '
11 )
12 frame += 'allow="encrypted-media"></iframe>'
13
14 constants = {
15     "Music_Dataset": "Datasets/Music Recommendation/muse_v3.csv",
16     "emotion_model": fer.FER(),
17     "frame": frame,
18     "coordinates": {
19         "angry": (2.50, 5.93, 5.14),
20         "happy": (8.21, 5.55, 7.00),
21         "surprise": (7.21, 7.54, 7.25),
22         "disgust": (1.69, 3.33, 4.46),
23         "fear": (2.97, 5.16, 2.87),
24         "sad": (2.40, 2.81, 3.84),
25         "neutral": (4.12, 3.38, 4.43),
26     },
27     "colors": {
28         "angry": "#ff3300",
29         "happy": "#007399",
30         "surprise": "#e13a37",
31         "disgust": "#663300",
32         "fear": "#666699",
33         "sad": "#6b6b47",
34         "neutral": "#862d2d",
35         "left": "black",
36     },
37 }
38
39
40 def detect_emotion(img: numpy.ndarray) -> str:
41     try:
42         emotion_detection_model = constants["emotion_model"]
43         emotions = emotion_detection_model.detect_emotions(img)
44         weights = emotions[0]["emotions"]
45         weights = [(weights[emotion], emotion) for emotion in weights]
46         weights.sort()
47         return weights[-1][1]
48     except:
49         return None
50
```

Figure 4.3.1: Backend Script to Pre-process MuSe Dataset (1).

```

50
51
52 def distance(
53     x1: float,
54     y1: float,
55     z1: float,
56     x2: float,
57     y2: float,
58     z2: float,
59 ) -> float:
60     return (x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2
61
62
63 def map_emotion(
64     valence: float, arousal: float, dominance: float
65 ) -> tuple:
66     min_distance = 10**10
67     closest_emotion = None
68     for emotion in constants["coordinates"]:
69         x2, y2, z2 = constants["coordinates"][emotion]
70         distance_to_emotion = distance(
71             valence, arousal, dominance, x2, y2, z2
72         )
73         if distance_to_emotion < min_distance:
74             min_distance = distance_to_emotion
75             closest_emotion = emotion
76     return (closest_emotion, min_distance)
77
78
79 def get_centroids() -> tuple:
80     coordinates = constants["coordinates"]
81     coordinates = [(key, value) for key, value in coordinates.items()]
82     centroids = [list(item[1]) for item in coordinates]
83     centroids = numpy.array(centroids)
84     labels = [item[0] for item in coordinates]
85     return (labels, centroids)
86
87
88 def pre_process_static() -> dict:
89     dataframe = pandas.read_csv(constants["Music_Dataset"])
90     dataframe = dataframe[
91         [
92             "track",
93             "artist",
94             "genre",
95             "spotify_id",
96             "valence_tags",
97             "arousal_tags",
98             "dominance_tags",
99         ]
100    ]
101
102    tracks = collections.defaultdict(list)
103
104    for index, row in dataframe.iterrows():
105        valence, arousal, dominance = map(
106            float,
107            [
108                row["valence_tags"],
109                row["arousal_tags"],
110                row["dominance_tags"],
111            ],
112        )

```

Figure 4.3.2: Backend Script to Pre-process MuSe Dataset (2).

```

112         )
113     spotify_id = row["spotify_id"]
114     if pandas.notna(spotify_id):
115         emotion, distance = map_emotion(valence, arousal, dominance)
116         tracks[emotion].append((spotify_id, distance))
117
118     for track in tracks:
119         tracks[track].sort(key=lambda x: x[-1])
120         tracks[track] = [item[0] for item in tracks[track]]
121
122     return tracks
123
124
125 def pre_process_cluster() -> dict:
126     data = pandas.read_csv(constants["Music_Dataset"])
127     data = data[data["spotify_id"].notna()]
128     labels, centroids = get_centroids()
129     x = data.iloc[:, 5:8].values
130     k_means_optimum = KMeans(
131         n_clusters=7,
132         n_init=1,
133         init=centroids,
134         random_state=50,
135         tol=1e-8,
136     )
137     y = k_means_optimum.fit_predict(x)
138     data["cluster"] = y
139     tracks = collections.defaultdict(list)
140     for index, row in data.iterrows():
141         spotify_id = row["spotify_id"]
142         cluster = row["cluster"]
143         tracks[labels[cluster]].append(spotify_id)
144
145     return tracks
146
147
148 # Uncomment the following line for
149 # static pre-processing
150 # Static pre-processing uses defined coordinates
151
152 # tracks = pre_process_static()
153
154 # The following relies on KMeans
155 # Clustering which gives better results
156
157 tracks = pre_process_cluster()
158
159
160 def get_top_k(emotion: str, k=15, sample_size=100) -> list:
161     top_k = random.sample(tracks[emotion][:sample_size], k)
162     return top_k
163

```

Figure 4.3.3: Backend Script to Pre-process MuSe Dataset (3).

## 4.4. The Frontend

Because the focus was on analysing the methodologies for constructing a Context-aware Recommendation System, little work was put into developing the Web Application's front-end. The Front-end of the application was built using Streamlit, a free Python module. Streamlit makes it very simple to code the front-end using Python's core capabilities.

```
● ● ●

1 import streamlit, UTILS, PIL, numpy
2 import streamlit.components.v1 as components
3
4
5 def populate_tracks(tracks: list) -> None:
6     tag = "<h4 style='text-align: center;'>Top Tracks</h4>"
7     streamlit.markdown(tag, unsafe_allow_html=True)
8     with streamlit.container():
9         col1, col2, col3 = streamlit.columns([3, 3, 3])
10        for i, track in enumerate(tracks):
11            if i % 3 == 0:
12                with col1:
13                    components.html(frame.format(track), height=400)
14                elif i % 3 == 1:
15                    with col2:
16                        components.html(frame.format(track), height=400)
17                else:
18                    with col3:
19                        components.html(frame.format(track), height=400)
20
21
22 def populate_emotion(emotion: str) -> None:
23     left_color = UTILS.constants["colors"]["left"]
24     left_span = '<span style = "color: {}>Identified Emotion: </span>'
25     left_span = left_span.format(left_color)
26
27     emotion_color = UTILS.constants["colors"][emotion]
28     right_span = '<span style = "color: {}>{}</span>'
29     right_span = right_span.format(emotion_color, emotion.capitalize())
30
31     tag = "<h2 style='text-align: center;'>{}</h2>"
32     tag = tag.format(left_span + right_span)
33     streamlit.markdown(tag, unsafe_allow_html=True)
34
35
```

Figure 4.4.1: Script to create User Interface using Streamlit (1).

```

22 def populate_emotion(emotion: str) -> None:
23     left_color = UTILS.constants["colors"]["left"]
24     left_span = '<span style = "color: {}">Identified Emotion: </span>'
25     left_span = left_span.format(left_color)
26
27     emotion_color = UTILS.constants["colors"][emotion]
28     right_span = '<span style = "color: {}">{}</span>'
29     right_span = right_span.format(emotion_color, emotion.capitalize())
30
31     tag = "<h2 style='text-align: center;'>{}</h2>"
32     tag = tag.format(left_span + right_span)
33     streamlit.markdown(tag, unsafe_allow_html=True)
34
35
36 if __name__ == "__main__":
37     tracks = UTILS.tracks
38     frame = UTILS.constants["frame"]
39     picture = streamlit.camera_input("")
40     if picture is not None:
41         picture = PIL.Image.open(picture)
42         pixels = numpy.array(picture)
43         emotion = UTILS.detect_emotion(pixels)
44         if emotion:
45             tracks = UTILS.get_top_k(emotion)
46             populate_emotion(emotion)
47             populate_tracks(tracks)
48         else:
49             tag = (
50                 '<h3 style="text-align: center;">No face detected</h3>'
51             )
52             streamlit.markdown(tag, unsafe_allow_html=True)
53

```

Figure 4.4.2: Script to create User Interface using Streamlit (2).

## 5. Results

### 5.1. Accurate Predictions

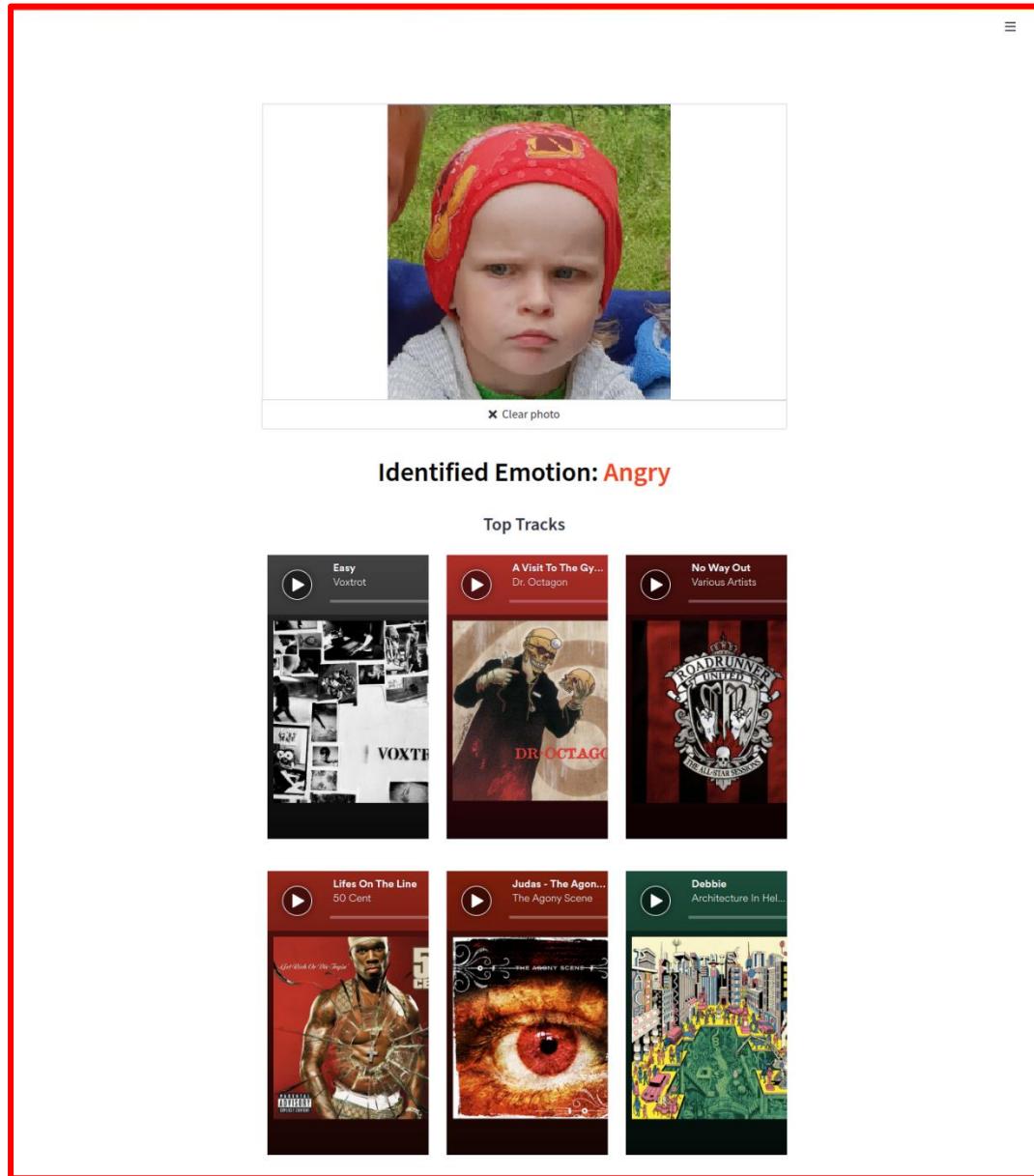


Figure 5.1.1: Application correctly Identifying Angry Emotion.

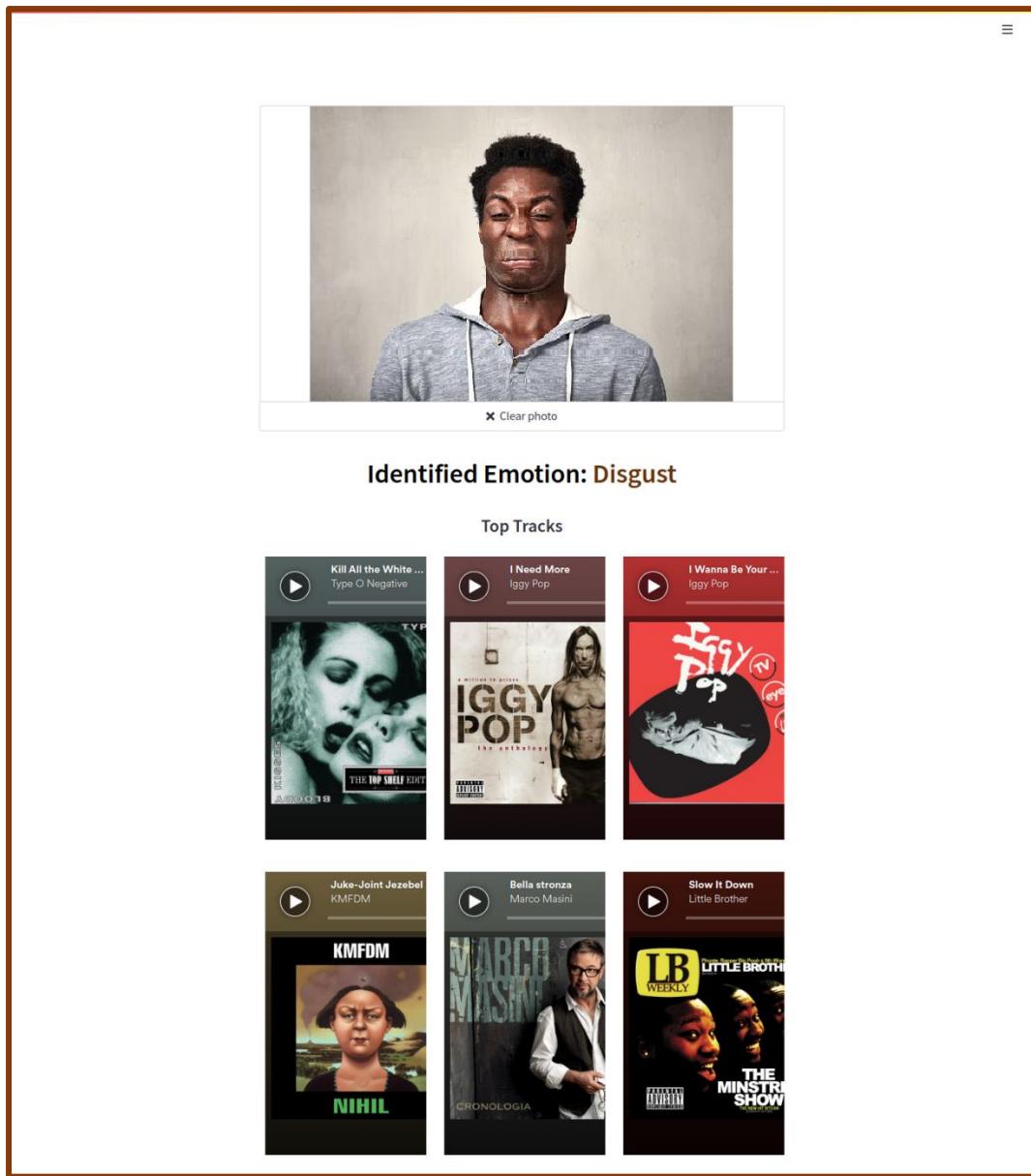


Figure 5.1.2: Application correctly Identifying Disgust Emotion.

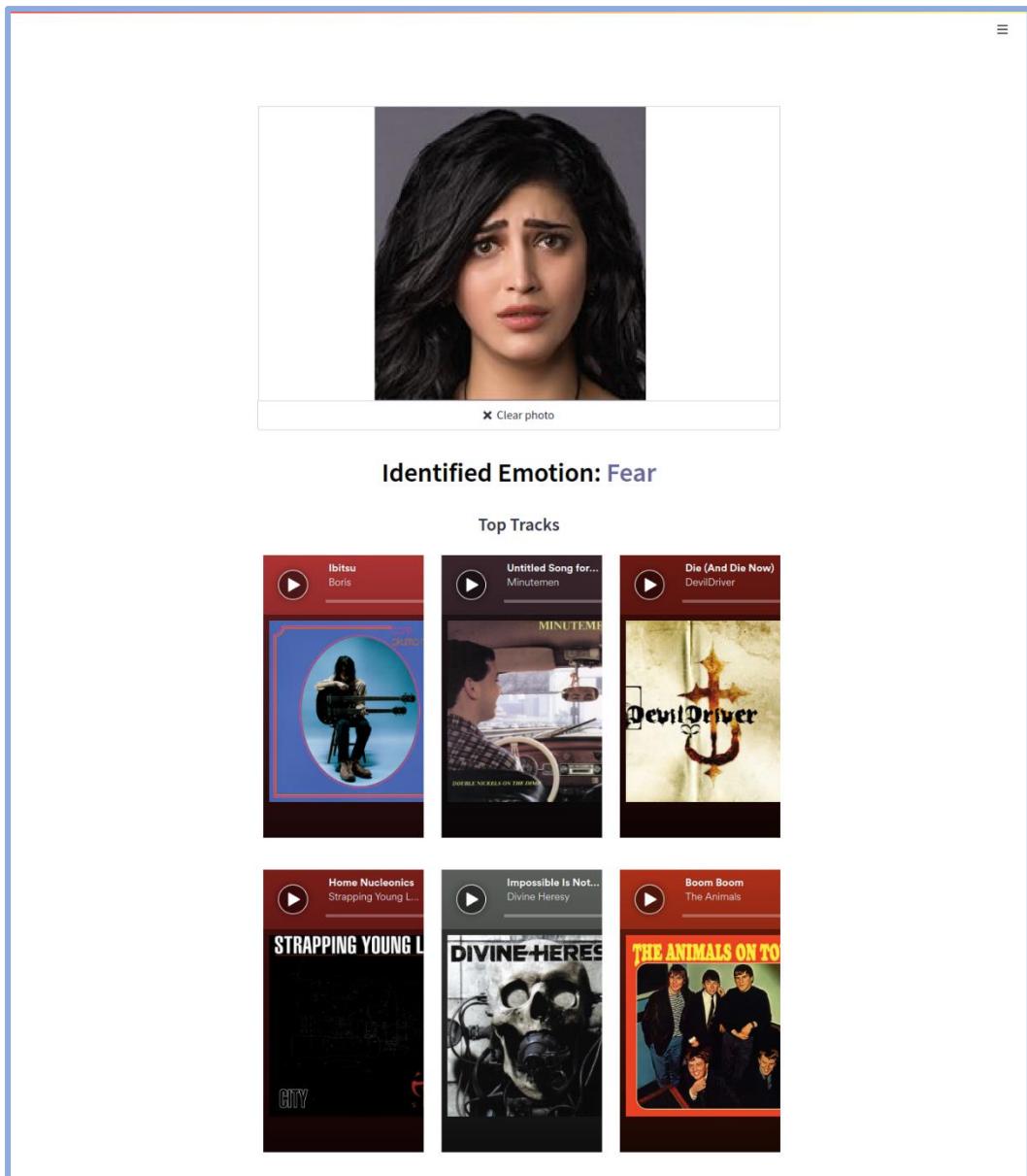


Figure 5.1.3: Application correctly Identifying Fear Emotion.

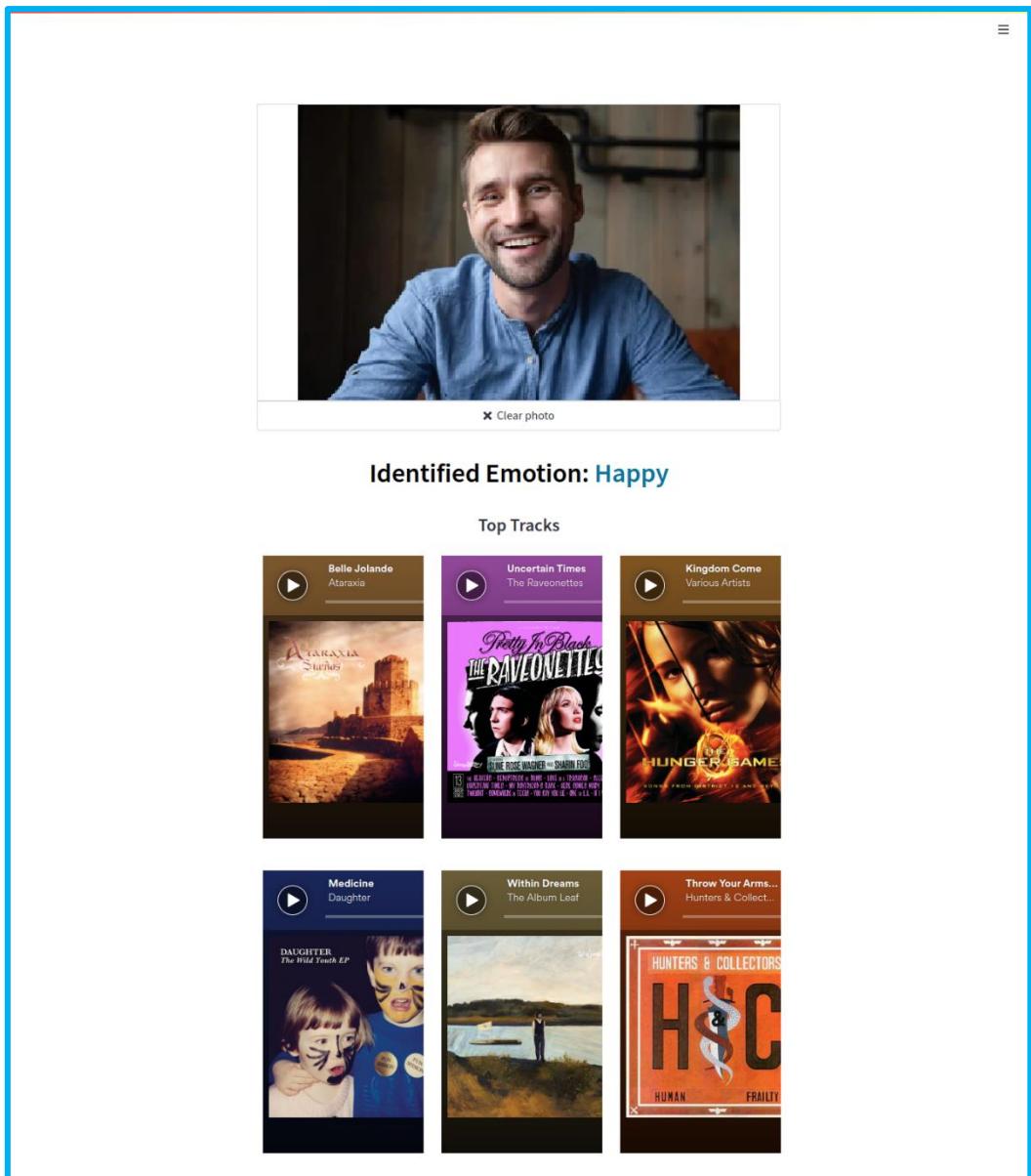


Figure 5.1.4: Application correctly Identifying Happy Emotion.

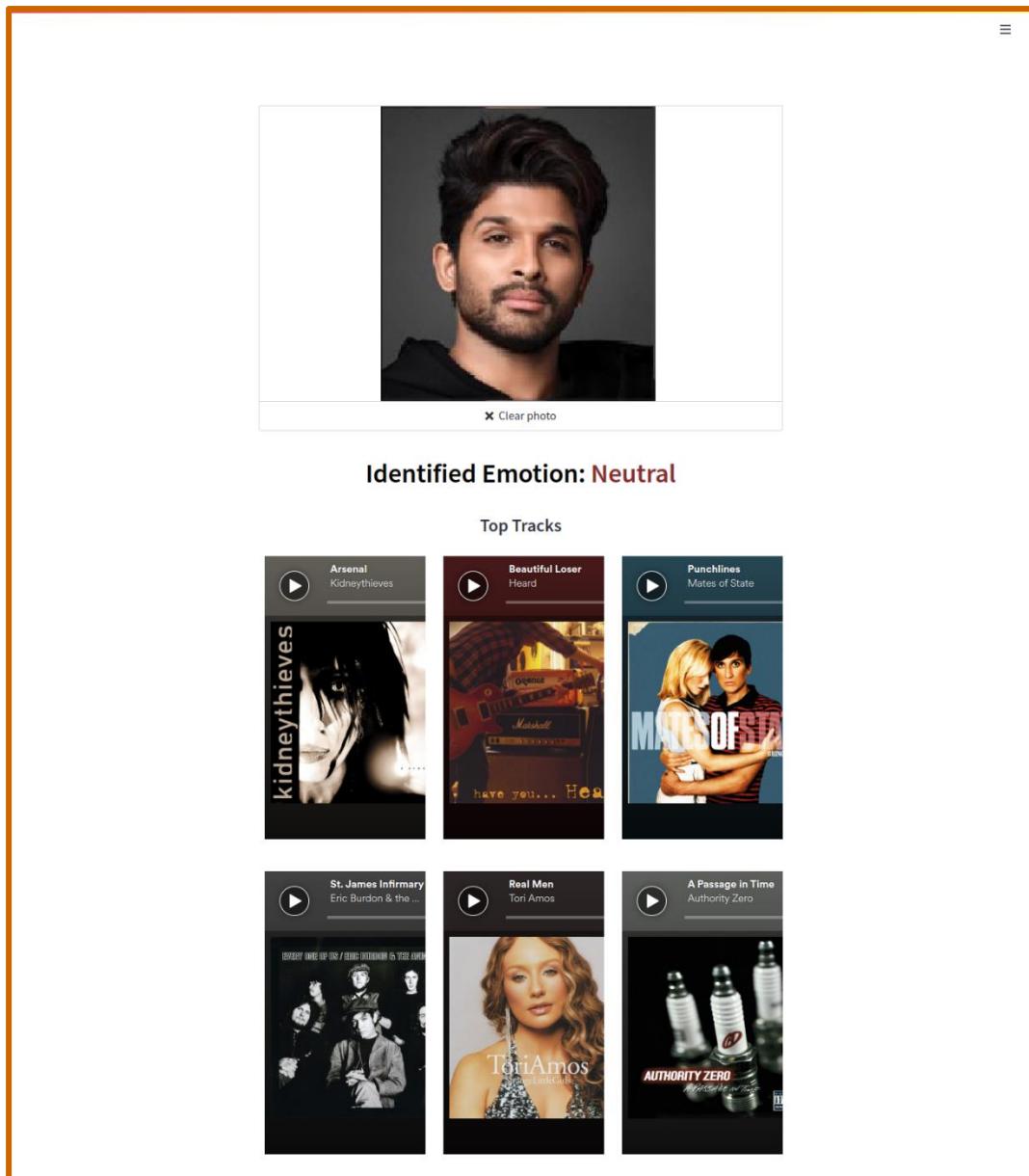


Figure 5.1.5: Application correctly Identifying Neutral Emotion.

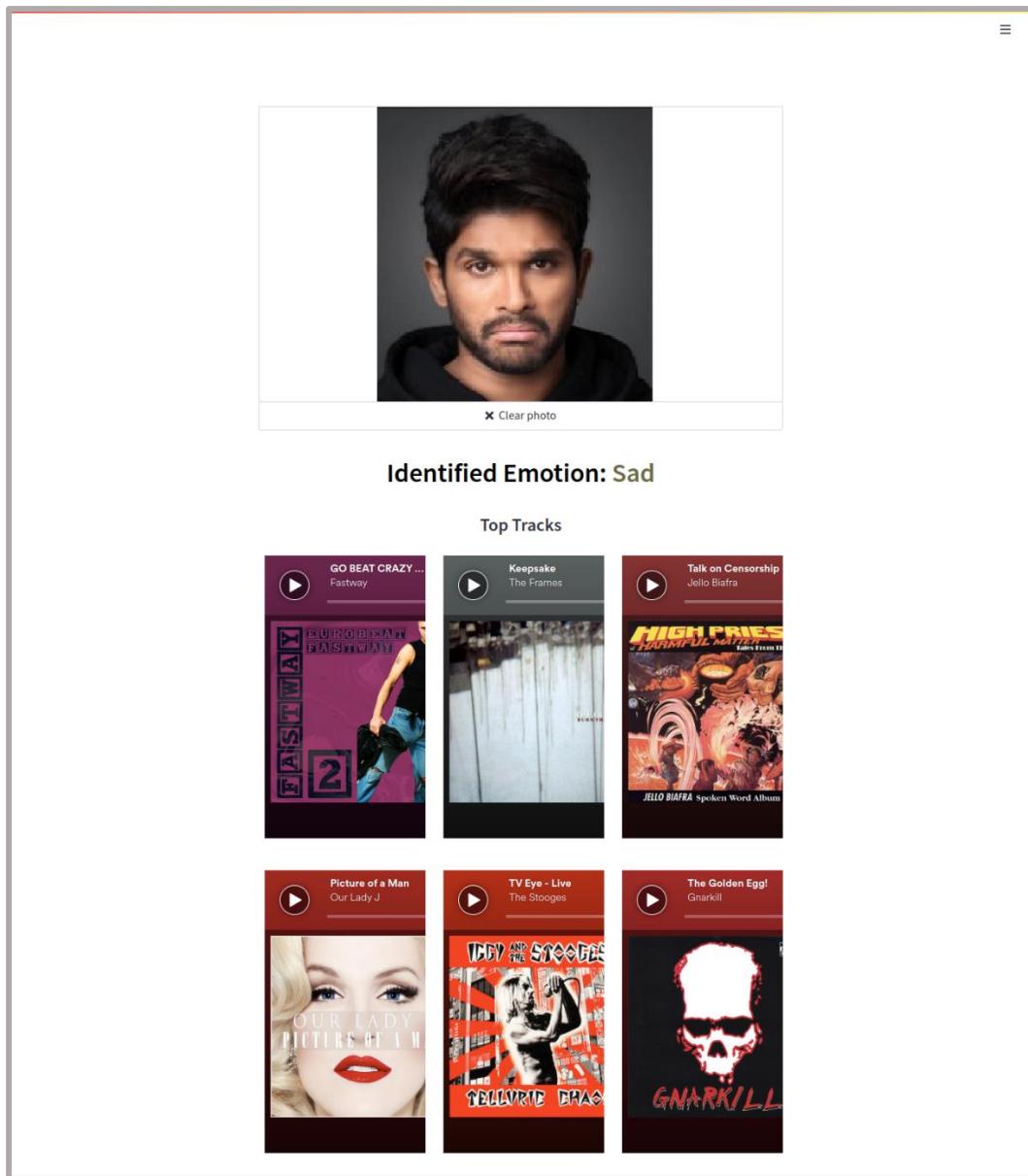


Figure 5.1.6: Application correctly Identifying Sad Emotion.

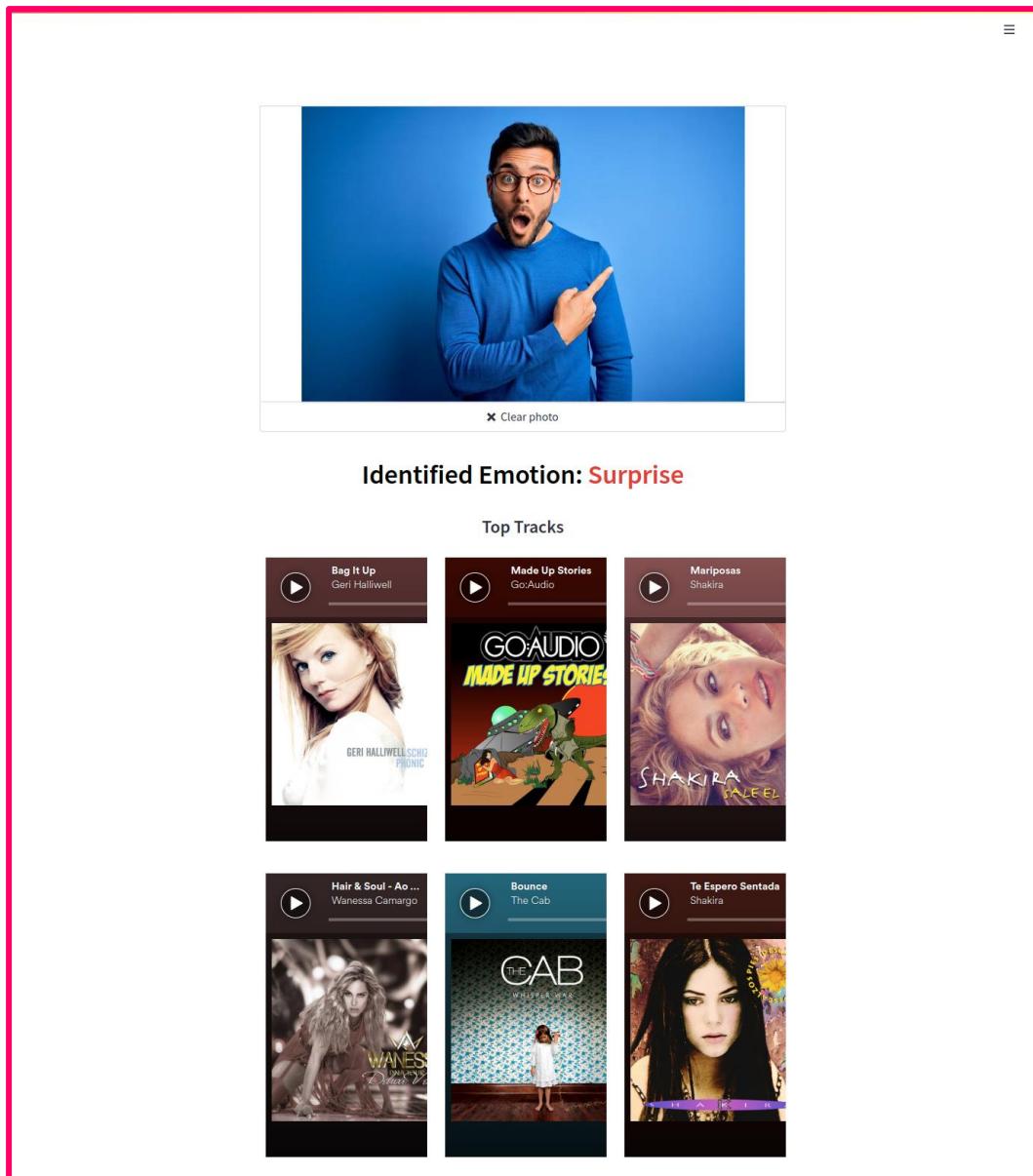


Figure 5.1.7: Application correctly Identifying Surprise Emotion.

## 5.2. Inaccurate Predictions

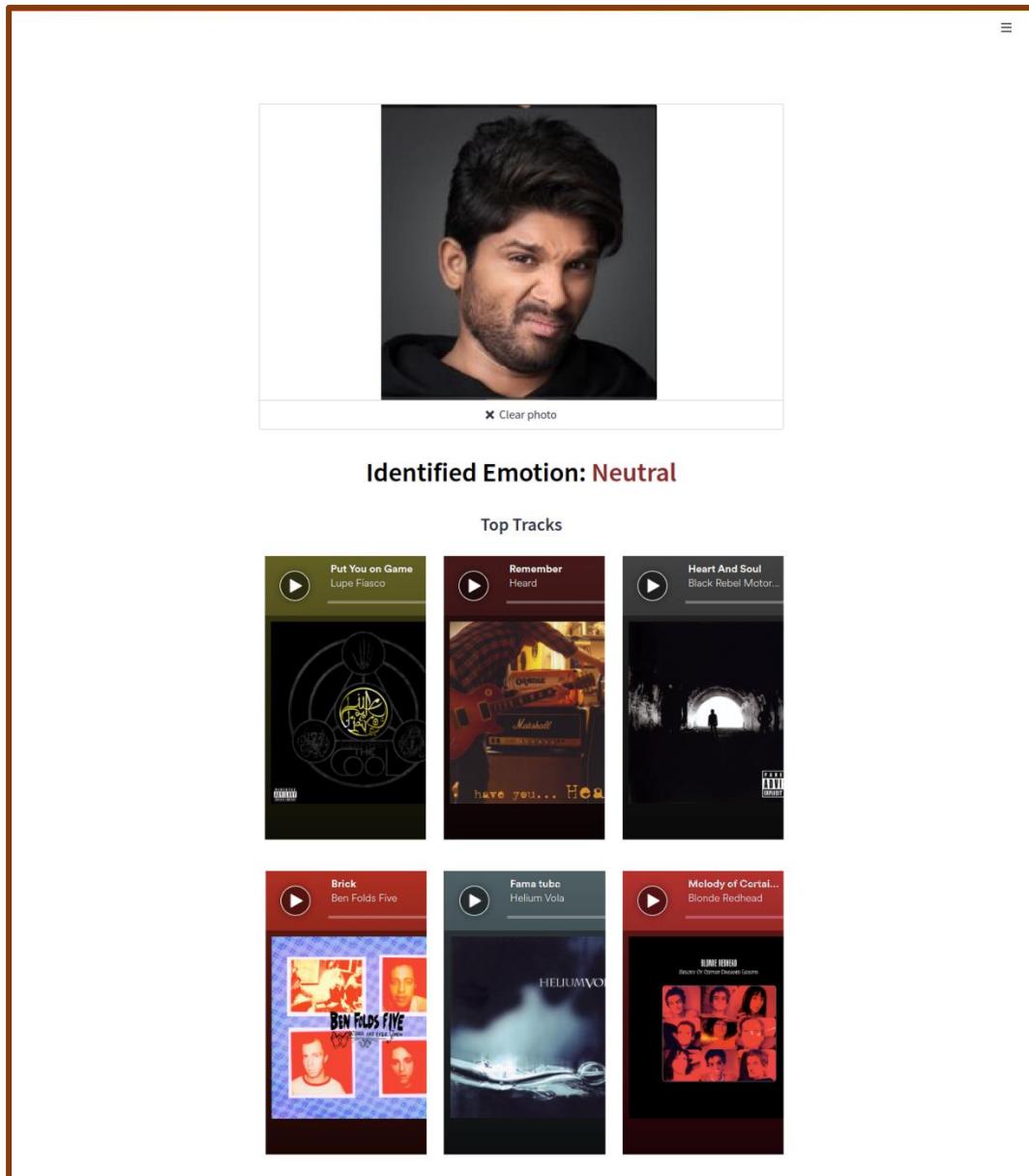


Figure 5.2.1: Application wrongly classifying Disgust as Neutral.

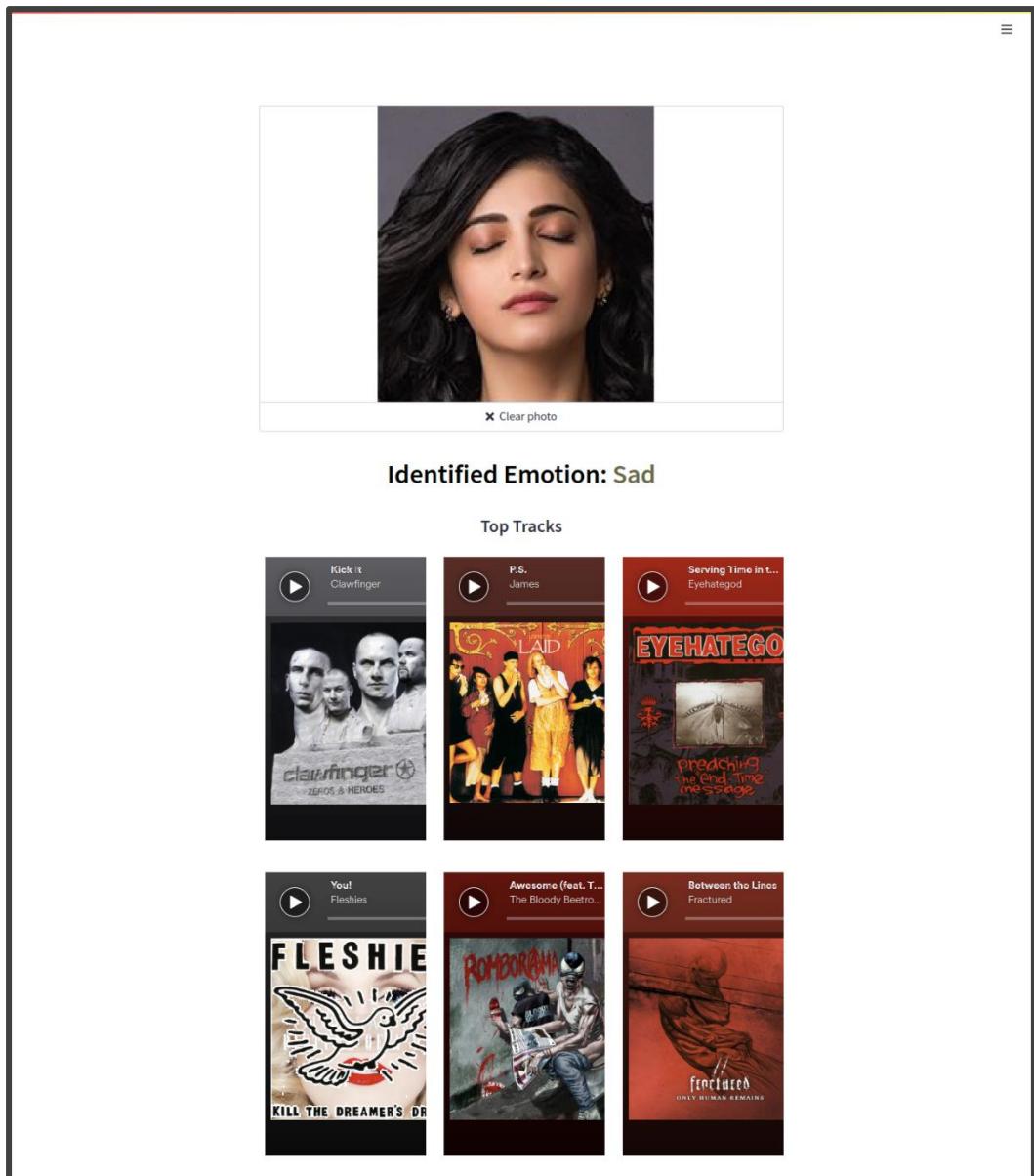


Figure 5.2.2: Application wrongly classifying Neutral as Sad.

### 5.3. No-Face-Detected

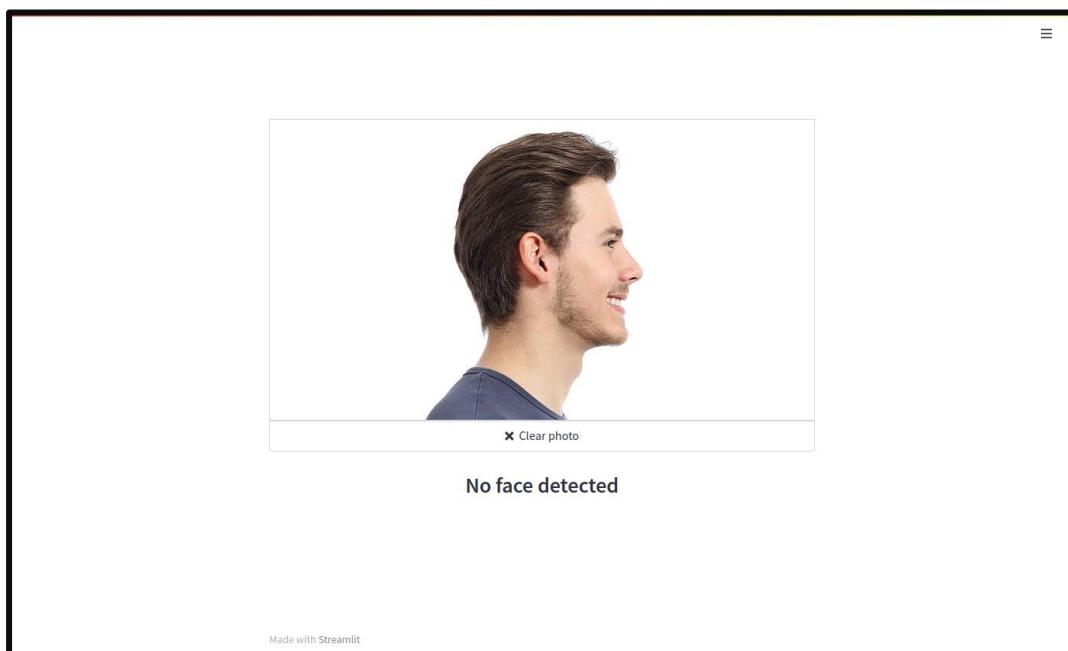


Figure 5.3.1: Application failing to detect Face.

## 6. Testing

### 6.1. Accuracy of CNN Model

```
● ● ●  
1 Testing: Neutral  
2 Accuracy for Neutral : 0.326  
3 Label Present: 500  
4 Label Correct: 163  
5  
6 Testing: Sad  
7 Accuracy for Sad : 0.596  
8 Label Present: 500  
9 Label Correct: 298  
10  
11 Testing: Happy  
12 Accuracy for Happy : 0.920  
13 Label Present: 500  
14 Label Correct: 460  
15  
16 Testing: Anger  
17 Accuracy for Anger : 0.350  
18 Label Present: 500  
19 Label Correct: 175  
20  
21 Testing: Disgust  
22 Accuracy for Disgust : 0.000  
23 Label Present: 500  
24 Label Correct: 0  
25  
26 Testing: Surprise  
27 Accuracy for Surprise : 0.116  
28 Label Present: 500  
29 Label Correct: 58  
30  
31 Testing: Fear  
32 Accuracy for Fear : 0.088  
33 Label Present: 500  
34 Label Correct: 44  
35  
36 Total Accuracy: 0.342  
37 Total Present: 3500  
38 Total Correct: 1198  
39
```

Figure 6.1.1: Accuracy of CNN Model against AffectNet Test.

## 6.2. Accuracy for Deep Face Model

### 6.2.1. Dataset: AffectNet Test

```
● ○ ●  
1 Currently Testing Neutral  
2  
3 Processed 1000 images so far  
4 Current Accuracy: 0.348  
5  
6 Processed 2000 images so far  
7 Current Accuracy: 0.355  
8  
9 Processed 3000 images so far  
10 Current Accuracy: 0.362  
11  
12 Processed 4000 images so far  
13 Current Accuracy: 0.354  
14  
15 Processed 5000 images so far  
16 Current Accuracy: 0.351  
17  
18 Processed 6000 images so far  
19 Current Accuracy: 0.349  
20  
21 Processed 7000 images so far  
22 Current Accuracy: 0.345  
23  
24 Processed 8000 images so far  
25 Current Accuracy: 0.345  
26  
27 Processed 9000 images so far  
28 Current Accuracy: 0.344  
29  
30 Processed 10000 images so far  
31 Current Accuracy: 0.344  
32  
33 Processed 11000 images so far  
34 Current Accuracy: 0.344  
35  
36 Processed 12000 images so far  
37 Current Accuracy: 0.344  
38  
39 Processed 13000 images so far  
40 Current Accuracy: 0.345  
41  
42 Accuracy for Neutral: 0.345  
43 Details: 4527/13093  
44  
45 Currently Testing Sad  
46  
47 Processed 14000 images so far  
48 Current Accuracy: 0.338  
49  
50 Processed 15000 images so far  
51 Current Accuracy: 0.330  
52  
53 Processed 16000 images so far  
54 Current Accuracy: 0.323  
55  
56 Processed 17000 images so far  
57 Current Accuracy: 0.317  
58  
59 Accuracy for Sad: 0.223  
60 Details: 1014/4529
```

Figure 6.2.1: Accuracy of Deep Face Model against AffectNet Test.

### 6.2.2. Dataset: AffectNet Train

```
61
62 Currently Testing Happy
63
64 Processed 18000 images so far
65 Current Accuracy: 0.318
66
67 Processed 19000 images so far
68 Current Accuracy: 0.330
69
70 Processed 20000 images so far
71 Current Accuracy: 0.338
72
73 Processed 21000 images so far
74 Current Accuracy: 0.347
75
76 Processed 22000 images so far
77 Current Accuracy: 0.355
78
79 Processed 23000 images so far
80 Current Accuracy: 0.363
81
82 Processed 24000 images so far
83 Current Accuracy: 0.369
84
85 Processed 25000 images so far
86 Current Accuracy: 0.376
87
88 Processed 26000 images so far
89 Current Accuracy: 0.381
90
91 Processed 27000 images so far
92 Current Accuracy: 0.386
93
94 Processed 28000 images so far
95 Current Accuracy: 0.390
96
97 Processed 29000 images so far
98 Current Accuracy: 0.395
99
100 Processed 30000 images so far
101 Current Accuracy: 0.400
102
103 Processed 31000 images so far
104 Current Accuracy: 0.404
105
106 Processed 32000 images so far
107 Current Accuracy: 0.407
108
109 Processed 33000 images so far
110 Current Accuracy: 0.411
111
112 Processed 34000 images so far
113 Current Accuracy: 0.414
114
115 Processed 35000 images so far
116 Current Accuracy: 0.417
117
118 Processed 36000 images so far
119 Current Accuracy: 0.421
120
121 Processed 37000 images so far
122 Current Accuracy: 0.424
```

Figure 6.2.2: Accuracy of Deep Face Model against AffectNet Train (1).

```
123
124 Processed 38000 images so far
125 Current Accuracy: 0.427
126
127 Processed 39000 images so far
128 Current Accuracy: 0.430
129
130 Processed 40000 images so far
131 Current Accuracy: 0.433
132
133 Accuracy for Happy: 0.525
134 Details: 12265/23325
135
136 Currently Testing Anger
137
138 Processed 41000 images so far
139 Current Accuracy: 0.434
140
141 Processed 42000 images so far
142 Current Accuracy: 0.428
143
144 Processed 43000 images so far
145 Current Accuracy: 0.422
146
147 Processed 44000 images so far
148 Current Accuracy: 0.416
149
150 Processed 45000 images so far
151 Current Accuracy: 0.411
152
153 Accuracy for Anger: 0.172
154 Details: 745/4314
155
156 Currently Testing Disgust
157
158 Accuracy for Disgust: 0.054
159 Details: 36/657
160
161 Currently Testing Surprise
162
163 Processed 46000 images so far
164 Current Accuracy: 0.404
165
166 Processed 47000 images so far
167 Current Accuracy: 0.397
168
169 Processed 48000 images so far
170 Current Accuracy: 0.391
171
172 Accuracy for Surprise: 0.091
173 Details: 216/2357
174
175 Currently Testing Fear
176
177 Processed 49000 images so far
178 Current Accuracy: 0.386
179
180 Accuracy for Fear: 0.204
181 Details: 225/1101
182
183 Total Accuracy: 0.385
184 Total Correct: 19028
185 Total Present: 49376
186
187
```

Figure 6.2.3: Accuracy of Deep Face Model against AffectNet Train (2).

```
123
124 Processed 38000 images so far
125 Current Accuracy: 0.427
126
127 Processed 39000 images so far
128 Current Accuracy: 0.430
129
130 Processed 40000 images so far
131 Current Accuracy: 0.433
132
133 Accuracy for Happy: 0.525
134 Details: 12265/23325
135
136 Currently Testing Anger
137
138 Processed 41000 images so far
139 Current Accuracy: 0.434
140
141 Processed 42000 images so far
142 Current Accuracy: 0.428
143
144 Processed 43000 images so far
145 Current Accuracy: 0.422
146
147 Processed 44000 images so far
148 Current Accuracy: 0.416
149
150 Processed 45000 images so far
151 Current Accuracy: 0.411
152
153 Accuracy for Anger: 0.172
154 Details: 745/4314
155
156 Currently Testing Disgust
157
158 Accuracy for Disgust: 0.054
159 Details: 36/657
160
161 Currently Testing Surprise
162
163 Processed 46000 images so far
164 Current Accuracy: 0.404
165
166 Processed 47000 images so far
167 Current Accuracy: 0.397
168
169 Processed 48000 images so far
170 Current Accuracy: 0.391
171
172 Accuracy for Surprise: 0.091
173 Details: 216/2357
174
175 Currently Testing Fear
176
177 Processed 49000 images so far
178 Current Accuracy: 0.386
179
180 Accuracy for Fear: 0.204
181 Details: 225/1101
182
183 Total Accuracy: 0.385
184 Total Correct: 19028
185 Total Present: 49376
186
187
```

Figure 6.2.4: Accuracy of Deep Face Model against AffectNet Train (3).

### 6.3. Accuracy of MTCNN based FER Model

```
● ○ ●

1 Testing: Neutral
2 Accuracy for Neutral : 0.574
3 Label Present: 500
4 Label Correct: 287
5
6 Testing: Sad
7 Accuracy for Sad : 0.374
8 Label Present: 500
9 Label Correct: 187
10
11 Testing: Happy
12 Accuracy for Happy : 0.690
13 Label Present: 500
14 Label Correct: 345
15
16 Testing: Anger
17 Accuracy for Anger : 0.252
18 Label Present: 500
19 Label Correct: 126
20
21 Testing: Disgust
22 Accuracy for Disgust : 0.102
23 Label Present: 500
24 Label Correct: 51
25
26 Testing: Surprise
27 Accuracy for Surprise : 0.184
28 Label Present: 500
29 Label Correct: 92
30
31 Testing: Fear
32 Accuracy for Fear : 0.336
33 Label Present: 500
34 Label Correct: 168
35
36 Total Accuracy: 0.358
37 Total Present: 3500
38 Total Correct: 1256
```

Figure 6.3.1: Accuracy of MTCNN based FER model.

## 6.4. Accuracy for K-Means Clustering

```
● ● ●  
1 Parsing Dataset...  
2 Performing K-Means Clustering without initial centroids...  
3 Silhouette score: 0.309  
4  
5 Performing K-Means Clustering with initial centroids...  
6 Silhouette score: 0.318  
7  
8 Splitting Dataframe into Train(80%) and Test(20%) Set...  
9 Performing K-Means Clustering on Train and Test Dataset...  
10 Total Rows in Test Set: 12326  
11 Total Correctly Classified: 9583  
12 Accuracy: 77.746  
13
```

Figure 6.4.1: Accuracy for K-MEANS Clustering.

## 6.5. Compare & Contrast Emotion detection models

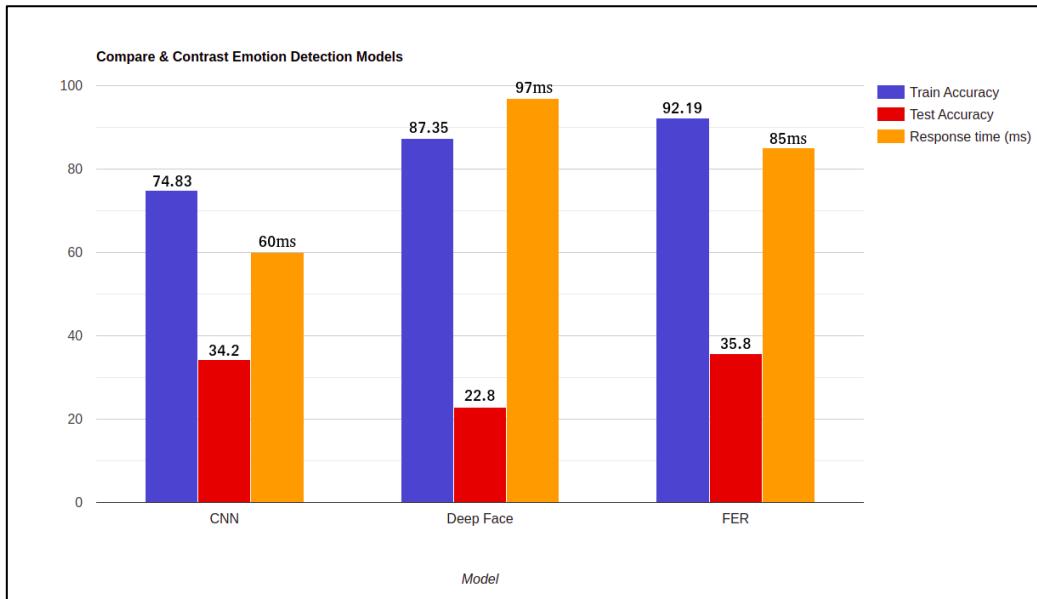


Figure 6.5.1: Compare & Contrast of Emotion Detection Models.

## 6.6. Emotion-wise Accuracies of all Models

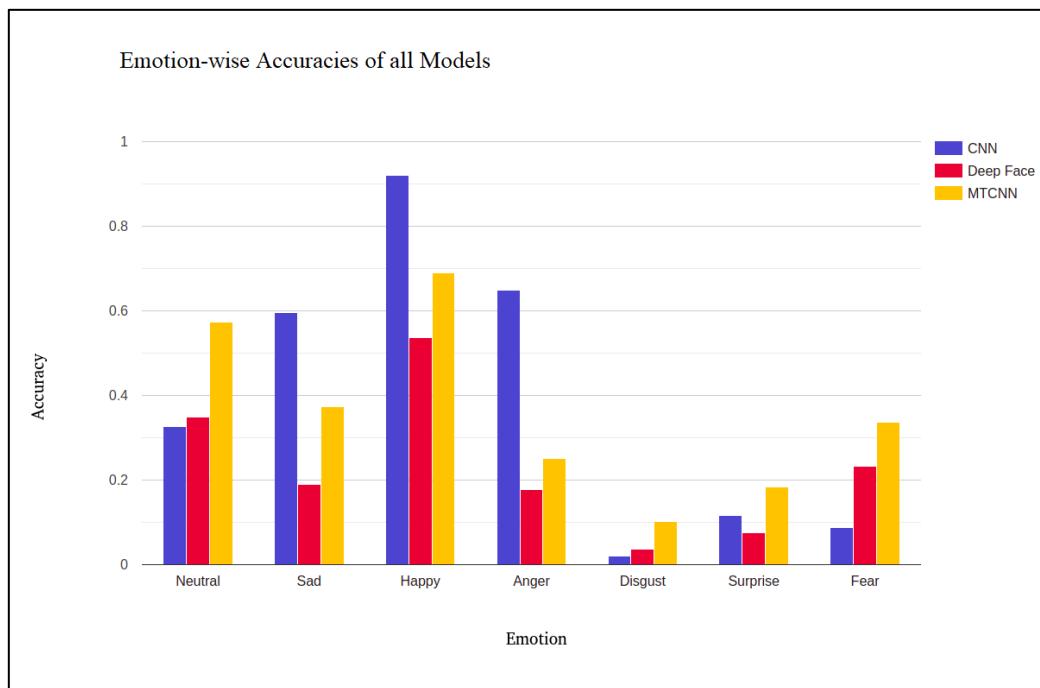


Figure 6.6.1: Emotion-wise Accuracies of all Models.

## 7. Conclusion and Future Work

General music recommendation systems are quite ineffective because they don't consider contextual information. The aim of this work was to build a comprehensive context-aware music recommendation system, which recommends music based on the user's emotion. The overarching goal of this novel system is to blend user emotion with music recommendation, to improve accuracy and enhance the listening experience.

After extensive research and experimentation, we have proposed an integrated system that uses the MTCNN model for emotion detection and content-based filtering for music recommendation. The results suggest that the performance of the system was on par with current state-of-the-art systems.

We acknowledge some shortcomings in our work that opens up possibilities for future work. We would like to explore the following for future work:

1. Incorporate additional contextual information such as mouse click patterns, keystrokes, and user heart rate.
2. Introduce a rating system where users can rate the recommended songs. This helps in improving the accuracy of further recommendations.
3. Improve the accuracy of the MTCNN model by training against extremely large datasets using GPUs.

## References

- [1]. Sunitha M, Adilakshmi T (2018) Music recommendation system with user-based and item-based collaborative filtering technique. In Networking Communication and Data Knowledge Engineering: Springer.
- [2]. H.-C. Kwon and M. Kim. Lyrics-based emotion classification using feature selection by partial syntactic analysis. 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence (ICTAI 2011), 2011.
- [3]. J. A. Speck, E. M. Schmidt, B. G. Morton, and Y. E. Kim. A comparative study of collaborative vs. traditional musical mood annotation. In A. Klapuri and C. Leider, editors, ISMIR, University of Miami, 2011.
- [4]. K. Bischoff, C. Firat, R. Paiu, W. Nejdl, C. Laurier and M. Sordo. Music Mood and Theme Classification a Hybrid Approach. In Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009.
- [5]. Hyung, Z. Park, J.S. Lee, K. Utilizing context-relevant keywords extracted from a large collection of user-generated documents for music discovery. Inf. Process. Manag. 2017.
- [6]. M. V. Zaanen and P. Kanters. Automatic mood classification using tf\*idf based on lyrics. In Proceedings of the 11th International Society for Music Information Retrieval Conference, Utrecht, The Netherlands, August 9-13, 2010.
- [7]. R. Malheiros, R. Panda, P. Gomes, and R. Paiva. Bi-modal music emotion recognition: Novel lyrical features and dataset. In 9th International Workshop on Music and Machine Learning MML2016 in conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML/PKDD 2016.
- [8]. Ji-Oh Yoo Han-Saem Park and Sung-Bae Cho. L. Wang. A context-aware MRS using fuzzy Bayesian networks with utility theory. In FSKD 2006.
- [9]. Giz H. He, J. Jin, Y. Xiong, B. Chen, W. Sun, and L. Zhao. Language Feature Mining for Music Emotion Classification via Supervised Learning from Lyrics. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

- [10]. Y. Hu, X. Chen, and D. Yang. Lyric-based Song Emotion Detection with Affective Lexicon and Fuzzy Clustering Method. In ISMIR, 2009.
- [11]. C. Laurier, J. Grivolla, and P. Herrera. “Multimodal Music Mood Classification Using Audio and Lyrics. 2008 Seventh International Conference on Machine Learning and Applications, 2008.
- [12]. F. H. Rachman, R. Sarno, and C. Faticahah, “Music Emotion Classification based on LyricsAudio using Corpus based Emotion,” Int. Journal of Electrical and Computer Engineering (IJECE), 2018.
- [13]. R. Malheiro, R. Panda, P. Gomes, and R. Paiva. Music emotion recognition from lyrics: A comparative study. In 6th International Workshop on Machine Learning and Music, 2013.
- [14]. Shlok Gilda, Husain Zafar, Chintan Soni, and Kshitija Waghurdekar. Smart music player integrating facial emotion recognition and music mood recommendation. In 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). IEEE, 2017.
- [15]. I. Bakker, T. van der Voordt, P. Vink, and J. de Boon. Pleasure, arousal, dominance: Mahrabian and russell revisited. Current Psychology, 2014.
- [16]. Emotion based Music Recommendation System (<https://github.com/Udhay-Brahmi/Emotion-based-music-recommendation-system>).
- [17]. Retinaface (<https://github.com/serengil/retinaface>).
- [18]. FER (<https://github.com/justinshenk/fer>).
- [19]. Realtime Emotion Detection (<https://github.com/neha01/Realtime-Emotion-Detection>).
- [20]. Emotion Detection (<https://github.com/dhruvpandey662/Emotion-detection>).
- [21]. Keras Optimizers (<https://keras.io/api/optimizers/>).
- [22]. Keras Metrics (<https://keras.io/api/metrics/>).
- [23]. Keras Losses (<https://keras.io/api/losses/>).
- [24]. Future Computers will be Radically Different (<https://www.youtube.com/watch?v=GVsUOuSjvcg>).