

INDEX

TOPICS	Page No's
➤ Certificates	
➤ Acknowledgement	
➤ Abstract	
➤ Figures/Tables	
CHAPTER-1: INTRODUCTION	1
CHAPTER-2: LITERATURE SURVEY	2-3
CHAPTER-3: SYSTEM ANALYSIS	
3.1 Existing System	4
3.2 Proposed System	4-5
CHAPTER-4: SYSTEM REQUIREMENTS	
4.1 Hardware Requirements	7
4.2 Software Requirements	7
CHAPTER-5: SYSTEM STUDY	
5.1 Feasibility Study	8
5.2 Feasibility Analysis	9
CHAPTER-6: SYSTEM ARCHITECTURE	
6.1 Data Flow Diagram	9
6.2 UML Diagrams	12-15
CHAPTER-7: INPUT AND OUTPUT DESIGN	
7.1 Input Design	16
7.2 Output Design	17

CHAPTER-8: IMPLEMENTATION

8.1MODULES 18

8.1.1 Module Description 18-19

CHAPTER-9: SOFTWARE ENVIRONMENT

9.1 PYTHON 20-43

9.2 Source Code 44-48

CHAPTER-10: RESULTS/DISCUSSIONS

10.1 System Test 49-52

10.2 Output Screens 52-56

CHAPTER-11: CONCLUSION 57

CHAPTER-12: REFERENCES/BIBLIOGRAPHY 58

LIST OF FIGURES

FIGURES/TABLES

S.No	Figures/Tables	Page No.
1	Data Flow Diagrams	9
2	UML Diagrams	12-14
3	Compiling and interpreting python source code	37-43
4	Output Screens	53-56

EMOTION BASED MUSIC RECOMMENDATION SYSTEM USING WEARABLE PHYSIOLOGICAL SENSORS

ABSTRACT

Most of the existing music recommendation systems use collaborative or content based recommendation engines. However, the music choice of a user is not only dependent to the historical preferences or music contents. But also dependent to the mood of that user. This paper proposes an emotion based music recommendation framework that learns the emotion of a user from the signals obtained via wearable physiological sensors. In particular, the emotion of a user is classified by a wearable computing device which is integrated with a galvanic skin response (GSR) and photo plethysmography (PPG) physiological sensors. This emotion information is feed to any collaborative or content based recommendation engine as a supplementary data. Thus, existing recommendation engine performances can be increased using these data. Therefore, in this paper emotion recognition problem is considered as arousal and valence prediction from multi-channel physiological signals. Experimental results are obtained on 32 subjects' GSR and PPG signal data with/out feature fusion using decision tree, random forest, support vector machine and k-nearest neighbors algorithms. The results of comprehensive experiments on real data confirm the accuracy of the proposed emotion classification system that can be integrated to any recommendation engine.

Index Terms—Emotion Aware Recommendation Engine, Emotion Recognition, Galvanic Skin Response, Machine Learning, Physiological Signals, Photo Plethysmography.

CHAPTER-1

INTRODUCTION

WEARABLE computing is the study or practice of inventing, designing, building or using body-worn computational and sensory devices that leverages a new type of human-computer interaction with a body-attached component that is always up and running. As the number of wearable computing device users are growing every year, their areas of utilization are also rapidly increasing. They have influenced medical care, fitness, aging, disabilities, education, transportation, finance, gaming, and music industries [1], [2] . Recommendation engines are algorithms which aim to provide the most relevant items to the user by filtering useful information from a huge pool of data. Recommendation engines may discover data patterns in the data set by learning user's choices and produce the outcomes that co-relates to their needs and interests [3]. Most of the recommender systems do not consider human emotions or expressions. However, emotions have noticeable influence on daily life of people. For a rich set of applications including human-robot interaction, computer aided tutoring, emotion aware interactive games, neuro marketing, socially intelligent software apps, computers Manuscript received March 31, 2018; revised May 26, 2018. The authors are with Department of Computer Engineering, Faculty of Computer and Informatics, Istanbul Technical University, Istanbul, Turkey (email:{ayatadeger, yyaslan, kamasak}@itu.edu.tr). should consider the emotions of their human conversation partners. Speech analytics and facial expressions [4], [5] have been used for emotion detection. However, in case of human beings prefer to camouflage their expressions, using only speech signals or facial expression signals may not be enough to detect emotions reliably. Compared with facial expressions, using physiological signals is a more reliable method to track and recognize emotions and internal cognitive processes of people. Our motivation in this work is to use emotion recognition techniques with wearable computing devices to generate additional inputs for music recommender system's algorithm, and to enhance the accuracy of the resulting music recommendations. In our previous works, we have studied emotion recognition from only GSR signals. In this study we are enriching signals with PPG and propose a data fusion based emotion recognition method for music recommendation engines [6]. The proposed wearable attached music recommendation framework utilizes not only the user's demographics but also his/her emotion state at the time of recommendation

CHAPTER-2

LITERATURE SURVEY

- 1. Title:** Emotion Based Music Recommendation System Using Wearable Physiological Sensors

Authors: Deger Ayata; Yusuf Yaslan; Mustafa E. Kamasak

Abstract: Most of the existing music recommendation systems use collaborative or content based recommendation engines. However, the music choice of a user is not only dependent to the historical preferences or music contents. But also dependent to the mood of that user. This paper proposes an emotion based music recommendation framework that learns the emotion of a user from the signals obtained via wearable physiological sensors. In particular, the emotion of a user is classified by a wearable computing device which is integrated with a galvanic skin response (GSR) and photo plethysmography (PPG) physiological sensors. This emotion information is feed to any collaborative or content based recommendation engine as a supplementary data. Thus, existing recommendation engine performances can be increased using these data. Therefore, in this paper emotion recognition problem is considered as arousal and valence prediction from multi-channel physiological signals. Experimental results are obtained on 32 subjects' GSR and PPG signal data with/out feature fusion using decision tree, random forest, support vector machine and k-nearest neighbors algorithms. The results of comprehensive experiments on real data confirm the accuracy of the proposed emotion classification system that can be integrated to any recommendation engine

- 2. Title:** EMOTION BASED MUSIC RECOMMENDATION SYSTEM

Authors: M.Sree Van, N.Sree Divya.

Abstract: : Music plays a very important role in enhancing an individual's life as it is an important medium of entertainment for music lovers and listeners and sometimes even imparts a therapeutic approach. In today's world, with ever increasing advancements in the field of multimedia and technology, various music players have been developed with features like fast forward, reverse, variable playback speed (seek & time compression) local playback,

streaming playback with multicast streams and including volume modulation, genre classification etc. The motivation of this work comes from the possibility of reducing the human effort in creating music playlists manually, thus generating them automatically based on the user's emotional state. The human face plays an important role in knowing an individual's mood. The required input is extracted from the human face directly using a camera. One of the applications of this input can be for extracting the information to deduce the mood of an individual. This data can then be used to get a list of songs that comply with the "mood" derived from the input provided earlier. This eliminates the time-consuming and tedious task of manually Segregating or grouping songs into different lists and helps in generating an appropriate playlist based on an individual's emotional features.

3. Title: Emotion Based Music Recommendation System Using Wearable Physiological Sensors

Authors: Değer Ayata, Y. Yaslan, M. Kamasak

Abstract: Most of the existing music recommendation systems use collaborative or content based recommendation engines. However, the music choice of a user is not only dependent to the historical preferences or music contents. But also dependent to the mood of that user. This paper proposes an emotion based music recommendation framework that learns the emotion of a user from the signals obtained via wearable physiological sensors. In particular, the emotion of a user is classified by a wearable computing device which is integrated with a galvanic skin response (GSR) and photo plethysmography (PPG) physiological sensors. This emotion information is feed to any collaborative or content based recommendation engine as a supplementary data. Thus, existing recommendation engine performances can be increased using these data. Therefore, in this paper emotion recognition problem is considered as arousal and valence prediction from multi-channel physiological signals. Experimental results are obtained on 32 subjects' GSR and PPG signal data with/out feature fusion using decision tree, random forest, support vector machine and k-nearest neighbors algorithms. The results of comprehensive experiments on real data confirm the accuracy of the proposed emotion classification system that can be integrated to any recommendation engine.

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Up to now, most of the research on PDS has focused on how to enforce user privacy preferences and how to secure data when stored into the PDS. In contrast, the key issue of helping users to specify their privacy preferences on PDS data has not been so far deeply investigated. This is a fundamental issue since average PDS users are not skilled enough to understand how to translate their privacy requirements into a set of privacy preferences. As several studies have shown, average users might have difficulties in properly setting potentially complex privacy preferences.

Disadvantages

Personal data we are digitally producing are scattered in different online systems managed by different providers (e.g., online social media, hospitals, banks, airlines, etc). In this way, on the one hand users are losing control on their data, whose protection is under the responsibility of the data provider, and, on the other, they cannot fully exploit their data, since each provider keeps a separate view of them.

3.2 PROPOSED SYSTEM

Personal Data Storage (PDS) has inaugurated a substantial change to the way people can store and control their personal data, by moving from a service-centric to a user-centric model. PDSs enable individuals to collect into a single logical vault personal information they are producing. Such data can then be connected and exploited by proper analytical tools, as well as shared with third parties under the control of end users.

CHAPTER-4

SYSTEM REQUIREMENTS

REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

REQUIREMENT SPECIFICATION

Functional Requirements

- Graphical User interface with the User.

Software Requirements

For developing the application the following are the Software Requirements:

1. Python

Operating Systems supported

1. Windows

Technologies and Languages used to Develop

1. Python

4.1 HARDWARE REQUIREMENTS

1. System : Pentium IV 2.4 GHz.
2. Hard Disk : 40 GB.
3. Ram : 512 Mb.

4.2 Software Requirements

- Operating system : Windows.
- Coding Language : python
- IDE : Pycharm 2020 Community.

CHAPTER-5

SYSTEM STUDY

5.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

5.2 FEASIBILITY ANALYSIS

Three key considerations involved in the feasibility analysis are,

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

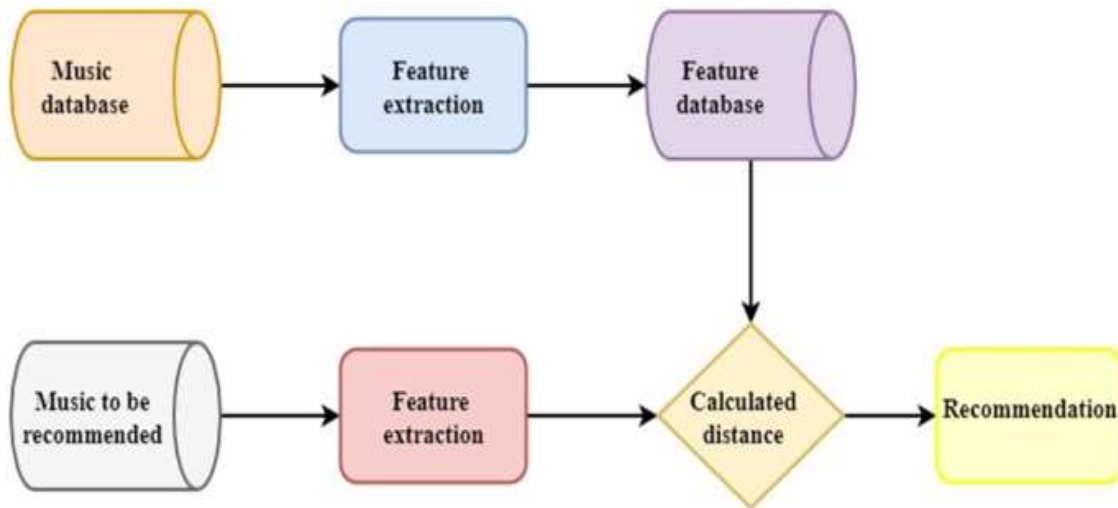
SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER-6

SYSTEM ARCHITECTURE

6.1 DATA FLOW DIAGRAM



6.2 UML DIAGRAMS

UML stands for **Unified Modeling Language**. This object-oriented system of notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation. These renowned computer scientists fused their respective technologies into a single, standardized model. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling object oriented programs.

There are three classifications of UML diagrams

- **Behavior diagrams.** A type of diagram that depicts behavioral features of a system or business process. This includes activity, state machine, and use case diagrams as well as the four interaction diagrams.
- **Interaction diagrams.** A subset of behavior diagrams which emphasize object interactions. This includes communication, interaction overview, sequence, and timing diagrams.

- **Structure diagrams.** A type of diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object, and package diagrams.

Types of UML Diagrams

UML defines nine types of diagrams: class (package), object, use case, sequence, collaboration, statechart, activity, component, and deployment.

Class Diagrams

Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system.

Package Diagrams

Package diagrams are a subset of class diagrams, but developers sometimes treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependencies between packages.

Object Diagrams

Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.

Use Case Diagrams

Use case diagrams model the functionality of system using actors and use cases.

Sequence Diagrams

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

Collaboration Diagrams

Collaboration diagrams represent interactions between objects as a series of sequenced messages. Collaboration diagrams describe both the static structure and the dynamic behavior of a system.

Statechart Diagrams

Statechart diagrams describe the dynamic behavior of a system in response to external stimuli. Statechart diagrams are especially useful in modeling reactive objects whose states are triggered by specific events.

Activity Diagrams

Activity diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

Component Diagrams

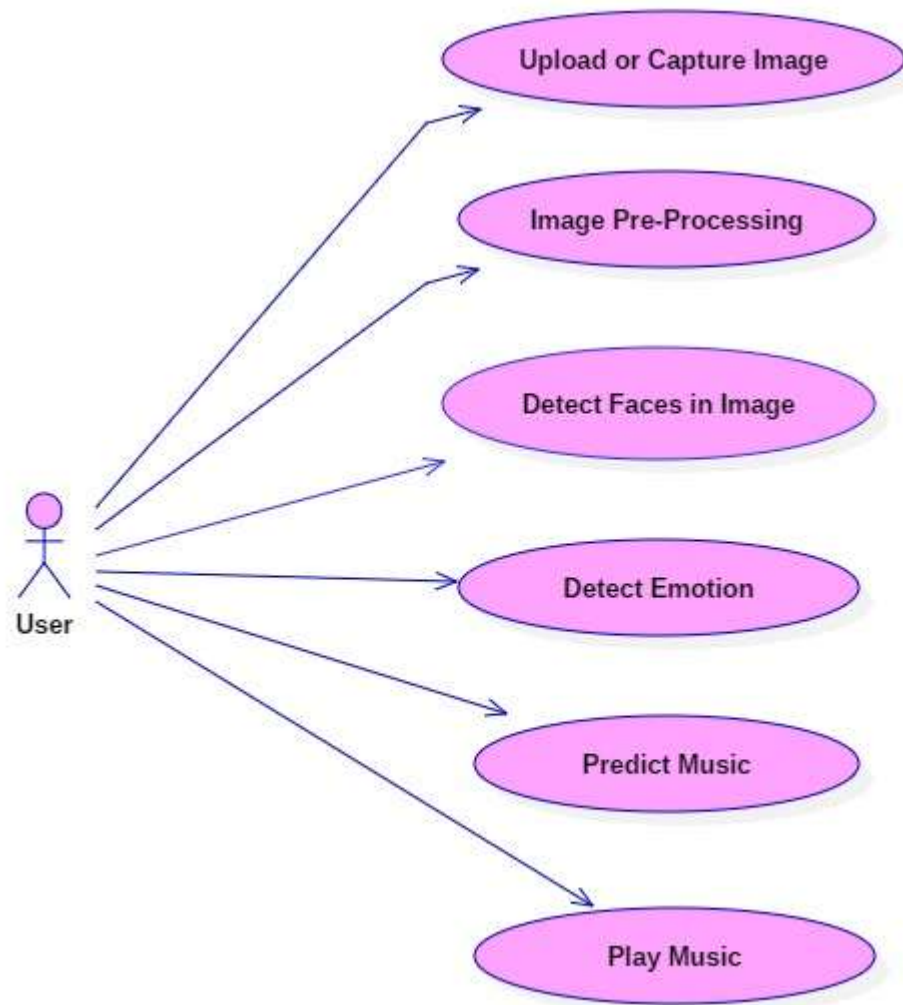
Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executables.

Deployment Diagrams

Deployment diagrams depict the physical resources in a system, including nodes, components, and connections.

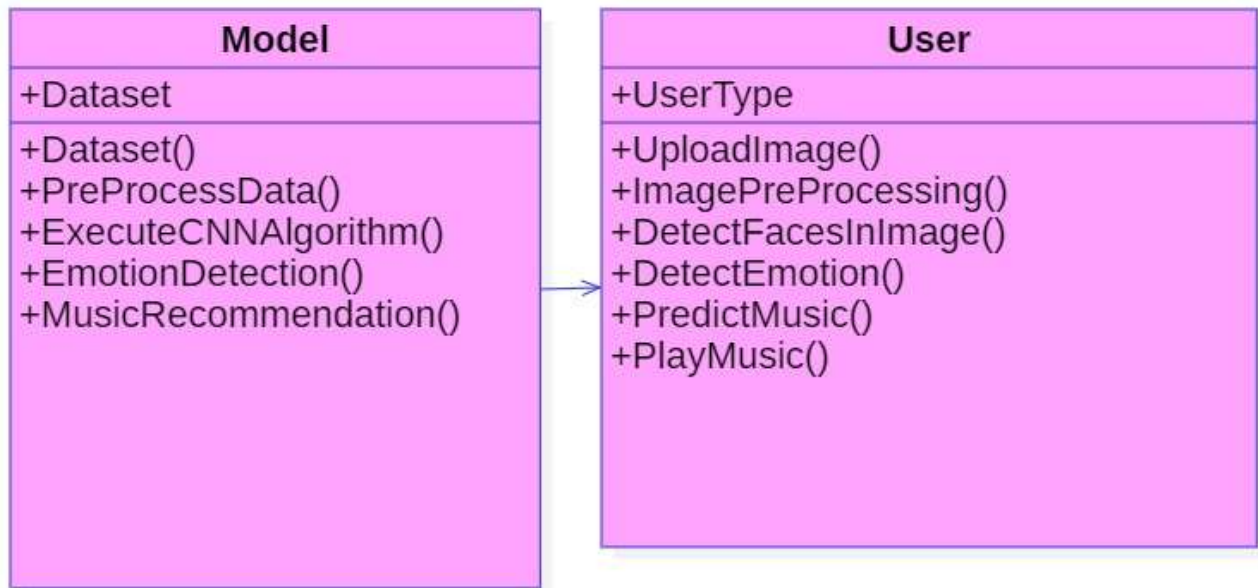
Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



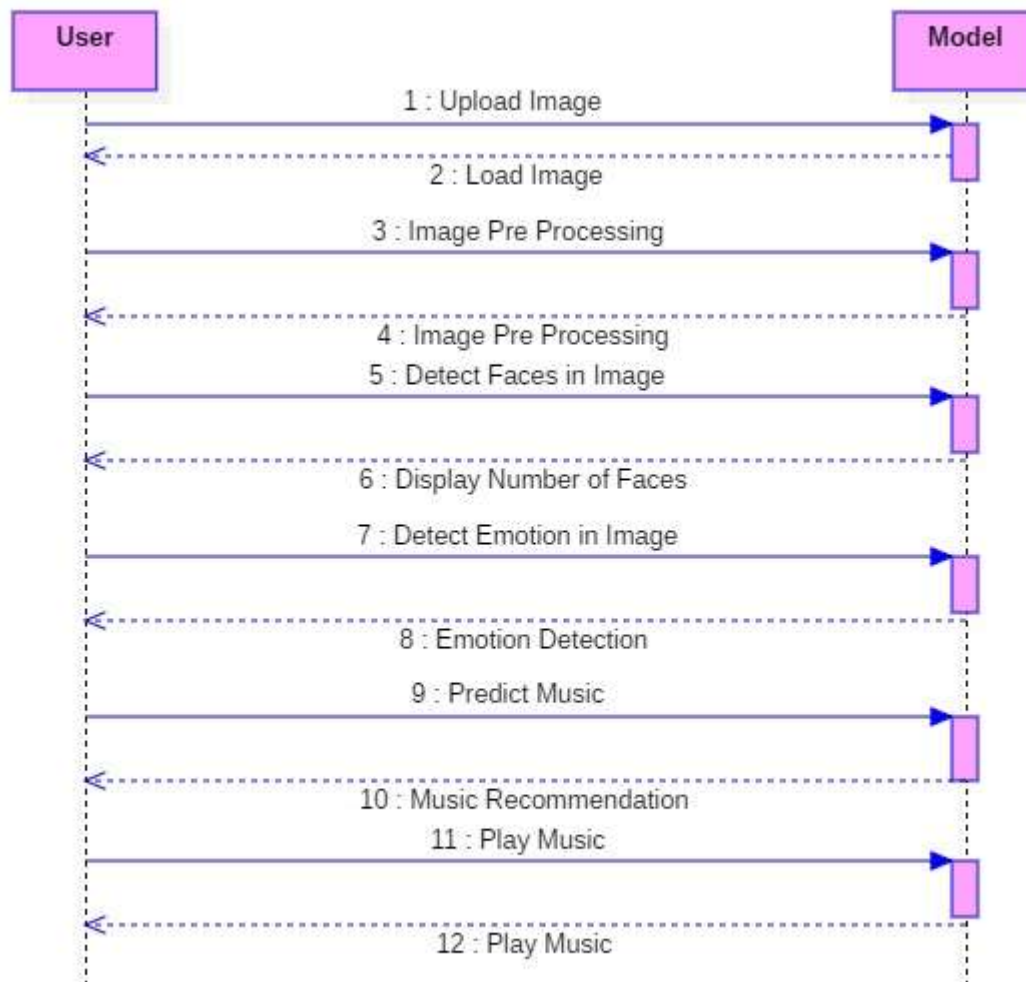
Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



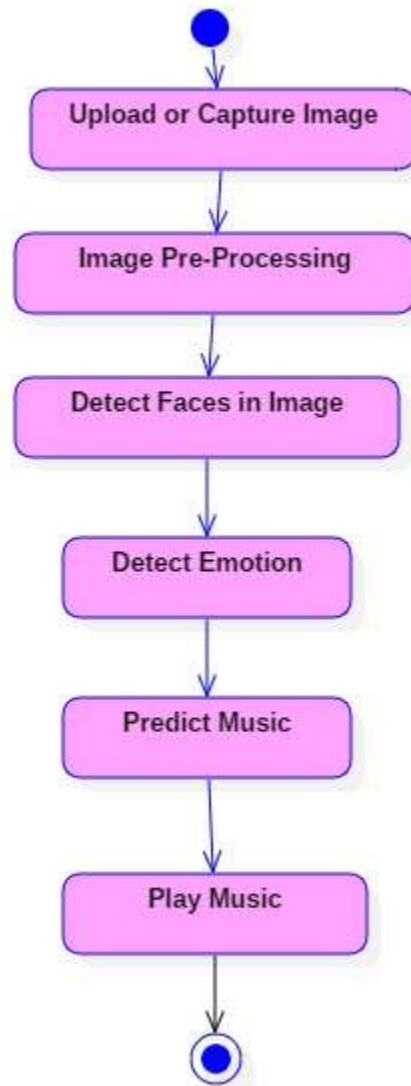
Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



CHAPTER-7

INPUT AND OUTPUT DESIGN

7.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

7.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER-8

IMPLEMENTATION

8.1 MODULES

This project consists of 4 modules

1. DATA COLLECTION
2. DATA PRE-PROCESSING
3. FEATURE EXTRATION
4. EVALUATION MODEL

8.1.1 Module Description

1. DATA COLLECTION

Data used in this paper is a set of product reviews collected from credit card transactions records. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called labelled data.

2. DATA PRE-PROCESSING

Organize your selected data by formatting, cleaning and sampling from it.

Three common data pre-processing steps are: Formatting:

The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file. Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be

removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be removed from the data entirely. Sampling: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

3. FEATURE EXTRACTION

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.

4. Evaluation Model

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid over fitting, both methods use a test set (not seen by the model) to evaluate model performance. Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs. Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions

CHAPTER-9

SOFTWARE ENVIRONMENT

9.1 PYTHON

What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading,*

databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print '**Hello World**'. But in Python, just a print statement will do. It is also quite **easy to learn, understand, and code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbannelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBaseConnectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's

influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning :-

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Learning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction*. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is

that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML –

- Emotion analysis
- Sentiment analysis
- Error detection and prevention

- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term “**Machine Learning**” in 1959 and defined it as a “**Field of study that gives computers the capability to learn without being explicitly programmed**”. And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of **\$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are

specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning** – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected

for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps :-

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the [Google Brain](#) team for internal Google use. It was released under the [Apache 2.0 open-source license](#) on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](#) shells, the [Jupyter](#) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error

charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. [Download the Python Cheatsheet here](#). The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **<https://www.python.org>**



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.6.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 3.7.18	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPU
Unipped source tarball	Source release		6d111671e5b3db4aef77b9ab1b7079be	23017663	345
GZ compressed source tarball	Source release		033e4aee6697051c3bca5ee3604803	17331432	345
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.5 and later	6428b4b71b3da71a41c3b9e08e6	34896438	345
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5d8625c10217e43773b5e4a598d41f	28882845	345
Windows 32-bit file	Windows		0b3990573a1c5682ac58cde0b47rd2	8131763	345
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	9800dc1b0f9c0b0ab0c2184e0172ba2	7564382	345
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	a702b4b4ad7a0e0d03043a583e5d3c00	26881348	345
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	2dc51c6080ef72aee053a3b031b4bd2	1362804	345
Windows x86 embeddable zip file	Windows		9fab1b0138841d79f0a941307412908	6741628	345
Windows x86 executable installer	Windows		71c1902942a5e44a5d84e147e294789	25663848	345
Windows x86 web-based installer	Windows		1b670cfa0d117d0b3c30963ea371287c	1324808	345

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

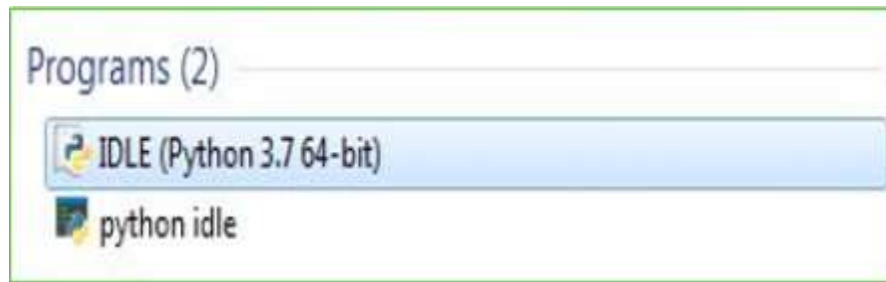
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

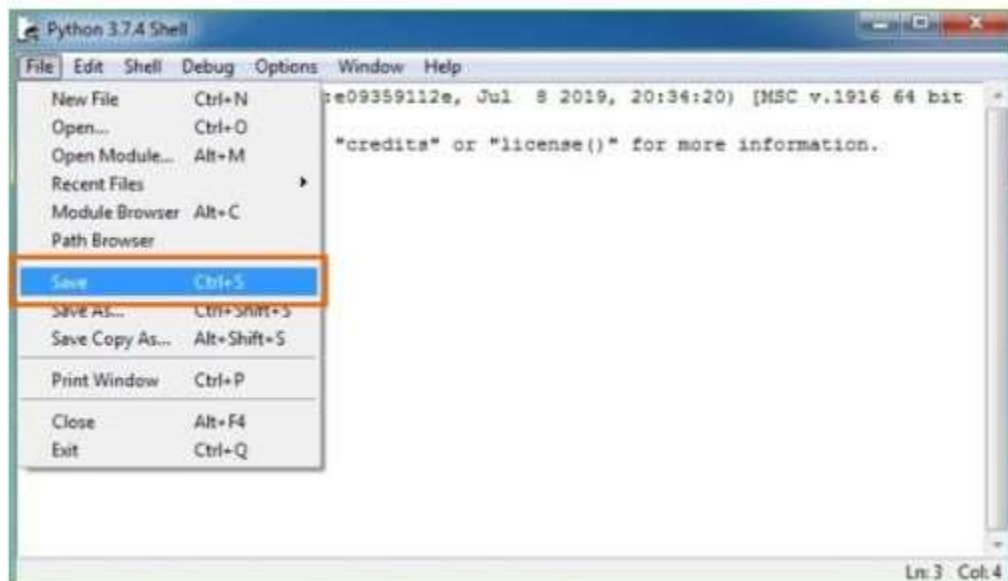
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

9.2 Source Code

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from imutils import paths
import numpy as np
from collections import defaultdict
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import imutils
import cv2
import numpy as np
import sys
from tkinter import ttk
import os
from playsound import playsound

main = tkinter.Tk()
main.title("EMOTION BASED MUSIC RECOMMENDATION SYSTEM")
main.geometry("1200x1200")

global value
global filename
global faces
global frame
```



```

detection_model_path = 'haarcascade_frontalface_default.xml'
emotion_model_path = '_mini_XCEPTION.106-0.65.hdf5'
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
EMOTIONS = ["angry","disgust","scared", "happy", "sad", "surprised","neutral"]

```

```

def upload():
    global filename
    global value
    filename = askopenfilename(initialdir = "images")
    pathlabel.config(text=filename)

```

```

def preprocess():
    global filename
    global frame
    global faces
    text.delete('1.0', END)
    orig_frame = cv2.imread(filename)
    orig_frame = cv2.resize(orig_frame, (48, 48))
    frame = cv2.imread(filename,0)
    faces =
face_detection.detectMultiScale(frame,scaleFactor=1.1,minNeighbors=5,minSize=(30,30),flags=cv2.C
ASCAD_SCALE_IMAGE)
    text.insert(END,"Total number of faces detected : "+str(len(faces)))

```

```

def detectEmotion():
    global faces
    if len(faces) > 0:
        faces = sorted(faces, reverse=True,key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]

```

```

(fX, fY, fW, fH) = faces
roi = frame[fY:fY + fH, fX:fX + fW]
roi = cv2.resize(roi, (48, 48))
roi = roi.astype("float") / 255.0
roi = img_to_array(roi)
roi = np.expand_dims(roi, axis=0)
preds = emotion_classifier.predict(roi)[0]
emotion_probability = np.max(preds)
label = EMOTIONS[preds.argmax()]
messagebox.showinfo("Emotion Prediction Screen", "Emotion Detected As : "+label)
value.clear()
path = 'songs'
for r, d, f in os.walk(path):
    for file in f:
        if file.find(label) != -1:
            value.append(file)
else:
    messagebox.showinfo("Emotion Prediction Screen", "No face detected in uploaded image")

def playSong():
    name = songslst.get()
    playsound('songs/song'+name)

font = ('times', 20, 'bold')
title = Label(main, text='EMOTION BASED MUSIC RECOMMENDATION SYSTEM USING
WEARABLE PHYSIOLOGICAL SENSORS')
title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=80)
title.place(x=5, y=5)

```

```

font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload Image With Face", command=upload)
upload.place(x=50,y=100)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=300,y=100)

preprocessbutton = Button(main, text="Preprocess & Detect Face in Image", command=preprocess)
preprocessbutton.place(x=50,y=150)
preprocessbutton.config(font=font1)

emotion = Button(main, text="Detect Emotion", command=detectEmotion)
emotion.place(x=50,y=200)
emotion.config(font=font1)

emotionlabel = Label(main)
emotionlabel.config(bg='brown', fg='white')
emotionlabel.config(font=font1)
emotionlabel.place(x=610,y=200)
emotionlabel.config(text="Predicted Song")

value = ["Song List"]
songslist = tk.Comboobox(main, values=value, postcommand=lambda:
songslist.configure(values=value))
songslist.place(x=760,y=210)
songslist.current(0)
songslist.config(font=font1)

```

```
playsong = Button(main, text="Play Song", command=playSong)
playsong.place(x=50,y=250)
playsong.config(font=font1)
```

```
font1 = ('times', 12, 'bold')
text=Text(main,height=10,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=300)
text.config(font=font1)
```

```
main.config(bg='brown')
main.mainloop()
```

CHAPTER-10

RESULTS/DISCUSSIONS

10.1 System Test

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software

under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

10.2 Output Screens

In this paper author is describing concept to recommend music to user by detecting moods of user. Existing technique were using collaboration technique which will use previous user data to recommend music to user, if there is no input from previous user then this technique will not useful. This existing technique requires lots of manual work to arrange different music to different categories such as happy, sad or angry etc.

To overcome from above issue author is using ‘Wearable Physiological Sensors’ and this sensors will send signals to application regarding user current status and then this application using SVM (support vector machine) and deep learning neural network algorithms will classify/predict the mood by extracting features from signal. SVM algorithm will be trained with all mood based signals data and whenever a new signal received then SVM/deep learning neural networks will apply new signal on train data to classify user moods. Based on detected user mood song list will be display/recommend to the user.

As student we cannot afford to buy sensors for this project so we are using images with faces to detect user mood and to recommend music. As faces are the best option to predict mood of the users. We tried a lot to search sensors data to classify mood but on internet no such datasets are available.

Implementation Details

In this application I am uploading image and then using python OPENCV i am pre-

processing image to extract features and then this features is applied on SVM/Deep Learning Neural Network Training Model to predict moods of user and based on user mood all songs will be detected and shown in drop down box and user can select any song and play.

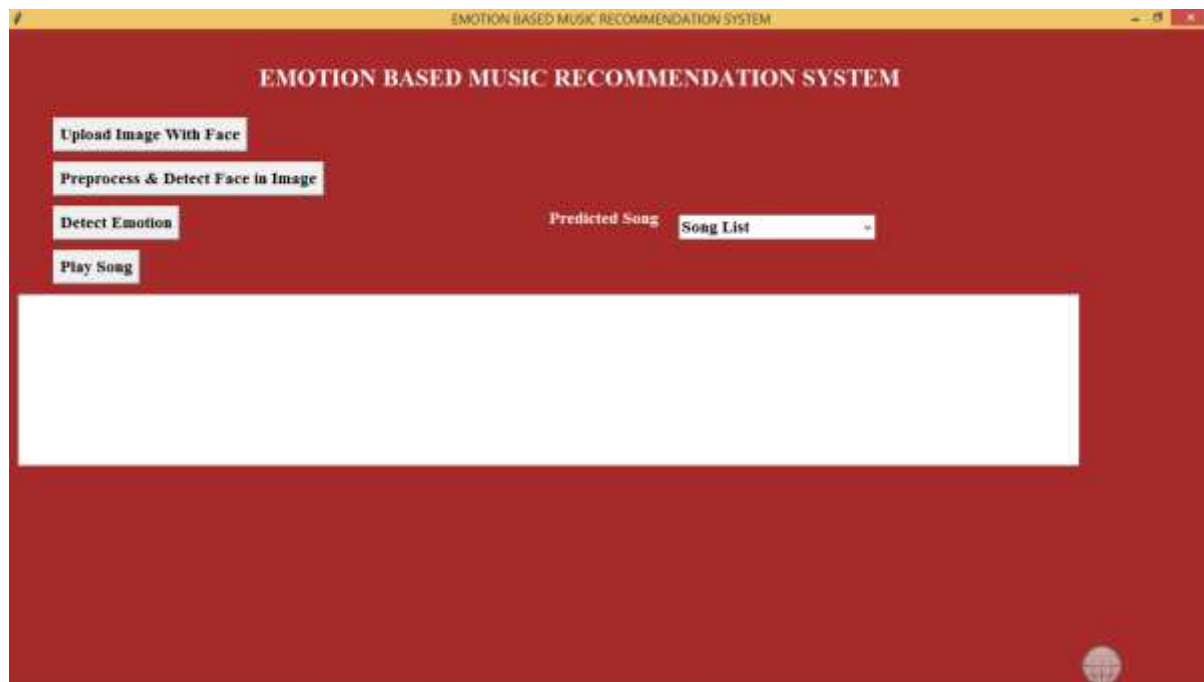
All sample images are in images folder and all songs are in songs folder and u too can include new songs to that folder and given name as happy1.mp3, happy2, happy3 or sad1, sad2 etc. Like this for all categories you can add songs. Currently i am using same song for all moods.

To run project install below package

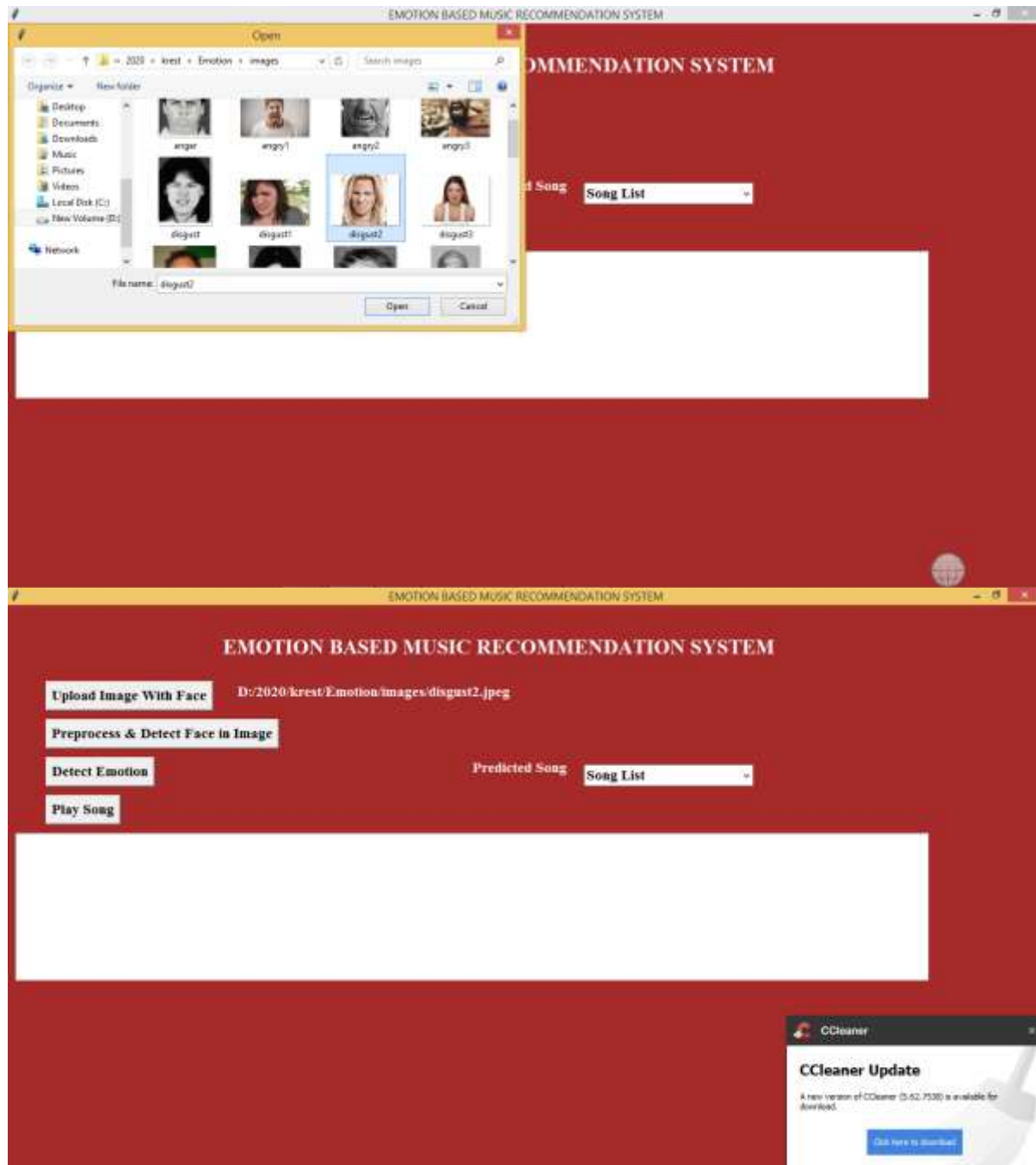
Pip install playsound

Screen shots

Double click on 'run.bat' file to get below screen



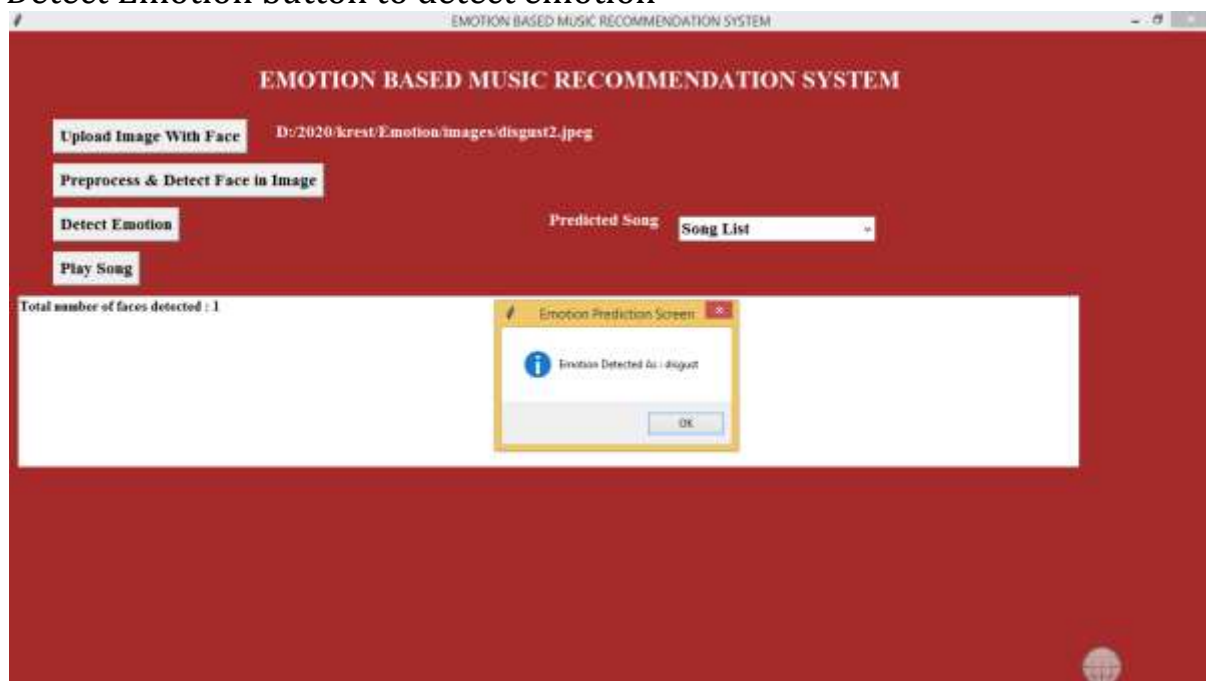
In above screen click on 'Upload Image With Face' button to upload image



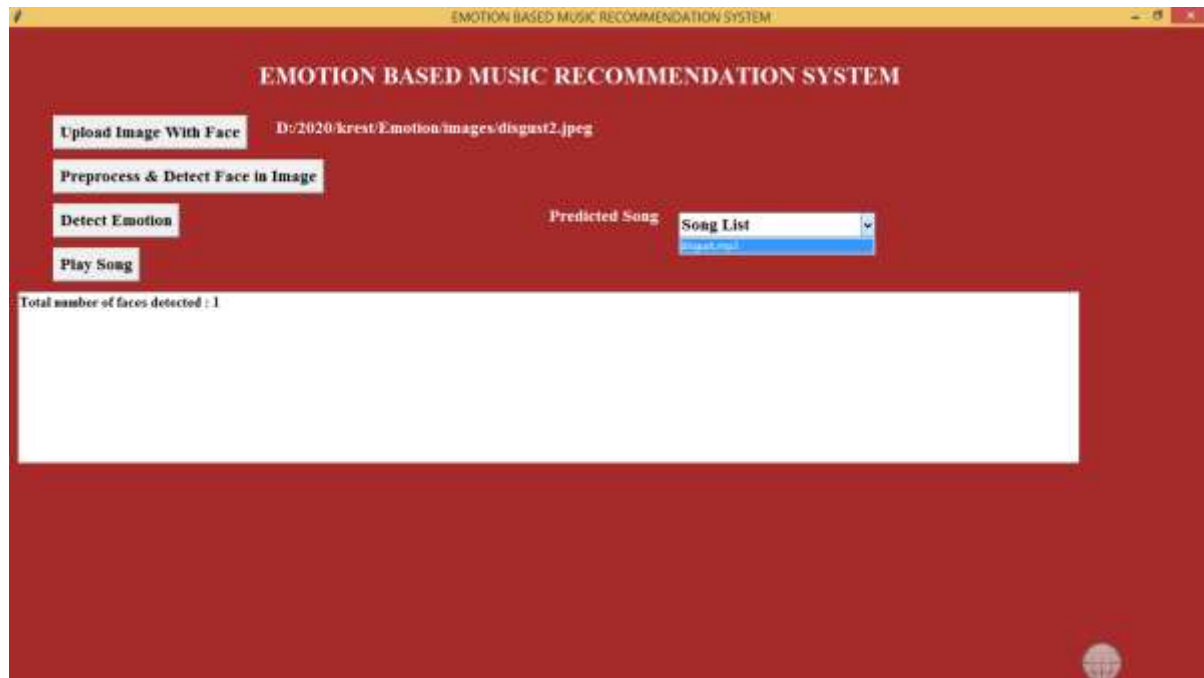
In above screen i am selecting one 'disgust' image. Now click on 'Pre-process & Detect Face In Image' button to perform pre-processing and to extract face from images



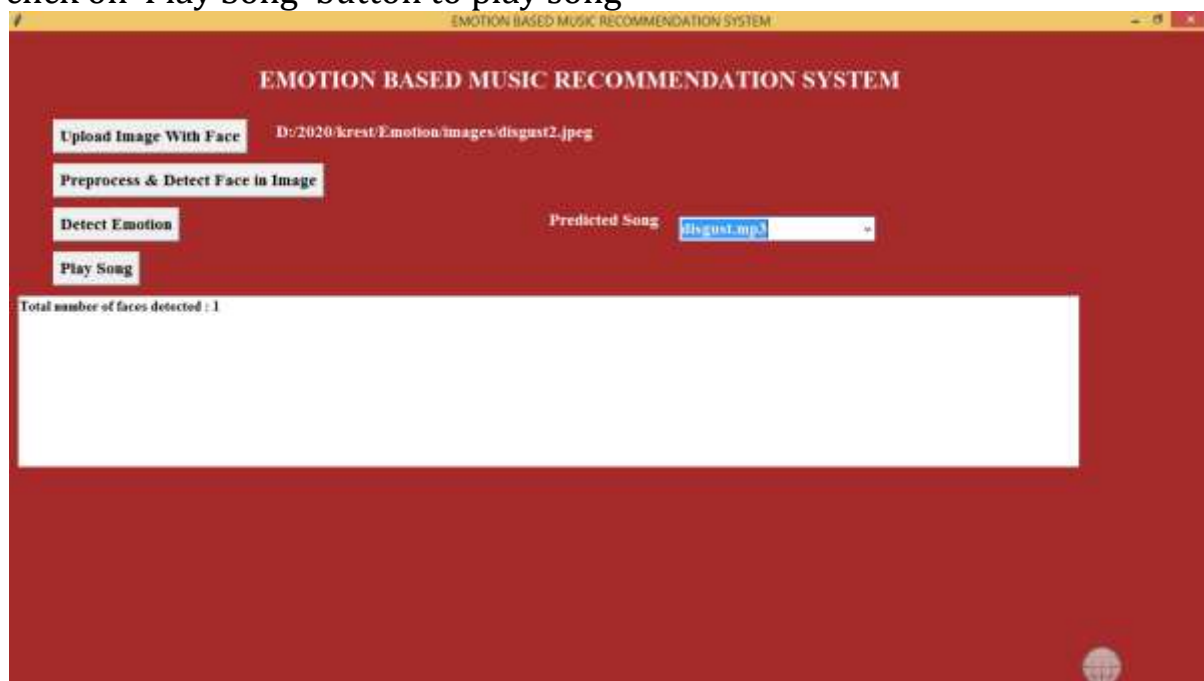
In above screen we can see in uploaded image one face is detected. Now click on Detect Emotion button to detect emotion



In above screen we can see emotion disgust is detected and now click on drop down arrow link to get all disgust songs list



In drop down box we can see 'disgust.mp3' songs is showing, select that song and click on 'Play Song' button to play song



If your system has audio driver then u can hear song.

Note: no technologies can detect 100% emotion from images but this project can detect upto 90%

CHAPTER-11

CONCLUSION

In this study, a framework for enhancing music recommendation engines performance via physiological signals has been introduced. Emotion recognition from multi-channel physiological signals was performed, data fusion techniques were applied to combine data from GSR and PPG sensors and FLF has been implemented. Considering emotion state of the listener improves the performance of recommendations. Recognizing arousal and valence values directly from only GSR and PPG signals is a challenging task. We have showed that there is relationship between GSR and PPG signals and emotional arousal and valence dimensions. For GSR only signal, we have obtained 71.53% and 71.04% accuracy rate for arousal and valence prediction respectively. For photoplethysmography only signal, we have obtained 70.93% and 70.76% accuracy rate for arousal and valence prediction respectively. Fusing GSR and PPG signals we have obtained the results, 72.06% and 71.05% accuracy rate for arousal and valence prediction respectively. Although there is only slight improvement using fusion in emotion recognition accuracy, the proposed framework is promising for music recommendation engines in terms of adding multi modal emotion phenomenon into music recommendation logic. Performance can be improved with the advancement of wearable sensor technologies and using different type of sensors. Using more than one sensor may also help for failure management. As future work, we will consider different combination of sensors that handle the failures of wearable sensors and additional sensors usage to increase performance. The results of this study can be used to increase user experience of multimedia tools and music recommendation engines. Since there is high correlation between physiological GSR and PPG data and affective state and cognitive state of a person multimedia recommendation engines can benefit from physiological computing systems.

CHAPTER-12

REFERENCES/BIBLIOGRAPHY

1. S. Jhajharia, S. Pal, and S. Verma, “Wearable computing and its application,” *Int. J. Comp. Sci. and Inf. Tech.*, vol. 5, no. 4, pp. 5700– 5704, 2014.
2. K. Popat and P. Sharma, “Wearable computer applications: A feature perspective,” *Int. J. Eng. and Innov. Tech.*, vol. 3, no. 1, 2013.
3. P. Melville and V. Sindhwani, “Recommender systems,” in *Encyc. of mach. learn.* Springer, 2011, pp. 829–838.
4. N. Sebe, I. Cohen, T. S. Huang et al., “Multimodal emotion recognition,” *Handbook of Pattern Recognition and Computer Vision*, vol. 4, pp. 387– 419, 2005.
5. R. W. Picard, E. Vyzas, and J. Healey, “Toward machine emotional intelligence: Analysis of affective physiological state,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 10, pp. 1175–1191, 2001.
6. D. Ayata, Y. Yaslan, and M. Kamasak, “Emotion recognition via galvanic skin response: Comparison of machine learning algorithms and feature extraction methods,” *IU J. of Elect. & Elect. Eng.*, vol. 17, no. 1, pp. 3129–3136, 2017.
7. P. Ekman, R. W. Levenson, and W. V. Friesen, “Autonomic nervous system activity distinguishes among emotions.” *Am. Assoc. for Adv. of Sci.*, 1983.
8. I.-h. Shin, J. Cha, G. W. Cheon, C. Lee, S. Y. Lee, H.-J. Yoon, and H. C. Kim, “Automatic stress-relieving music recommendation system based on photoplethysmography-derived heart rate variability analysis,” in *IEEE Int. Conf. on Eng. in Med. and Bio. Soc. IEEE*, 2014, pp. 6402–6405.
9. S. Nirjon, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, G. Shen, and F. Zhao, “Musicalheart: A hearty way of listening to music,” in *Proc. of ACM Conf. on Emb. Netw. Sens. Sys. ACM*, 2012, pp. 43–56.
10. H. Liu, J. Hu, and M. Rauterberg, “Music playlist recommendation based on user heartbeat and music preference,” in *Int. Conf. on Comp. Tech. and Dev.*, vol. 1. IEEE, 2009, pp. 545–549.
11. F. Isinkaye, Y. Folajimi, and B. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egypt. Inf. J.*, vol. 16, no. 3, pp. 261–273, 2015.