

# Emotion Sensitive Music Broadcasting by Analysing Facial Expressions using Machine Learning

MSc Research Project  
Master of Science in Data Analytics

Christy Lyona Joseph Vijayan  
Student ID: x21202583

School of Computing  
National College of Ireland

Supervisor: Taimur Hafeez

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Christy Lyona Joseph Vijayan
<b>Student ID:</b>	x21202583
<b>Programme:</b>	Master of Science in Data Analytics
<b>Year:</b>	2023
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Taimur Hafeez
<b>Submission Due Date:</b>	14/08/2023
<b>Project Title:</b>	Emotion Sensitive Music Broadcasting by Analysing Facial Expressions using Machine Learning
<b>Word Count:</b>	XXX
<b>Page Count:</b>	16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	
<b>Date:</b>	12th August 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Emotion Sensitive Music Broadcasting by Analysing Facial Expressions using Machine Learning

Christy Lyona Joseph Vijayan  
x21202583

## 1 Introduction

This config manual contains all of the information needed to obtain the output. This manual includes the layout of the requirements for replicating my thesis. This document acts as a connection bridging the final output and the thesis. Introduces the dataset and pre-processing of the dataset to be used, in my case the FER 2013 dataset provided by google. This manual also gives a brief introduction to the roadmap of this project, the methodology used and the end goal of my Auto Playlist Generator.

## 2 Requirements and Design of the CNN Model

This section introduces the dataset and pre-processing of the dataset to be used, in my case the FER 2013 dataset provided by google. This Chapter also gives a brief introduction to the roadmap of this project, the methodology used and the end goal of my Auto Playlist Generator.

This section is sub divided in 3 major parts:

1. Data Preparation.
2. Classification.
3. Key Aspects.

### 2.1 Data Preparation.

Here the steps of selecting the dataset, preparing the dataset for our particular observation and observing how the dataset is panned out and its various features will be addressed. This section is just an introduction to the Deep Learning CNN network creation.

#### 2.1.1 Data Gathering: FER 2013.

The Dataset selected for the CNN model is the famous FER 2013 dataset. It is an open source licensed dataset freely available on Kaggle. This dataset has images of 48x48 pixels grayscale each. Each Directory has 7 other Directories of 7 different emotions namely Angry, Disgust, Fear, Happy, Neutral, Sad and Surprised.

### 2.1.2 Data Pre-Processing.

We use the Keras Pre-Processing library to accomplish this, we read the images from the training dataset with respect to their emotion. Using the `load_img` command provided by Keras Preprocessing we load these images and display them using another common package namely Math Plot. (Fig. 1).

```
expression = ['angry','fear','sad','surprise','happy','disgust','neutral']

for e in expression:
    plt.figure(figsize= (4,4))
    for i in range(1, 10, 1):
        plt.subplot(3,3,i)

        img = load_img(folder_path+"train/"+e+"/"+
            os.listdir(folder_path + "train/" + e)[i], target_size=(picture_size, picture_size))

        plt.imshow(img)
    plt.show()
```

Figure 1: Using Keras Load Image to read the Dataset.

On Execution on the above code (Fig. 1) 7 different plots each of size 4x4 pixels and consisting of 9 first images present of each individual directories are displayed and can be observed below in Fig. 2.

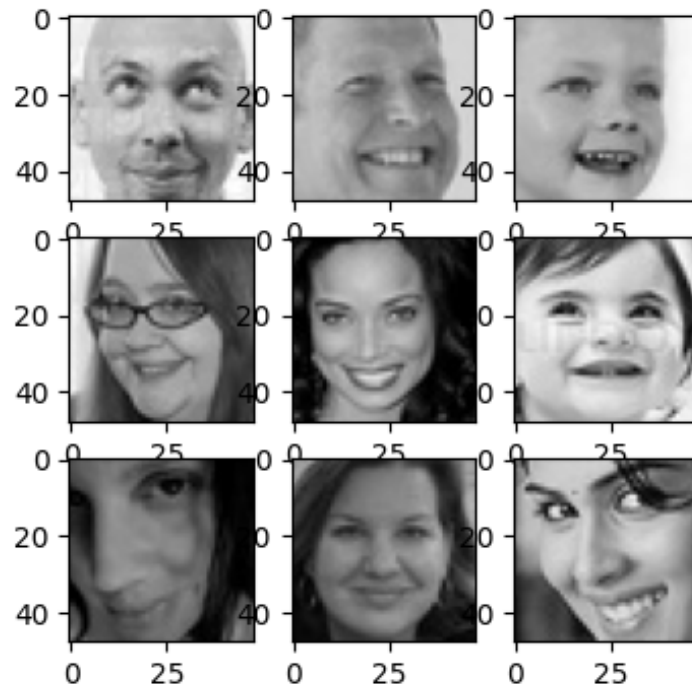


Figure 2: Happy

For this project as we plan on using VGG16 modified we now convert all images to an input size of 48 pixels as the input of VGG16 is 48.

## 2.2 Classification.

We select a batch size of 16 and take the test and train set as a set of grayscale images of size 48 through ImageDataGenerator function provided by Keras again. This can be seen in Fig. 3

```
batch_size=16
datagen_train=ImageDataGenerator()
datagen_val=ImageDataGenerator()

train_set= datagen_train.flow_from_directory(folder_path+"train",
                                             target_size=(picture_size,picture_size),
                                             color_mode="grayscale",
                                             batch_size=batch_size,
                                             class_mode="categorical",
                                             shuffle=True)
test_set=datagen_val.flow_from_directory(folder_path+"validation",
                                         target_size=(picture_size,picture_size),
                                         color_mode="grayscale",
                                         batch_size=batch_size,
                                         class_mode="categorical",
                                         shuffle=False)
```

Figure 3: Creating Train And Test sets.

The image in Fig. 5 is the layers inserted by running the code block in Fig. 4. The last layer has a shape of 7 that refers to our emotion classes of 7 different emotions ie. Angry, Disgust, Happy, Fear, Sad, Surprised and Neutral.

```
#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes,activation="softmax"))

opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Figure 4: Model Creation.

Now that the final layer has the right output classes we go ahead and fit this model to the training dataset. Thanks to Keras again we have libraries available to monitor the Episodes(aka. Epocs) and callbacks to attain and early stop if a sufficient accuracy or validation accuracy is reached.

## 2.3 Key Aspects.

Some Key Aspects of this Modelling plan can be mentioned as:

Training takes around 5 hours for just 10 epocs, the maximum of 13 epocs were reached which took 7 hours and 17 minutes. To avoid retraining of the entire model, the weights are stored in an h5 file that can be used to predict the output without training the entire model again as shown in Fig. 6

An Accuracy after 13 epocs was concluded at 72% and early stop was triggered as the Model was entering Over-fitting due to an increase in the Validation Loss.

```

dense_1 (Dense)          (None, 512)          131584
batch_normalization_5 (Batch Normalization) (None, 512)          2048
activation_5 (Activation) (None, 512)          0
dropout_5 (Dropout)      (None, 512)          0
dense_2 (Dense)          (None, 7)             3591
=====
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968

```

Figure 5: Summary of the Layers inserted.

```

model.save_weights('face_emotion_model.h5')

```

Figure 6: Saving the weights of the best trained Epochs.

```

Epoch 13/48
1801/1801 [=====] - ETA: 0s - loss: 0.7402 - accuracy: 0.7282Restoring model weights from the end of the best epoch: 10.
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
1801/1801 [=====] - 332s 184ms/step - loss: 0.7402 - accuracy: 0.7282 - val_loss: 1.2179 - val_accuracy: 0.5755 - lr: 0.0010
Epoch 13: early stopping

```

Figure 7: Accuracy of Face Recognition Model.

The Accuracy and Loss are plotted using Mat-Plot library to visualize the loss and accuracy of the model and gain a fair understanding of how accurately the model will perform as shown in Fig. 7, 8

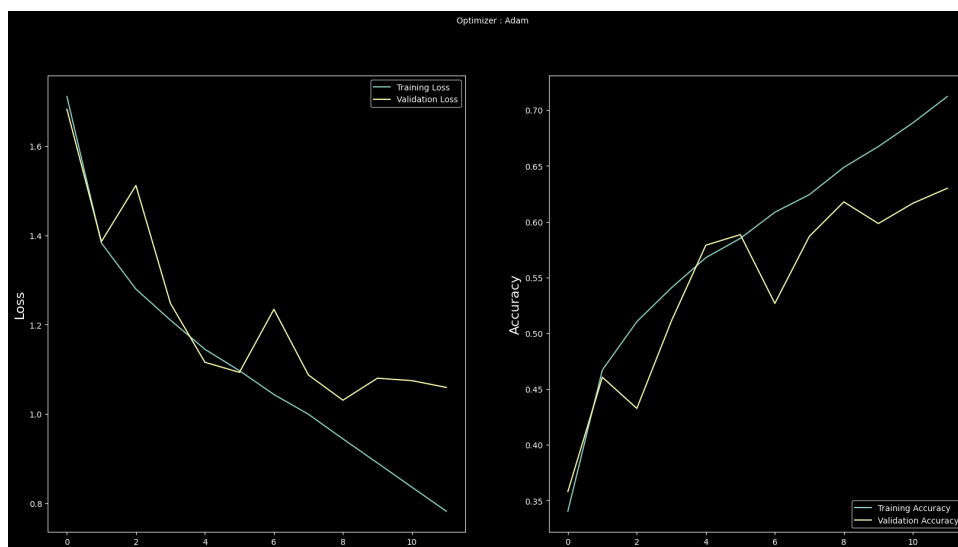


Figure 8: Comparison of Training and Validation Loss and Accuracy.

Detection of all emotions are done in real time and all the 7 emotions are correctly and accurately detected and the output is generated.

All these outputs are recorded per second and a counter function is set up to record the emotions and return the Two most prominent emotions. This can be seen in 9. The numbers represent the number of seconds the user displayed his/her emotion. For testing and evaluation purpose, i have tried to display all 7 emotions and 9 shows how long i held that emotion.

```
{'Fear': 7, 'Neutral': 410, 'Happy': 39, 'Sad': 137, 'Disgust': 30, 'Angry': 10, 'Surprised': 17}
Neutral, Sad
```

Figure 9: Emotions in an Array with Prominent two Emotions Detected.

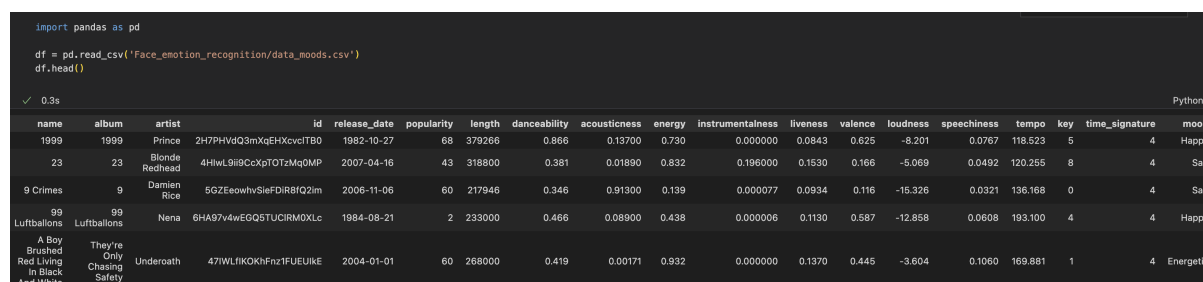
The end of the CNN and this section occurs when the two Prominent emotions are retrieved and this data is ready to be now fed as input to the RNN model where using a dataset we shall predict the songs that would help improve the users experience.

## 3 Requirements and Design of the RNN Model

### 3.1 Understanding the Design.

The Music Mood Dataset is a open dataset available at Kaggle provided by Spotify. The columns present in this dataset are a total of 19 in number however not all are useful for our purpose. The entire dataset is shown in Fig. 10. The reason for dropping few columns and retaining the rest will also be discussed in Data Preparation.

```
import pandas as pd
df = pd.read_csv('Face_emotion_recognition/data_moods.csv')
df.head()
```



name	album	artist	id	release_date	popularity	length	danceability	acousticness	energy	instrumentalness	liveness	valence	loudness	speechiness	tempo	key	time_signature	mood
1999	1999	Prince	2H7PHVdQ3mXgEHXcvcITB0	1982-10-27	68	379266	0.866	0.13700	0.730	0.000000	0.0843	0.625	-8.201	0.0767	118.523	5	4	Happy
23	23	Blondie Redhead	4HlwL9i9CxCxptOTzmqOMP	2007-04-16	43	318800	0.381	0.01890	0.832	0.196000	0.1530	0.166	-5.069	0.0492	120.255	8	4	Sad
9 Crimes	9	Damien Rice	5GZEeowhvsIeFDiR8fQ2im	2006-11-06	60	217946	0.346	0.91300	0.139	0.000077	0.0934	0.116	-15.326	0.0321	136.168	0	4	Sad
99 Luftballons	99 Luftballons	Nena	6HA97v4wEGQ5TUCiRM0Xlc	1984-08-21	2	233000	0.466	0.08900	0.438	0.000006	0.1130	0.587	-12.858	0.0608	193.100	4	4	Happy
A Boy Brushed Red Living In Black And White	They're Only Chasing Safety	Underoath	47IWLfIKOKHFnz1FUeUIKE	2004-01-01	60	268000	0.419	0.00171	0.932	0.000000	0.1370	0.445	-3.604	0.1060	169.881	1	4	Energetic

Figure 10: Raw Dataset.

There are two datasets and two different neural networks at play here. First a simple CNN that takes the live feed of the individual and gets the top two emotions of that individual. Second the Simple RNN that predicts the Mood of a song. Using both the Networks outputs we call a Recommend\_Song() function that with the use of the emotions of the individual and mood of the song provides the top 5 Popular songs to uplift or improve the individuals mood.

### 3.2 Introduction to SoundCloud.

Before we enter the technicality of creation of the RNN Network, let us understand what is SoundCloud and why I have chosen Sound Cloud as our Music player and not created a Default Music Player.

#### 3.2.1 Creating the SoundCloud Music Player.

The SoundCloud web-application can easily be called by using the web-driver provided by Selenium and taking control of Google Chrome. It is an automated driver that allows us to pass API requests and get data and post data onto any website of our choosing.

First we need to set up Chrome. For this the location of the chrome driver is needed and a get request is called to get the website we need to visit, for our purpose its the home page of SoundCloud. The track URL in fig.11 is the query we will append the name of the song we wish to play and pass once SoundCloud is up and running.



```

# Create a Selenium WebDriver instance
# Setting Up Chrome
browser = webdriver.Chrome()
# Open the webpage with the cookie pop-up
browser.get("https://soundcloud.com")

```

Figure 11: Setup for Google Chrome.

The function `Play_Songs` takes in two parameters `songs` and `click`, `Click` is the counter that facilitates the On Click Mouse operation in the code. The `songs` is a list of songs predicted, this is the inputs to the query to call in the songs from SoundCloud.

The `Play_Songs` function also provide the user with a continuous loop in which the user can easily switch from the current song to the next song by pressing the key 1. This is just an resemblance key for testing purpose and can be replaced by the next button on the device. Fig. 12 also sets the `Click` to 1 as after the first run there are no cookie and permission requests to accept hence allowing us to only change between tabs and play music using the mouse click which is also automated.

With this we come to our final function namely the `get_songs` function that gets the name of the songs, passes the argument and switches between tabs using the keyboard shortcuts and clicks the play button on the webpage to play the song track. This can be visualised in Fig 13

### 3.3 Creating the Music Recommender using RNN.

Our Target Variable to classify will be the mood of each song, this will be carried out by considering the inputs or the features of each song. Weights will be assigned depending on the values of each feature and the prediction will be made.

To understand the modelling, prediction and working of the entire RNN network let us begin from the dataset and getting a deeper understanding of it.

#### 3.3.1 Data Preparation.

With a total of 672 different artists, for the entire year of 2020 the Music Mood dataset also gives a numerical representation of the individual songs Popularity, length, danceability, loudness, energy, acoustics, instrumental, liveliness, valence, speech, temp and key. These are all used to predict the next column entry that being the "mood" of the song. To understand the selection process of each column a brief understanding of each column is needed and that is listed below.

This can be done by referring to the code snippet in Fig 14

- Target Variable.

1. Mood: This is our target variable column that is mainly categorical and are in 4 categories namely: 'Happy', 'Sad', 'Energetic' and 'Calm'. Our goal is to predict the mood depending purely on the attributes of the song. These can be seen in Fig( 15)

```

def Play_Songs(songs,click):
    # Start Chrome
    print(">>> Welcome to the Emotion Based Music Player")

    track_url, chrome_driver_binary, browser = StartChrome()

    # Top Songs to Play
    counter = 0
    get_song(songs[counter], browser,track_url,click)
    click = 1

    while True:
        print(">>>1 - Play Next Song")
        print(">>>0 - EXIT!!!")

        choice = int(input(">>> Your Choice: "))

        if choice == 0:
            browser.quit()
            break

        if choice == 1:
            get_song(songs[counter+1],browser,track_url,click)
            counter = counter +1
            continue

```

Figure 12: Play\_Songs Function

```

def get_song(song_name, browser, track_url, first_click):
    name = song_name
    print()
    "%20".join(name.split(" "))
    browser.get(track_url + name)

    url = track_url + name
    request = requests.get(url)
    soup = bs4.BeautifulSoup(request.text, 'html.parser')
    tracks = soup.select("h2")[3:]
    track_links = []
    track_names = []
    for index, track in enumerate(tracks):
        track_links.append(track.a.get("href"))
        track_names.append(track.text)

    if first_click == 0:
        mouse = CMouse()
        mouse.position = (1053.60546875, 786.0703125)
        time.sleep(9)
        mouse.click(Button.left, 2)
        mouse.position = (553.3046875, 653.88671875)
        time.sleep(6)
        mouse.click(Button.left, 2)
    else:
        keyboard = CKey()
        keyboard.press(Key.cmd)
        keyboard.press(Key.tab)
        keyboard.release(Key.cmd)
        keyboard.release(Key.tab)
        mouse = CMouse()
        mouse.position = (553.3046875, 653.88671875)
        time.sleep(5)
        mouse.click(Button.left, 2)

    while True:
        break

```

Figure 13: Get\_Songs Function.

```

df = df.drop(['release_date', 'album', 'length'], axis = 1)
df.insert(0, 'id_', range(1, 1 + len(df)))
music_player = df[['name', 'artist', 'mood', 'popularity', 'id_']]

X = df.drop(['mood', 'artist', 'name', 'id_'], axis = 1)
y = df['mood']
X.head()

```

id_	popularity	danceability	acousticness	energy	instrumentalness	liveness	valence	loudness	speechiness	tempo	key	time_signature	
0	1	68	0.866	0.13700	0.730	0.000000	0.0843	0.625	-8.201	0.0767	118.523	5	4
1	2	43	0.381	0.01890	0.832	0.196000	0.1530	0.166	-5.069	0.0492	120.255	8	4
2	3	60	0.346	0.91300	0.139	0.000077	0.0934	0.116	-15.326	0.0321	136.168	0	4
3	4	2	0.466	0.08900	0.438	0.000006	0.1130	0.587	-12.858	0.0608	193.100	4	4
4	5	60	0.419	0.00171	0.932	0.000000	0.1370	0.445	-3.604	0.1060	169.881	1	4

Figure 14: Input Columns with ID variable.

With this detailed understanding of the columns taken into consideration and the columns dropped, we now move on to a much deeper understanding of how the rows are mapped. As we have not used the names of the song again due to one-hot encoding issue as our prediction is of a regression-classifier, we have no logical connection between the song attributes and names. Hence, to solve this i introduce an id\_ variable that creates a simple id count for each row. This can help us map the name of the predicted row to the attributes of the predicted row and the target variable being the mood of the song. The code snippet in Fig 14 gives you the clear understanding of how this is accomplished and Fig 15 shows you the Target variable to be predicted and its unique values or unique classes.

```

y.head()

```

0	Happy
1	Sad
2	Sad
3	Happy
4	Energetic

```

Name: mood, dtype: object

y.unique()

```

```

array(['Happy', 'Sad', 'Energetic', 'Calm'], dtype=object)

```

Figure 15: Output or Target Column.

### 3.3.2 Modelling.

I will be using multiple regression models over this dataset and judge their performance by the accuracy of each of the models. A few minute basic steps of Outlier detection and Removal of Duplicate or N/A values are skipped as this dataset has no such irregularities. We will be including 4 models here, namely

1. Gaussian Naive Bayes.

2. Logistic Regression.
3. SVC or as commonly known Support Vector Classifier.
4. Decision Tree Classifier.

```

gnb = GaussianNB()
LR = LogisticRegression()
SVC = svm.SVC()
DT = tree.DecisionTreeClassifier()

```

Figure 16: Models used.

Amongst these Gaussian Naive Bayes surprisingly out-performs each and every other model. For designing the model, i have created 2 empty lists for accuracy, one that holds all the accuracy values for the current model under test and feeds the average of the accuracy after 100 Monte-Carlo runs into the next list that holds the average of each and every other model after the 100 Monte-Carlo runs.(refer fig( 17))

```

acc=[]
acc1=[]
acc_SVC=[]
acc_DT = []
for i in [0.2,0.3,0.4,0.5,0.6,0.7,0.8]:
    for r in range(100):
        X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=i)

        gnb.fit(X_train, y_train)
        LR.fit(X_train, y_train)
        SVC.fit(X_train, y_train)
        DT.fit(X_train, y_train)

        y_pred = gnb.predict(X_test)
        y_pred1= LR.predict(X_test)
        y_predSVC = SVC.predict(X_test)
        y_predDT = DT.predict(X_test)

        actual=y_test
        prediction=y_pred
        prediction1=y_pred1
        predSVC = y_predSVC
        predDT = y_predDT

        accuracy=accuracy_score(actual,prediction)
        accuracy1=accuracy_score(actual,prediction1)
        accSVC =accuracy_score(actual,predSVC)
        accDT = accuracy_score(actual,predDT)
        acc.append(accuracy)
        acc1.append(accuracy1)
        acc_SVC.append(accSVC)
        acc_DT.append(accDT)
print("GNB={}, LR={}, SVC={}, DT={} for {} Split Dataframe".format(mean(acc), mean(acc1), mean(acc_SVC),mean(acc_DT),i))

```

Figure 17: Monte-Carlo Runs with Variable Data Split Size.

train\_test\_split is called: This splits the input and output dataframe under consideration randomly. As i have mentioned the split to be at 0.5, it means 50% of the dataframe will be considered for Training and the other 50% is considered for test purpose. However, this split will be completely random and different for all 100 Monet-Carlo runs. Hence, not just ensuring the best accuracy for a Dataframe split differently but also with different dimensions and picking up the best split with the best model. This is found to be Gaussian Naive Bayes for a split of 0.2 (refer fig( 18))

```

GNB=0.7998550724637679, LR=0.6328985507246377, SVC=0.30768115942028984, DT=0.7480434782608696 for 0.2 Split Dataframe
GNB=0.7973061770085831, LR=0.6328813141972702, SVC=0.30204446320529055, DT=0.7465945546644153 for 0.3 Split Dataframe
GNB=0.7952950270966312, LR=0.6318481488587863, SVC=0.29909630880352706, DT=0.7466994000793072 for 0.4 Split Dataframe
GNB=0.793118500643173, LR=0.6320639542096873, SVC=0.2960787913694091, DT=0.7445726550157485 for 0.5 Split Dataframe
GNB=0.7902423733300725, LR=0.6311899983192062, SVC=0.29440186804698365, DT=0.7436241434300745 for 0.6 Split Dataframe
GNB=0.7870973346704172, LR=0.6299798842542241, SVC=0.29250744721171024, DT=0.7410416025465452 for 0.7 Split Dataframe
GNB=0.7840823888640008, LR=0.6282861667274042, SVC=0.2906608199623662, DT=0.7376661563331565 for 0.8 Split Dataframe

```

Figure 18: Accuracy according to Models and Different Test Train Split.

## 4 Implementation and Model Integration

### 4.1 Integration: Facial Recognition and Music Recommender.

In this chapter we will be connecting the building block together. On one side we have the Facial Emotion Recognition by CNN and on the other we have the Song Mood Prediction.

#### 4.1.1 AOutput of the CNN.

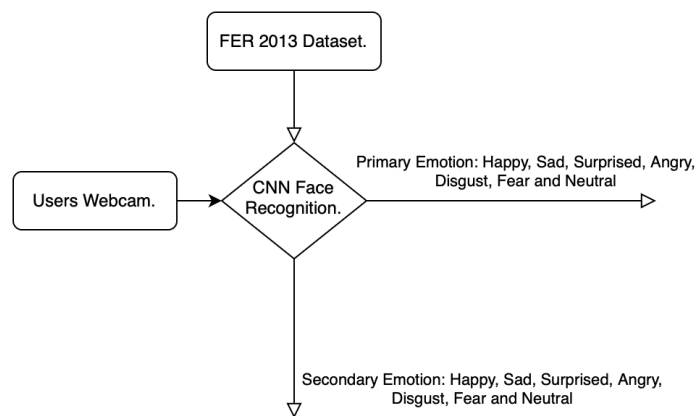


Figure 19: Facial Recognition.

We will first indulge in the Facial Recognitions CNN model. The fig 19 gives us the overview of what we have already covered in section 4. Just for test purposes, I have facilitated a key press button key 'q' from the keyboard such that once the button is pressed the webcam is turned off and the emotion is predicted.

This function we mentioned above can be seen in fig. 20. The first line of this code takes in the number of emotions detected in the span of time the webcam was on and turns the respective value into a dictionary with the key as the names of the emotions and the value as the times it was encountered.

This would generate the output for the CNN a simple list of just two entries namely a Primary and Secondary Emotion.

#### 4.1.2 Output of the RNN.

As you have read in above section already about the RNN Song Prediction or Song Recommendation in detail, the fig 21 gives us the Flowchart of how it works in a superficial sense. The dataset from Spotify namely the Music Mood Dataset, the emotion list from

```
def top2emo(emo):
    max_count = max(emo.values())
    max_emo = max(emo, key=emo.get)

    max2 = 0
    for feeling, v in emo.items():
        if (v>max2 and v<max_count):
            max2 = v
            max2_emo = feeling

    # Handling if there is just one major emotion.
    if max2 == 0:
        max2_emo = 'Neutral'

    # Handling if 1st Emotion is Neutral
    if max_emo == 'Neutral':
        temp = max2_emo
        max2_emo = max_emo
        max_emo = temp
    return(max_emo, max2_emo)
```

Figure 20: Function to get the top two emotions.

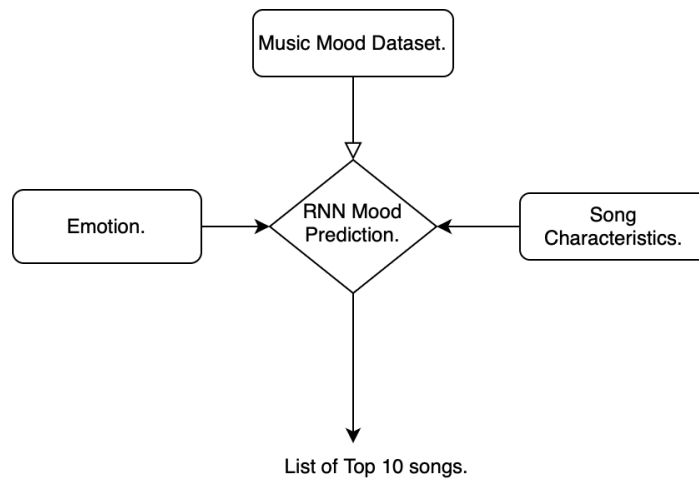


Figure 21: Song Prediction.

the facial emotion recognition CNN model and the characteristics of the songs taken in as features from the music mood dataset are used to predicts a list of 10 top popular songs that are suitable for this emotion.

### 4.1.3 Generating the Song List.

The Song List is now generated by 2 emotions detected by the CNN network. The Fig 22 gives the method it can be done. The CNN model has a prediction of 7 emotions; Happy, Sad, Neutral, Fear, Disgust, Surprised and Angry, and the Music Mood generates 4 different moods; Calm, Energetic, Sad and Happy.

```
# Happy

if( primary_emo=='Happy' and secondary_emo =='Neutral'):
    Play = music_player[music_player['mood'] =='Energetic' ]
    Play = Play.sort_values(by="popularity", ascending=False)
    Play = Play[:5].reset_index(drop=True)

if( primary_emo=='Happy' and secondary_emo =='Fear'):
    Play= music_player.loc[(music_player['mood'] =='Calm') & (music_player['SE']=='Happy')]
    Play = Play.sort_values(by="popularity", ascending=False)
    Play = Play[:5].reset_index(drop=True)
```

Figure 22: Function to bridge the CNN and RNN.

## 4.2 Creating the Runner to facilitate execution.

The runner is a piece of code that executes all the different functions and pieces of code together. Fig 23 shows the entire runner code.

```
from Face_Emotion_run import avr_emotion, detect_emo, top2emo
from Song_Mood import Recommend_Songs
from SoundCloud import Play_Songs

# Start Face-Emotion Detection.
detected_emotions = detect_emo()
emo = avr_emotion(detected_emotions)

# Extract Emotions
maxemo, max2emo = top2emo(emo)
print(emo, '\n', maxemo, max2emo)

# Recommend Songs
songs=Recommend_Songs(maxemo,max2emo)
print('\n',"-----")
print('Songs Recommended:')
print(songs)

# Start SoundCloud
print('\n',"-----")
Play_Songs[songs]
```

Figure 23: Runner Code.



1. First the Face-Emotion needs to be detected, in order to do this we save all the detected emotion in the variable named `detected_emotion` and send this variable to get the top two emotions by calling the `avr_emotion` and saving this list of two prominent emotions into the `emo` variable.
2. Next we extract the Primary and Secondary emotion by calling the `top2emo` function and just for testing purpose we shall retain this and print the top two prominent emotions.
3. Once this is completed we can now recommend the songs. For this we will pass in the Primary emotion(`maxemo`) and Secondary emotion(`max2emo`). This statement will give a list of top 10 songs and again for testing purpose we shall print the list of songs recommended.
4. Finally with the song list we have we shall now call the `Play_Songs` function which will open SoundCloud and pass each song as an argument, play it and if the user wants to go to the next song, it will skip the current song and play the next as well. All completely automated with tab changes and mouse clicks.

### 4.3 Results.

The results for a complete execution is shown in the figure below. Fig 4.3 shows a snapshot of the emotion captured of the user, this once processed gives an intermediate result. After a couple of seconds the SoundCloud application is invoked on Chrome and the first song from the list is searched and played automatically. As this is commenced a popup list in the terminal is processed where the user can enter 1 to skip the current song or 0 to end the program. If the choice is 1 the window changes back to the SoundCloud application and the next song from the list begins to play automatically.

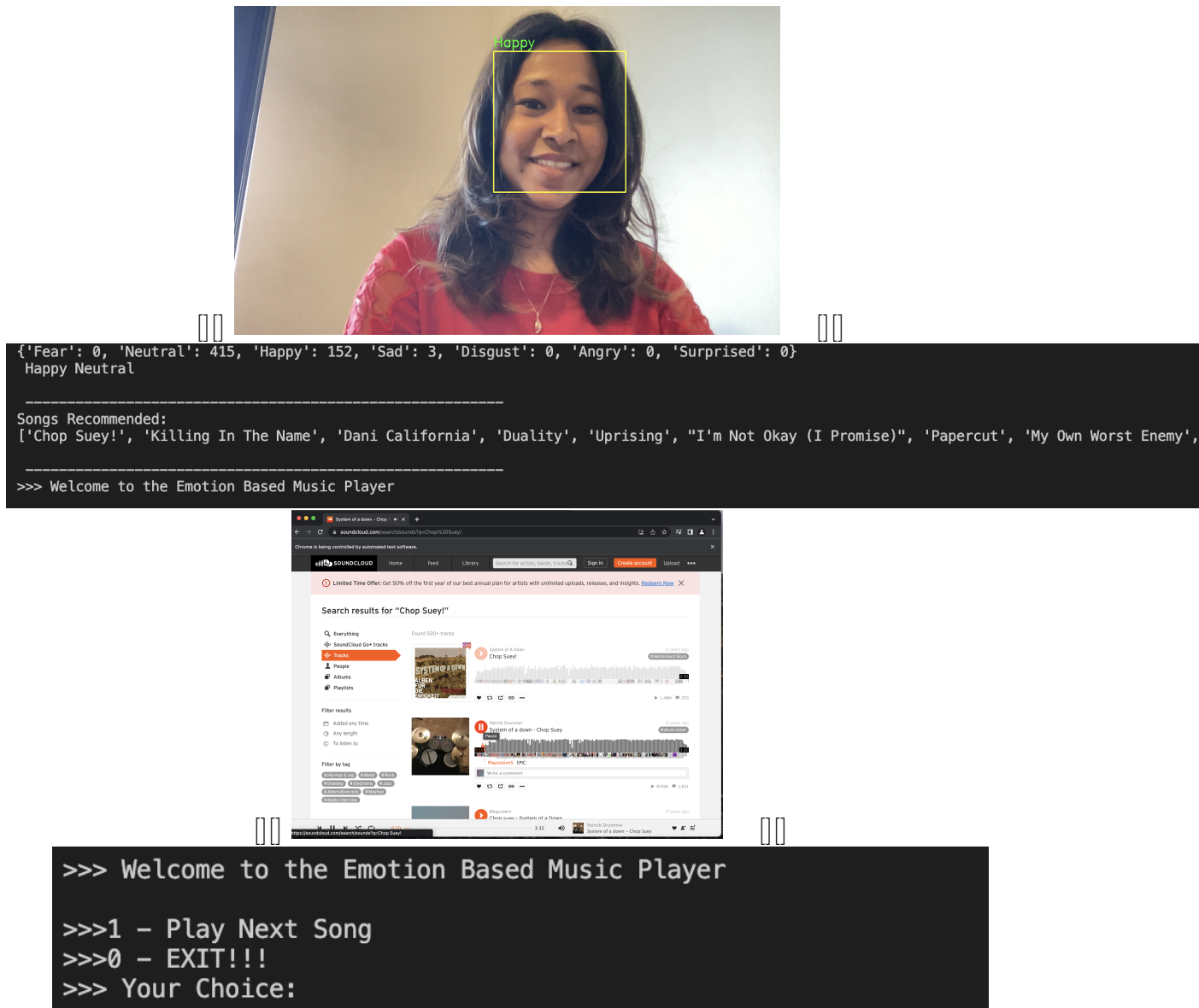


Figure 24: A set of four subfigures: `fig:ResultFace` describes the first subfigure; `fig:resultemo` describes the second subfigure; `fig:SoundCloudRes` describes the third subfigure; `fig:SoundSkip` describes the last subfigure.