

ECG BASED EMOTION RECOGNITION USING NEURAL NETWORK

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By
RUTHWIK BHARADWAJ
(REG.NO.37110003)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL
OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of RUTHWIK BHARADWAJ .(REG.NO)37110003 who carried out the project entitled **"ECG BASED EMOTION RECOGNITION USING NEURAL NETWORK"** under my supervision from **August 2020** to **March 2021**.

Internal Guide

Dr.A.Jesudoss M.E.,PH.D.

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **RUTHWIK BHARADWAJ** hereby declare that the Project Report entitled “**ECG BASED EMOTION RECOGNITION USING NEURAL NETWORK**” is done by me under the guidance of **Dr. A. JESUDOSS, M.E.,PH.D.** Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide of **Dr.A.JESUDOSS.M.E.,PH.D.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ECG BASED EMOTION RECOGNITION USING NEURAL NETWORK

ABSTRACT

Emotions are fundamental to humans. They affect perception and everyday activities such as communication, learning and decision making. Various emotion recognition devices have been developed incorporating facial expressions, body postures and speech recognitions as a means of recognition. The accuracy of most of the existing devices is dependent on the expressiveness of the user and can be fairly manipulated. We proposed a physiological signal based solution to provide reliable emotion classification without possible manipulation and user expressiveness. Electrocardiogram (ECG) signals are extracted using shimmer sensors and are used for recognition of basic human emotions.

The statistical analysis is intended as a preliminary feature selection procedure aimed to select the parameters that significantly vary between the arousal and valence levels, using data gathered from the training set exclusively.

CHAPTER 1

INTRODUCTION

Overview

To predict different emotion states, the hidden states of all iteration processes are projected to desired dimension. During the training process, we adopt the cross entropy function to measure the similarities between predicted label vectors and real label vectors.

To investigate the influences of unbalanced training data, we conducted positive-negative-neutral classification protocol and the data of negative emotions were larger than positive emotions and neutrality, which was challenging with these unbalanced training data and testing data.

Scope of the Project

The main contributions of this project therefore are:

- ▶ Data Analysis
- ▶ Dataset Preprocessing
- ▶ Training the Model
- ▶ Testing of Dataset

Domain Overview

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a

pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unravelling this wealth of information and are increasingly adapting to AI systems for automated support.

Deep Learning About

Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures. Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame.

DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and

complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very

fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.

- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays. an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as

wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib,

and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general- purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

Deep Learning

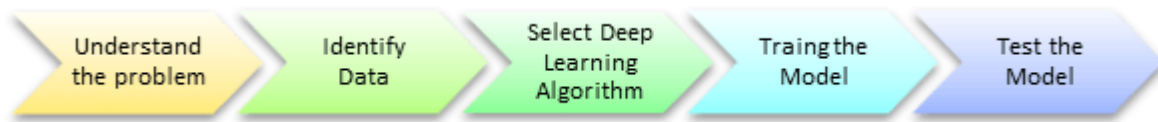
Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer

- All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.



To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

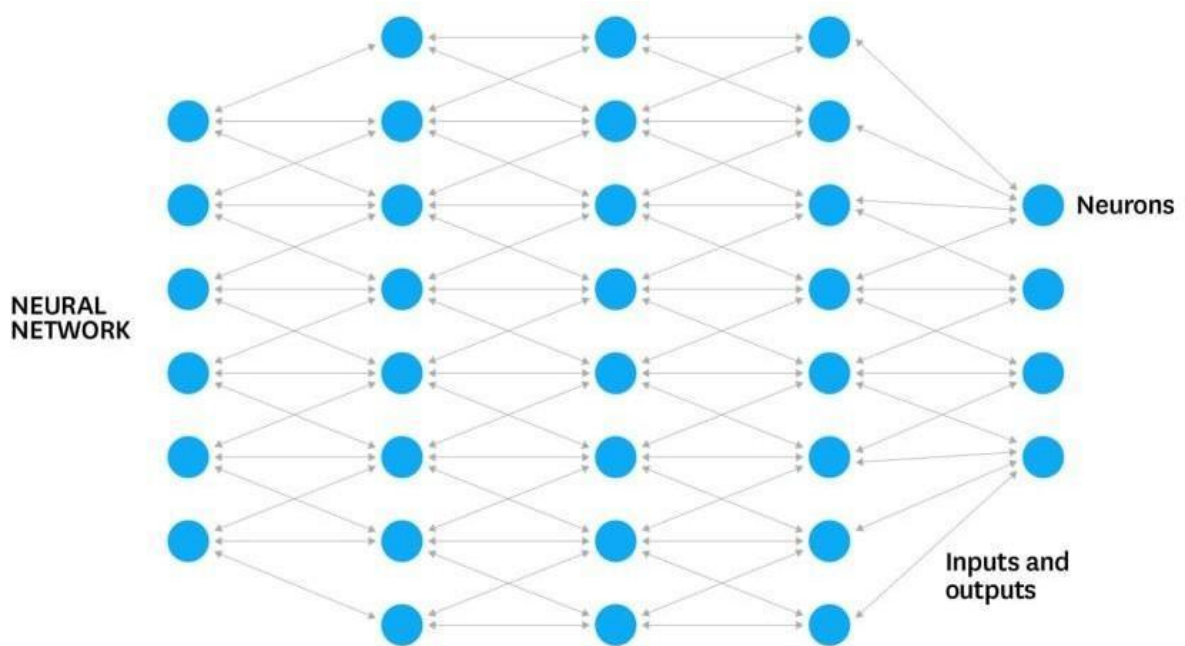
1. The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
2. The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

1. Shallow neural network: The Shallow neural network has only one hidden layer between the input and output.
2. Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN).



Neuron

Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters. Inputs accepted by the neuron are separately weighted.

Inputs are summed and passed through a non-linear function to produce output. Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes

Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another. Inputs

CHAPTER 2

LITERATURE SURVEY

MPED: A Multi-Modal Physiological Emotion Database for Discrete Emotion Recognition

Literature Survey 1	
Title	MPED: A Multi-Modal Physiological Emotion Database for Discrete Emotion Recognition
Authors	TENGFEI SONG, WENMING ZHENG, (SENIOR MEMBER, IEEE), CHENG LU ¹ , YUAN ZONG, XILEI ZHANG ¹ AND ZHEN CUI , (Member, IEEE)
Published Year	2019
Efficiency	✓ Find emotion-relevant features and to correlate them with emotional states ✓ Effectively select significant ECG features associated with emotions ✓ Reduce the dimensions of the selected features
Drawbacks	□ Multi-modal data features are not combined for emotional detection □ More hidden information are not obtained □ Does not measure positive and negative emotions in real time

Description	To alleviate the influence of culture dependent elicitation materials and evoke desired human emotions, we specifically collect an emotion elicitation material database selected from more than 1500 video clips. By considerable amount of strict man-made labelling, we elaborately choose 28 videos as standardised elicitation samples, which are assessed by psychological methods. The physiological signals of participants were synchronously recorded when they watched these standardised video clips that described six discrete emotions and neutral emotion. With three types of classification protocols, different feature extraction methods and classifiers (SVM and KNN) were used to recognize the physiological responses of different emotions, which presented the baseline results. Simultaneously, we present a novel attention-LSTM (A- LSTM) which strengthens the effectiveness of useful sequences to extract more discriminative features. After signals acquisition, a key process was feature extraction from these physiological signals. Effective features were easier to be distinguished by the classification model. In this section, we presented different feature extraction methods and a slide window of one second with no overlap was conducted to divide the physiological signals into many 1-second samples. Before EEG feature extraction, we conducted the independent component analysis (ICA) to remove electrooculography (EOG) artifacts. The raw EEG signals were decomposed to many independent components, which consist of EOG artifact and EEG signals. A classifier trained was used to identify the EOG artifacts and EEG signals, which were used to reconstruct the EEG signals without EOG artifacts. We used the fast fourier transform (FFT) to calculate the discrete fourier transform (DFT). A 1000 samples window was applied for the estimation of PSD from the five frequency bands respectively. The logarithms of the PSD from different frequency bands were used as the features. ECG signals describe the changes of muscular contraction related with cardiovascular activity. To make ECG signals
-------------	--

	easier for classification, we extracted the frequency domain features and time domain features respectively.
--	--

DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices

Literature Survey 2	
Title	DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices
Authors	Stamos Katsigiannis and Naeem Ramzan, Senior Member, IEEE,
Published Year	2015
Efficiency	✓ Allow the use of affective computing methods in everyday applications ✓ Ability to Deliver High Quality Results ✓ Elimination the need for Data Labeling
Drawbacks	□ It is not appropriate when the dataset available is large □ Lead to the generation of bad candidate proposals □ Cannot be implemented real time
Description	All the signals were captured using portable, wearable, wireless, low- cost and off-the-shelf equipment that has the potential to allow the use of affective computing methods in everyday applications. A baseline for participant-wise affect recognition using EEG and ECG -based features, as well as their fusion, was established through supervised classification experiments using Support Vector Machines (SVMs). The selfassessment of the participants was evaluated through comparison with the self-assessments from another study using the same audio-visual stimuli. Classification results for valence, arousal and dominance of the proposed database

are comparable to the ones achieved for other databases that use non-portable, expensive, medical grade devices. These results indicate the prospects of using low-cost devices for affect recognition applications. The proposed database will be made publicly available in order to allow researchers to achieve a more thorough evaluation of the suitability of these capturing devices for affect recognition applications.

The captured EEG signals are contaminated with noise and artefacts that were detected but did not originate from the brain. Signals caused from cardiac activity, eye movement and muscular activity, as well as power line noise are also captured by the EEG device and thus downgrade the quality of the recorded data, making the use of denoising methods a necessity. On the other hand, ECG signals are less susceptible to interferences due to their higher voltage amplitudes and thus require no further processing.

The ASR method consists of a sliding window Principal Component Analysis (PCA), which statistically interpolates any highvariance signal components exceeding a threshold relative to the covariance of an automatically detected data section that is relatively free of artefacts. The final step for preparing the EEG data for further analysis is the application of the Common Average Reference (CAR) method, as recommended by Cohen [32], which computes the average value over all electrodes and subtracts it from each

sample of each electrode. Removal of bad channels is performed prior to the CAR method in order to avoid introducing noise to all channels due to a possible bad channel. Due to the variable duration of the film clips used as stimuli and in order to allow time for a specific emotion to become dominant, the signal recordings corresponding to the last 60 seconds of each film clip were used for further feature extraction and analysis. The higher frequency components of EEG signals carry more important information regarding positive emotions compared to negative ones (high and low valence respectively), while a correlation between increased beta power and positive emotional self-induction.

ASCERTAIN: Emotion and Personality Recognition using Commercial Sensors

Literature Survey 3	
Title	ASCERTAIN: Emotion and Personality Recognition using Commercial Sensors
Authors	Ramanathan Subramanian, Senior Member, IEEE, Julia Wache Student Member, IEEE, Mojtaba Khomami Abadi, Student Member, IEEE, Radu L. Vieriu, Member, IEEE, Stefan Winkler, Senior Member, IEEE, Nicu Sebe, Senior Member, IEEE
Published Year	2016
Efficiency	✓ Supports linear and non-linear physiological correlates of emotion and personality ✓ Attempt binary emotion and personality trait recognition using physiological features ✓ Recognition is achieved for both affective and personality dimensions
Drawbacks	□ Less Secure, High Embedding extraction time. □ Poor Application Performance □ Not based on real-time datasets
Description	This paper presents ASCERTAIN– a multimodal database for implicit personality and Affect recognition using commercial physiological sensors. To our knowledge, ASCERTAIN is the first database to connect personality traits and emotional states via physiological responses. ASCERTAIN contains big-five personality scales and emotional self-ratings of 58 users along with their Electroencephalogram (EEG), Electrocardiogram (ECG), Galvanic Skin Response (GSR) and facial activity data, recorded using off-the-shelf sensors while viewing affective movie clips. We first examine relationships between users’ affective ratings and personality scales in the context of prior observations, and then study linear and non-

linear physiological correlates of emotion and personality. Our analysis suggests that the emotion–personality relationship is better captured by non-linear rather than linear statistics. We finally attempt binary emotion and personality trait recognition using physiological features.

Each user performed the experiment in a session lasting about 90 minutes. Viewing of each movie clip is denoted as a trial. After two practice trials involving clips that were not part of the actual study, users watched movie clips randomly shown in two blocks of 18 trials, with a short break in-between to avoid fatigue. In each trial (Fig. 1(b)), a fixation cross was displayed for four seconds followed by clip presentation. After viewing each clip, users self-reported their emotional state in the form of affective ratings within a time limit of 30 seconds.

Galvanic Skin Response (GSR): GSR measures transpiration rate of the skin. When two electrodes are positioned on the middle and index finger phalanges and a small current is sent through the body, resistance to current flow changes with the skin transpiration rate. Most of the GSR information is contained in low-frequency components, and the signal is recorded at 100 Hz sampling frequency with a commercial bluetooth sensor.

Electroencephalography (EEG): EEG measures small changes in the skull's electrical field produced by neural activity, and information is encoded in the EEG signal amplitude as well as in certain frequency components. We used a commercial, single dry-electrode EEG sensor², which records eight information channels sampled at 32 Hz. The recorded information includes frontal lobe activity, level of facial activation, eye-blink rate and strength, which are relevant emotional responses.

Recognizing Emotions Induced by Affective Sounds through Heart Rate Variability

Literature Survey 4	
Title	Recognizing Emotions Induced by Affective Sounds through Heart Rate Variability
Authors	Mimma Nardelli, Gaetano Valenza, Member, IEEE, Alberto Greco, Antonio Lanata, Member, IEEE, and Enzo Pasquale Scilingo, Member, IEEE
Published Year	2015
Efficiency	✓ Automatic classification system for the recognition of the four classes of arousal and two classes of valence ✓ Estimated through standard and nonlinear analysis of Heart rate variability (HRV) exclusively ✓ Statistical significant differences between the elicited arousal or valence levels are considered
Drawbacks	□ Not Adaptable to Large Samples □ Tedious message updating □ Narrowly specialized knowledge
Description	Emotional states are modeled as a combination of arousal and valence dimensions according to the well-known circumplex model of affect, whereas the ANS dynamics is estimated through standard and nonlinear analysis of Heart rate variability (HRV) exclusively, which is derived from the electrocardiogram (ECG). In addition, Lagged Poincaré Plots of the HRV series were also taken into account. The affective sounds were gathered from the International Affective Digitized Sound System and grouped into four different levels of arousal (intensity) and two levels of valence (unpleasant and pleasant). A group of 27 healthy volunteers were administered with these standardized stimuli while ECG signals were

	continuously recorded. Then, those HRV features showing significant changes ($p < 0.05$ from statistical tests) between the arousal and valence dimensions were used as input of an automatic classification system for the recognition of the four classes of arousal and two classes of valence. The methodology of signal processing applied to HRV features can be divided into two sections: the extraction of standard and nonlinear features from the HRV series and the Leave One Subject Out Procedure. This procedure applies to the training set through feature selection (statistical analysis) and normalization The triangular interpolation of NN interval histogram, that is the baseline width of the distribution measured as a base of a triangle approximating the NN interval distribution (TINN) Between these parameters RMSSD indicates the short term variability, instead SDNN and HRV triangular index are features of the entire HRV. Statistical analysis and pattern recognition methodologies were used in this study to automatically recognize the autonomic response on the cardiovascular control of the subjects to the emotional sounds, and to associate the values of the HRV parameters with the kind of elicitation as expressed in terms of arousal and valence.
--	--

CHAPTER 3

System Analysis

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

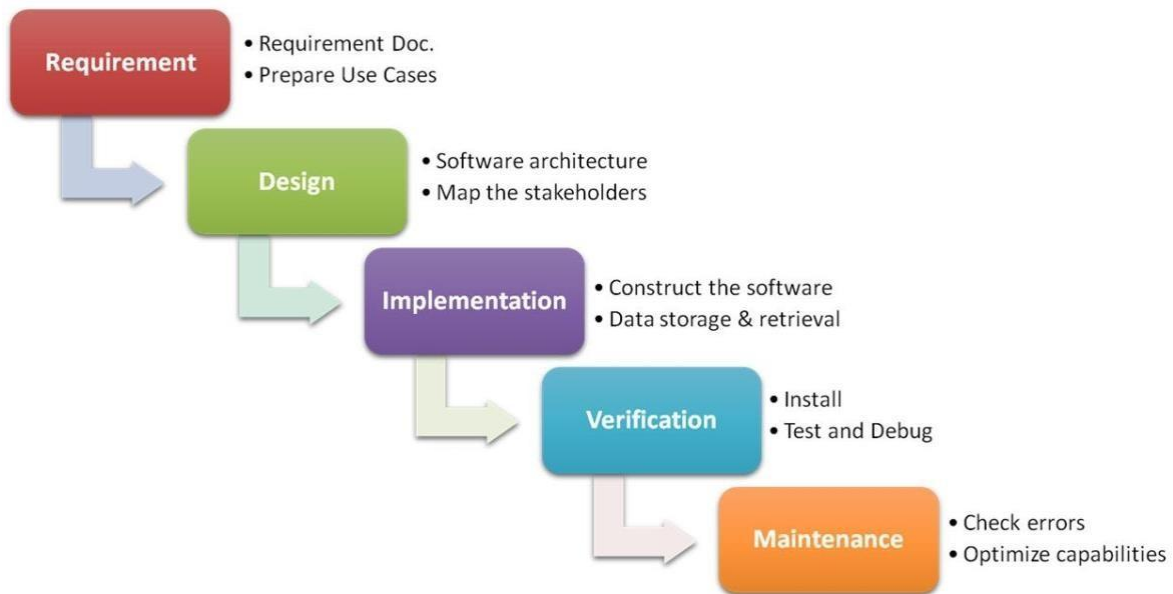
Waterfall Model

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

Requirement gathering and documentation

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that

has been allotted on your team.



(Reference : <https://www.lucidchart.com/blog/waterfall-project-management-methodology>)

2. System design

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

3. Implementation

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

4. Testing

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for reevaluation.

5. Delivery/deployment

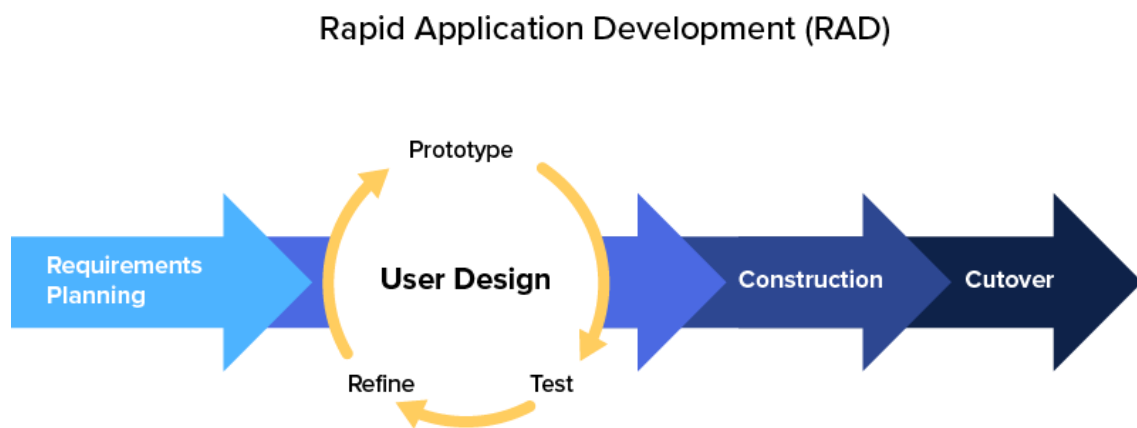
In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

6. Maintenance

The final solution has been implemented to the client and is being used. As troubles arise, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

RAD Model

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.



References (<https://www.lucidchart.com/blog/rapid-application-development-methodology>)

Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project

During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

Phase 2: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks. Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

- Preparation for rapid construction

- Program and application development

- Coding

- Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training. All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

- ▶ RAD lets you break the project down into smaller, more manageable tasks.
- ▶ The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and
- ▶ experience. Clients get a working product delivered in a shorter time frame.
- ▶ Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

Existing System

The existing system adopts a musical induction method to induce participants' real emotional states and collect their ECG signals without any deliberate laboratory setting. Afterward, we develop an automatic ECG-based emotion recognition algorithm to recognize human emotions elicited by listening to music. Physiological ECG features extracted from the time-, and frequency-domain, and nonlinear analyses of ECG signals are used to find emotion- relevant features and to correlate them with emotional states. Subsequently, we develop a sequential forward floating selection-kernel-based class separability-based (SFFS-KBCS-based) feature selection algorithm and utilize the generalized discriminant analysis

(GDA) to effectively select significant ECG features associated with emotions and to reduce the dimensions of the selected features, respectively. Positive/negative valence,

high/low arousal, and four types of emotions (joy, tension, sadness, and peacefulness) are recognized using least squares support vector machine (LS-SVM) recognizers.

Not enough time for some participants to reach a neutral state or emotional states elicited by the musical excerpts during the baseline and music listening stages.

The selected stimuli music does not have enough intensity to elicit emotions for the participants in the emotion classification tasks.

The difficulty of subject-independent classification is the intricate variety of nonemotional individual contexts among the participants, rather than an individual ECG specificity in emotion.

The participants cannot faithfully report their emotional state on the GEMS-9 questionnaire because of the unconcentrated condition during music listening or the misunderstanding of the meaning of the GEMS-9.

There is no one-to-one relation between emotion and physiological changes in ECG-based features: Feeling changes may occur without concomitant autonomic changes in ECG-based features and vice versa

Each participant signed a consent form and filled out a questionnaire before participating in this experiment. Next, the electrodes of the ECG recording instrument were placed on the participants and the recorded ECG signals should be checked by the experimenter to avoid interference resulting from incorrect placement of the electrodes or a destructive ECG recording instrument. Then, the participants were given instructions to comprehend the experimental protocol and the meanings of the scale used for self-assessment.

Drawbacks of Existing System

- Does not support multimodal data fusion
- Classification results are not combined linearly
- Sampling points intercepted from ECG data
- Emotional fatigue detection is not implemented

- The connection parameters between each layer of the feature extraction layer are not adjusted

Proposed System

Recognition of natural emotions from human faces is an interesting topic with a wide range of potential applications like human-computer interaction, automated tutoring systems, image and video retrieval, smart environments, and driver warning systems. Traditionally, facial emotion recognition systems have been evaluated on laboratory controlled data, which is not representative of the environment faced in real-world applications. To robustly recognize facial emotions in real-world natural situations, this paper proposes an approach called Extreme Sparse Learning (ESL), which has the ability to jointly learn a dictionary (set of basis) and a non-linear classification model. The proposed approach combines the discriminative power of Extreme Learning Machine (ELM) with the reconstruction property of sparse representation to enable accurate classification when presented with noisy signals and imperfect data recorded in natural settings. Additionally, this work presents a new local spatio-temporal descriptor that is distinctive and pose-invariant.

Then a bi-directional temporal RNN layer is further used to learn the discriminative features characterizing the temporal dependencies of the sequences, where sequences are produced from the spatial RNN layer. To further select those salient regions with more discriminative ability for emotion recognition, we impose sparse projection onto those hidden states of spatial and temporal domains to improve the model discriminant ability.

Consequently, the proposed two-layer RNN model provides an effective way to make use of both spatial and temporal dependencies of the input signals for emotion recognition.

Emotions can be described as discrete and consistent response to internal and external events which have a particular significance on human. Emotions are typically expressed using the face, the speech or body language. Reference [6] defines face emotions as a positive or negative experience that is associated with a particular pattern of physiological activity. However, they are brief in duration and

consist of coordinated set of responses which may include verbal, physiological, behavioral and neural mechanisms. Emotion is often a driving force behind positive

or negative motivation. The basic types of face emotion are Anger, Disgust, Fear, Happiness, Sadness, Contempt and Surprise. Other forms of emotion which can be expressed on the face includes shame, sensory pleasure, satisfaction, relief, pride in achievement, guilt, excitement, embarrassment, contentment, contempt, and amusement.

In the Deep Neural Network, the input data is consumed by the neurons in the first layer (not hidden) which then provide an output to the neurons within next layer and so on which provides the final output. The output might be a prediction like Yes or No. Each layer can have one or many neurons and each of them will compute a small function i.e. activation function. The activation function mimics the signal to pass to the next connected neurons. If the incoming neurons result in a value greater than a threshold, the output is passed else ignored. The connection between two neurons of successive layers would have an associated weight. The weight defines the influence of the input to the output for the next neuron and eventually for the overall final output. In a neural network, the initial weights would be all random but during the model training, these weights are updated iteratively to learn to predict a correct output. Decomposing the network, we can define few logical building blocks like a neuron, layer, weight, input, output, an activation function and finally a learning mechanism (optimizer) that will help the neural network incrementally update the weights (that were randomly initialized) to a more suitable weight that aids into correct prediction of the outcome.

Advantages of Proposed System

- ✓ Boosting in general decreases the bias error and builds strong predictive models.
- ✓ Stable and more robust model
- ✓ Can capture the linear as well as the non-linear relationships in the data.
- ✓ These networks can learn from examples and apply them when a similar event arises, making them able to work through real-time events.
- ✓ They can perform multiple tasks in parallel without affecting the system performance.

Requirement Specification

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

Hardware & Software Specification

Hardware Specification

Hardware Requirement

Hardware Minimum Requirement

Disk Space 32 GB or more, 10 GB or more for Foundation Edition

Processor 2.4 GHz 64 bit

Memory 8 GB

Display (1400 × 720) Capable video adapter and monitor

Software Specification

Backend Technologies

✓ Python

✓ Numpy

✓ Sci-lear

n

✓ Eclipse IDE

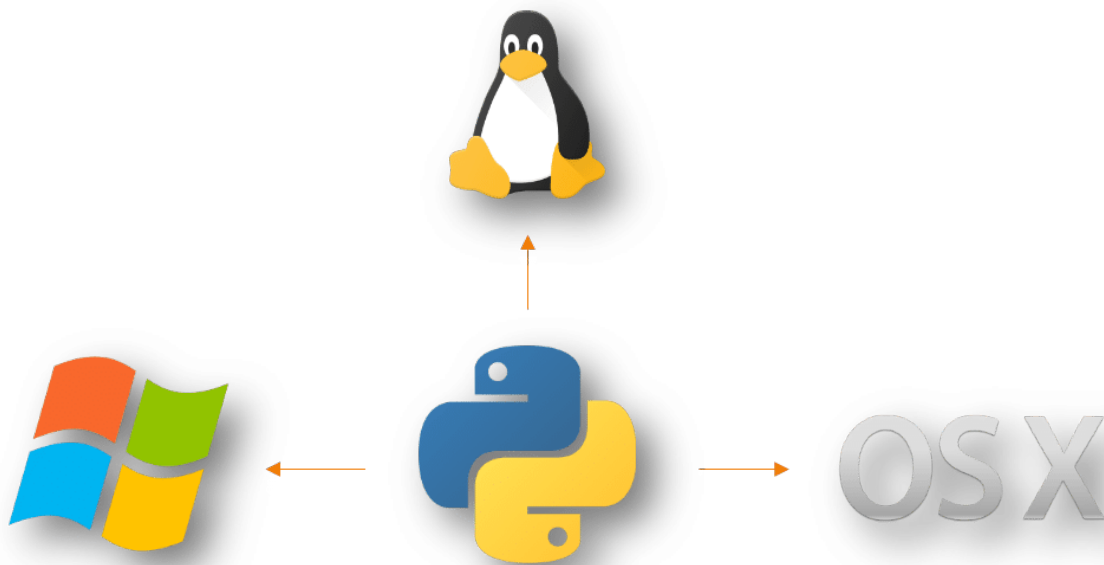
Frontend Technologies

✓ Web Technologies

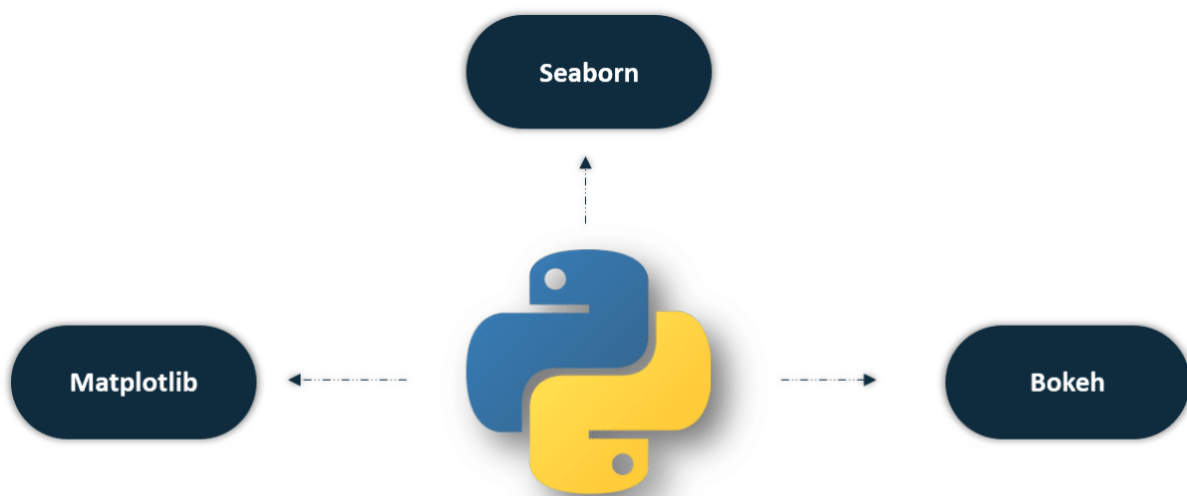
Software Description

About Python

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.



Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code. Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with `#` character. These comments end at the end of the physical line, which means that all characters starting after the `#` character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C,

Java, and more, there is no specific feature for multiline comments in Python. But that

does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability

- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth

About IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications. Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a

familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

- ▶ We can easily write our code in its text editor window.
- ▶ To store the resource files of a project, we have a project editor window.
It makes debugging of the code easier.
- ▶ An IDE's autosuggest feature helps us in writing efficient code.
- ▶ We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

✓ Komodo

✓

NetBeans

✓ Python for Visual Studio.NET

✓ PyDev

✓ PyStudio

✓ LiClipse (Eclipse)

✓ Jupyter Notebook

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger

- Error checking features
- An enhanced ability to analyze the code for quality and security concerns

- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application).

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

- ▶ Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.
- ▶ Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.
- ▶ GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.

- ▶ High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.
- ▶ Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.
- ▶ Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.
- ▶ Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.
- ▶ Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

System Design

Introduction

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

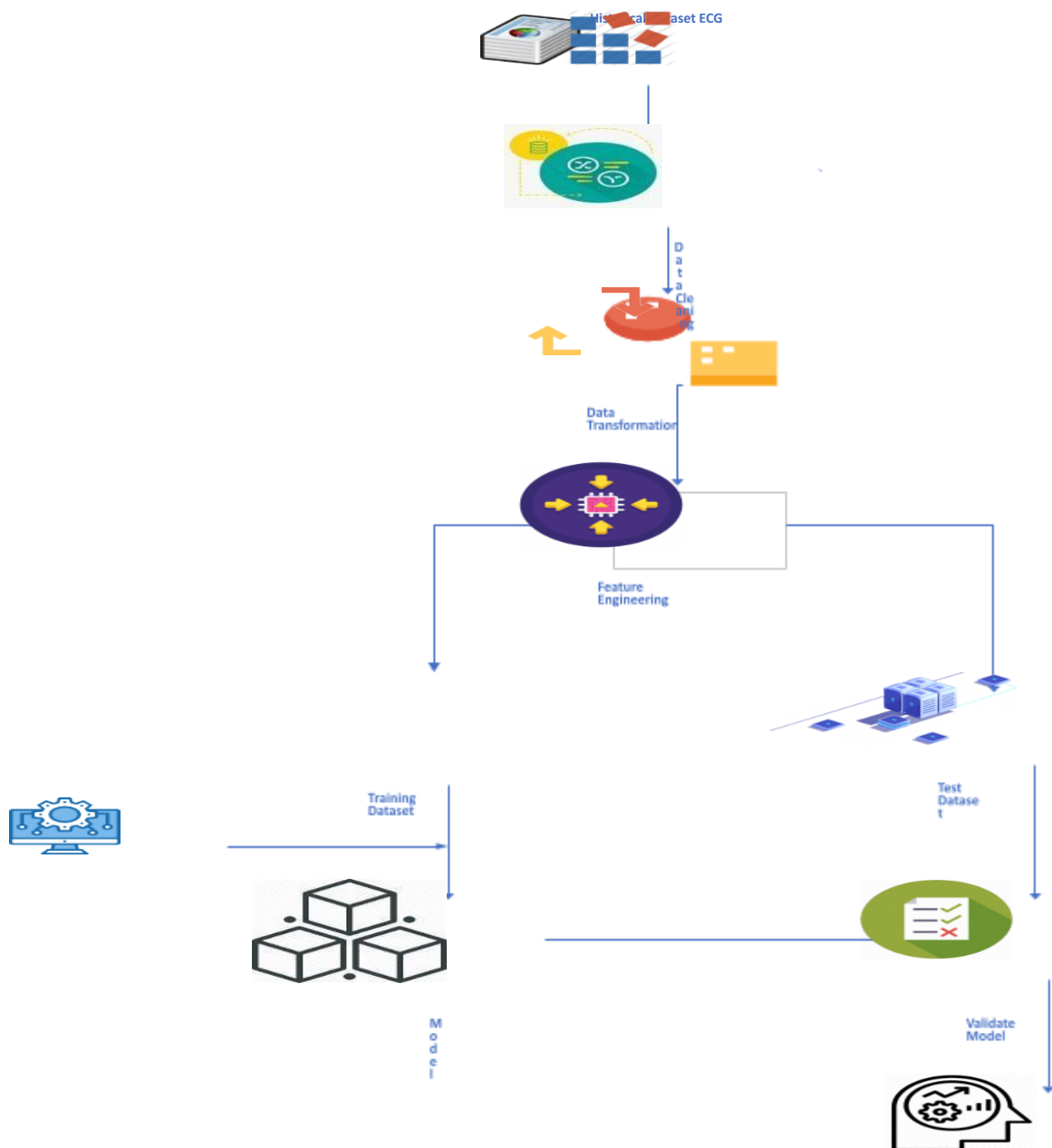
Architecture Diagram

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.



System Implementation

3.9.1 List of Modules

Module 1 : Preprocessing

Module 2 : Feature Selection

Module 3 : Model Creation and Evaluation

Module 1 : Preprocessing

Exploratory data analysis is generally cross-classified in two ways. First, each method is either non-graphical or graphical. And second, each method is either univariate or multivariate (usually just bivariate). Non-graphical methods generally involve calculation of summary statistics, while graphical methods obviously summarize the data in a diagrammatic or pictorial way. Univariate methods look at one variable (data column) at a time, while multivariate methods look at two or more variables at a time to explore relationships. Usually our multivariate EDA will be bivariate (looking at exactly two variables), but occasionally it will involve three or more variables. It is almost always a good idea to perform univariate EDA on each of the components of a multivariate EDA before performing the multivariate EDA.

A pre-processing technique used to reduce the effects of minor observation errors. The sample is divided into intervals and replaced by categorical values. Indicator variables: This technique converts categorical data into boolean values by creating indicator variables. If we have more than two values (n) we have to create $n-1$ columns. Centering & Scaling: We can centre the data of one feature by subtracting the mean to all values. To scale the data, we should divide the centered feature by the standard deviation.

Module 2 : Feature Selection

Scaling of Features is an essential step in modeling the algorithms with the datasets. The data that is usually used for the purpose of modeling is derived through various means. Standardization: StandardScaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation. Standardization can be helpful in

cases where the data follows a Gaussian distribution (or Normal distribution). However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a

bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high- dimensional data, it is computationally cheaper to use filter methods.

Module 3 : Prediction

The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. by default, the program ignores the original order of data.

It randomly picks data to form the training and test set, which is usually a desirable feature in real-world applications to avoid possible artifacts existing in the data preparation process. To disable this feature, simply set the shuffle parameter as False (default = True).

In the loss function, the first term calculates the mean negative logarithm value of the prediction probability of the training samples. The second and third terms ensure the sparse structure of the matrices in spatial and temporal projection. As elements of projection matrices indicate the importance of the corresponding spatial or temporal hidden states, the sparse structure is able to endow high weights to the discriminative hidden states while low weights to others, which achieves the purpose of selecting salient hidden states.

CHAPTER 4

SOFTWARE TESTING

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

Benefits of using Documentation

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks
- It provides feedback for your planning cycle.
- It creates objective evidence for the performance of the quality management system.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

the most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process. Let's take a look at the key requirements for these files and see how they contribute to the process.

Test strategy. An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

Test data. The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end-user in a ready solution.

Test plans. A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed planning. Such a document is distributed between team members and shared with all stakeholders.

Test scenarios. In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

Test cases. If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing — functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

Traceability Matrix. This software testing documentation maps test cases and their requirements. All entries have their custom IDs — team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.

External documentation collects information from inner documentation but also emphasize on providing a visual data representation — graphs, diagrams, etc.

External reports — these documents collect information on test results and can describe an entire project or a particular piece of functionality.

Test summary report — the file with final test results and findings, presented to stakeholders.

Bug reports — such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem.

The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

But as a curiosity or Out of the box thinking, have you ever wondered how developers test their own code? What method do they use to test before releasing

code for testing? How is dev-testing important in an agile process? The answer to all this is Unit Testing.

I want to educate you on the importance of Unit Testing so that development and testing teams can work more collaboratively to design, test and release an excellent application.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.



Unit Testing is used to design robust software components that help maintain code and eliminate the issues in code units. We all know the importance of finding and fixing defects in the early stage of the software development cycle. Unit Testing serves the same purpose.

Unit Testing is an integral part of the agile software development process. When a

nightly build run unit test suite should run and report should be generated. If any of the unit tests have failed then the QA team should not accept that build for verification.

If we set this as a standard process, many defects would be caught in the early development cycle, saving much testing time.

I know many developers hate to write unit tests. They either ignore or write bad unit test cases due to tight scheduled or lack of seriousness (yea they write empty unit tests, so 100% of them pass successfully ;-)). It's important to write good unit tests or don't write them at all. It's even more important to provide sufficient time and a supportive environment for real benefits.

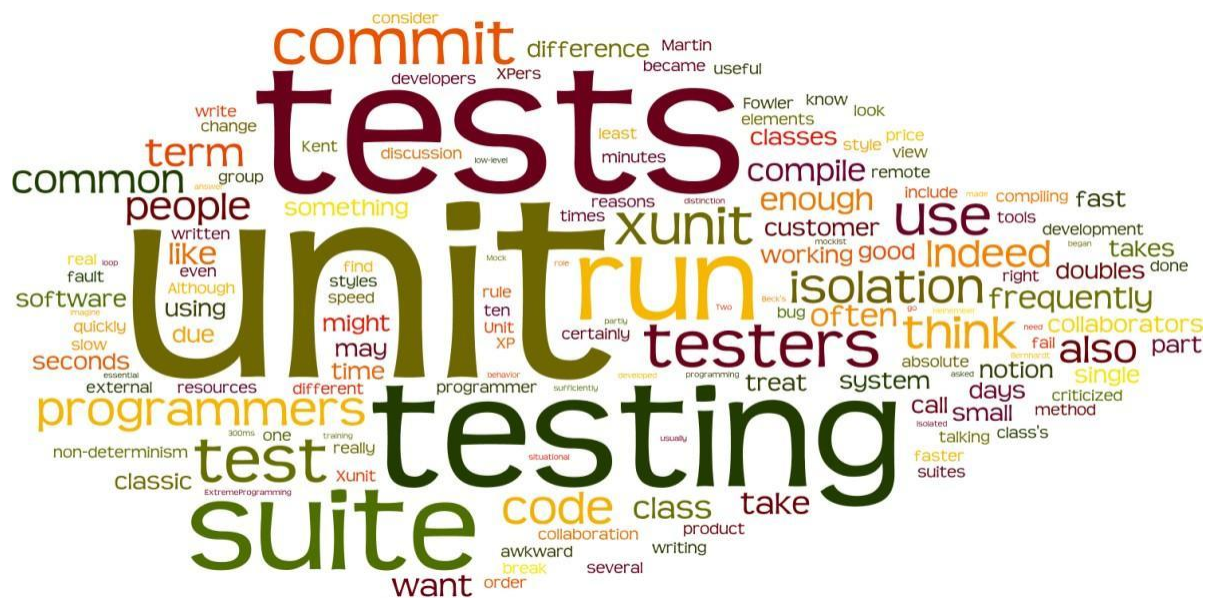
- Testing can be done in the early phases of the software development lifecycle when other modules may not be available for integration
- Fixing an issue in Unit Testing can fix many other issues occurring in later development and testing stages
- Cost of fixing a defect found in Unit Testing is very less than the one found in the system or acceptance testing
- Code coverage can be measured
- Fewer bugs in the System and Acceptance testing
- Code completeness can be demonstrated using unit tests. This is more useful in the agile process. Testers don't get the functional builds to test until integration is completed. Code completion cannot be justified by showing that you have written and checked in the code. But running Unit tests can demonstrate code completeness.
- Expect robust design and development as developers write test cases by understanding the specifications first.
- Easily identify who broke the build
- Saves development time: Code completion may take more time but due to decreased defect count overall development time can be saved.

Unit Testing frameworks are mostly used to help write unit tests quickly and easily. Most of the programming languages do not support unit testing with the inbuilt

compiler. Third-party open source and commercial tools can be used to make unit testing even more fun.

List of popular Unit Testing tools for different programming languages:

- ▶ Java framework – JUnit
- ▶ PHP framework – PHPUnit
- ▶ C++ frameworks – UnitTest++ and Google C++
- ▶ .NET framework – NUnit
- ▶ Python framework – py.test



Functional Testing is a type of black box testing whereby each part of the system is tested against functional specification/requirements. For instance, seek answers to the following questions,

Are you able to login to a system after entering correct credentials?

Does your payment gateway prompt an error message when you enter incorrect card number?

Does your “add a customer” screen adds a customer to your records successfully?

Well, the above questions are mere samples to perform full-fledged functional testing of a system.

Test Driven Development

Test Driven Development, or TDD, is a code design technique where the programmer writes a test before any production code, and then writes the code that will make that test pass. The idea is that with a tiny bit of assurance from that initial test, the programmer can feel free to refactor and refactor some more to get the cleanest code they know how to write. The idea is simple, but like most simple things, the execution is hard. TDD requires a completely different mind set from what most people are used to and the tenacity to deal with a learning curve that may slow you down at first.

Functional Testing types include:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Sanity Testing
5. Smoke Testing
6. Interface Testing
7. Regression Testing

Performance Testing

- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing

- Compliance Testing
- Localization Testing

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application.

Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.



Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The

system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”. The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load. Load testing comes under the Non Functional Testing & it is designed to test the non-functional requirements of a software application.

Load testing is perform to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero load.

CHAPTER 5

RESULTS

```
Desktop/Emotions-based-ECG x Emotions-based-ECG-ML-NN x +
localhost:8888/notebooks/Desktop/Emotions-based-ECG-ML-NN/Emotions-based-ECG-ML-NN.ipynb
jupyter Emotions-based-ECG-ML-NN Last Checkpoint: Last Tuesday at 17:39 (autosaved)
Python 3

In [105]: score = model2.evaluate(X_test_n, y_test_n, verbose=1)
7/7 [=====] - 0s 4ms/step - loss: 0.9915 - accuracy: 0.8125

In [106]: print(f'Loss: {score[0]}\nAccuracy: { round(score[1], 2)*100}%')
y_pred = np.argmax(model2.predict(X_test_n), axis = 1)
Loss: 0.9915047883987427
Accuracy: 81.0%

In [107]: cm = metrics.confusion_matrix(y_test_n, y_pred)
f1_score1 = round(f1_score(y_test_n, y_pred, average='micro'), 3)

In [108]: print(f'F1 Score: {f1_score1*100}%\n')
print(f'Confusion Matrix:\n{cm}')

F1 Score: 81.2%

Confusion Matrix:
[[10  0  0  2  0  0  0  1  0  0  0  1  0  1  2  1  0  0  0]
 [ 0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  0  6  0  0  0  0  0  1  1  2  0  0  0  0  0  4  0  0]
 [ 0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 13  0  1  0  0  0  0  0  0  1  0  0  0]
 [ 0  0  0  1  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  1  0  0  0  0  8  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  1 11 12  1  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  2 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0 14  0  0  0  0  0  0]
 [ 4  0  0  1  0  1  0  0  0  0  0  0  1  5  0  1  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6]]
```

```
Desktop/Emotions-based-ECG x Emotions-based-ECG-ML-NN x +
localhost:8888/notebooks/Desktop/Emotions-based-ECG-ML-NN/Emotions-based-ECG-ML-NN.ipynb
jupyter Emotions-based-ECG-ML-NN Last Checkpoint: Last Tuesday at 17:39 (autosaved)
Python 3

In [90]: print(f'Accuracy: {accuracy*100}%\n')
print(f'F1 Score: {f1*100}%\n')
print(f'Confusion Matrix:\n{cm}')

Accuracy: 80.0%

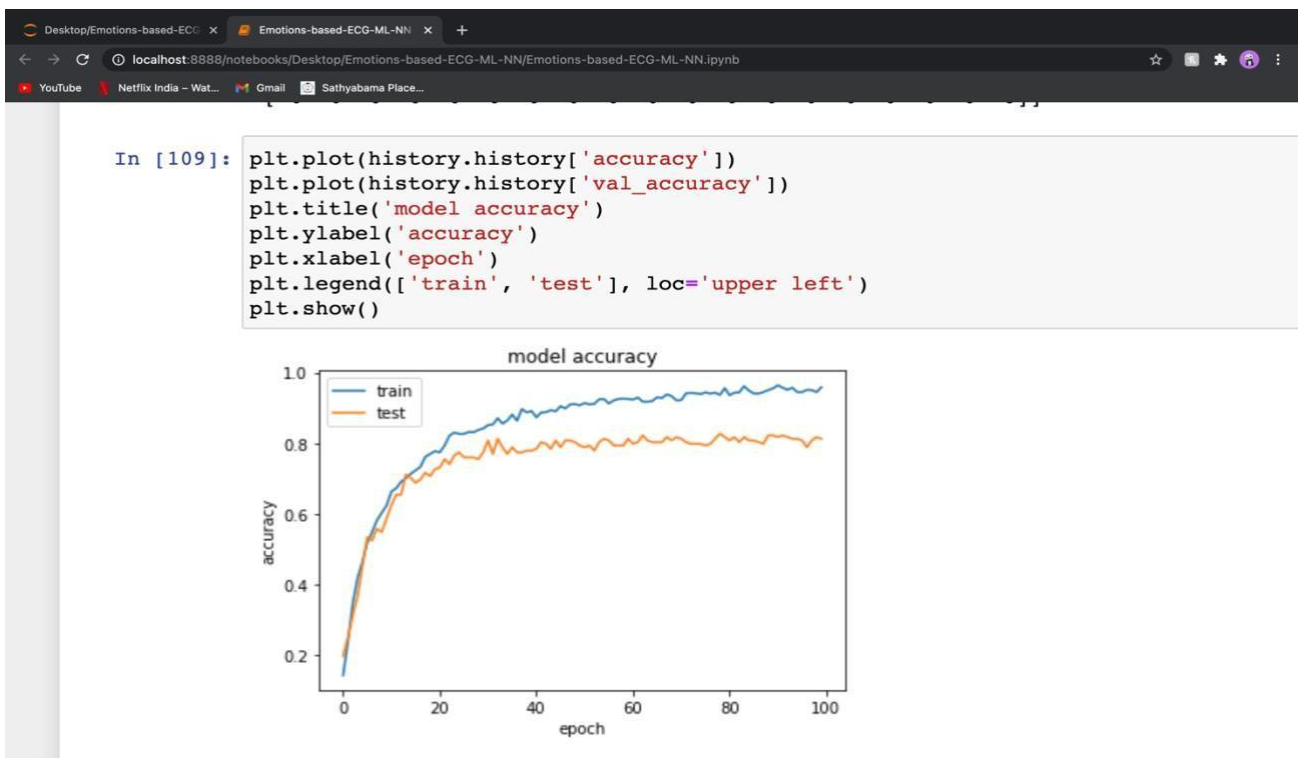
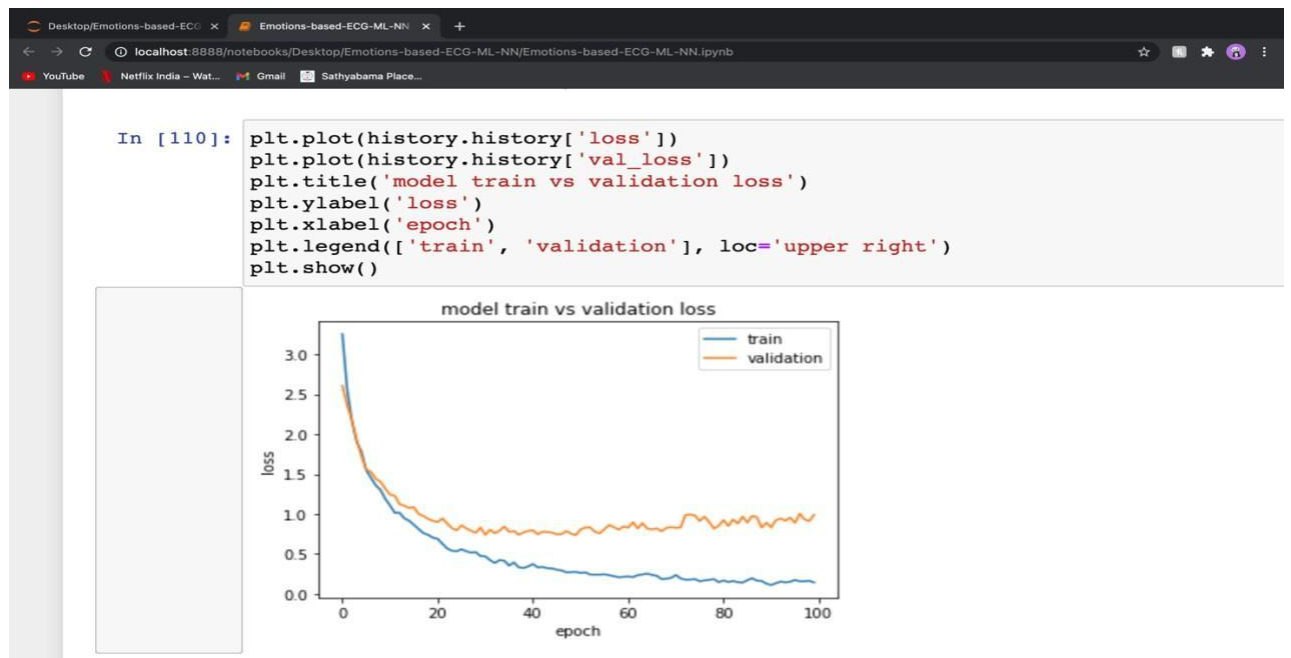
F1 Score: 80.0%

Confusion Matrix:
[[ 9  0  0  4  0  0  0  0  0  0  0  1  0  0  4  0  0  0  0]
 [ 0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  6  0  0  0  1  0  0  3  0  0  0  2  3  0  0  0]
 [ 0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0 12  0  1  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  1  0  0  0  0  7  0  0  0  0  1  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  1 13  0  1  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  3  0 11  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 13  1  1  0  0  0]
 [ 5  0  0  0  0  0  0  0  0  0  0  0  0  0  6  1  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  9  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6]]

In [91]: classifier = OneVsRestClassifier(xgb.XGBClassifier(n_jobs = -1))

In [92]: classifier.fit(X_train, y_train)

Out[92]: OneVsRestClassifier(estimator=XGBClassifier(base_score=None, booster=None,
colsample_bylevel=None,
colsample_bynode=None,
colsample_bytree=None, gamma=None,
gpu_id=None, importance_type='gain',
```



CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION

In this paper, we proposed a novel classification scheme, which is motivated by the recent advancements in the field of sparse representation and supervised dictionary learning. This system incorporates reconstruction properties of sparse representation and discriminative power of a Deep Neural Network for robust classification. In addition, we proposed a novel temporal descriptor for pose invariant facial emotion detection. We have performed extensive experiments on both acted and spontaneous emotion databases to evaluate the effectiveness of the proposed feature extraction and classification schemes under different scenarios.

FUTURE ENHANCEMENT

There are some improvements for the next steps. First, more subjects will be recruited to analyze the more parameters of ECG, including subjects of different ages and different occupations to establish emotional signal database. Next, more parameters will be considered to improve the algorithm of emotion recognition.

REFERENCES

- ▶ Ceballos R F, Largo F F. On The Estimation of the Hurst Exponent Using Adjusted Rescaled Range Analysis, Detrended Fluctuation Analysis and Variance Time Plot: A Case of Exponential Distribution[J]. "Imperial Journal of Interdisciplinary Research (IJIR), 2017, 3(8):424-434.
- ▶ Honghao Gao, Wanqiu Huang, Xiaoxian Yang, Yucong Duan, Yuyu Yin. Towards Service Selection for Workflow Reconfiguration: An Interface-Based Computing. Future Generation Computer Systems(FGCS). 2018,87:298- 311.
- ▶ Kaijian Xia, Hongsheng Yin, Yu-Dong Zhang, Deep Semantic Segmentation of Kidney and Space-Occupying Lesion Area Based on SCNN and ResNet Models Combined with SIFT-Flow Algorithm. J. Medical Systems, 2019, 43(1): 2:1-2:12.
- ▶ H. Becker, J. Fleureau, P. Guillotel, F. Wendling, I. Merlet, and L. Albera, "Emotion recognition based on high-resolution eeg recordings and reconstructed brain sources," IEEE Transactions on Affective Computing, 2017.
- ▶ W.-L. Zheng and B.-L. Lu, "Investigating critical frequency bands and channels for eeg- based emotion recognition with deep neural networks," IEEE Transactions on Autonomous Mental Development, vol. 7, no. 3, pp. 162– 175, 2015.
- ▶ Zhao L, Huang H, Li X, et al. An Accurate and Robust Approach of Device-Free Localization With Convolutional Autoencoder[J]. IEEE Internet of Things Journal, 2019, 6(3):5825-5840.
- ▶ Ping Z, Gong Z, Li S, et al. Learning to Diversify Deep Belief Networks for Hyperspectral Image Classification[J]. IEEE Transactions on Geoscience & Remote Sensing, 2017, 55(6):3516-3530.
- ▶ T. Zhang, W. Zheng, Z. Cui, Y. Zong, and Y. Li, "Spatial-temporal recurrent neural network for emotion recognition," IEEE Transactions on Cybernetics, vol. PP, no. 99, pp. 1–9, 2017.

- ▶ . Song, W. Zheng, P. Song, and Z. Cui, "Eeg emotion recognition using dynamical graph convolutional neural networks," *IEEE Transactions on Affective Computing*, 2018.
- ▶ Honghao Gao, Wanqiu Huang, Xiaoxian Yang. Applying Probabilistic Model Checking to Path Planning in an Intelligent Transportation System Using Mobility Trajectories and Their Statistical Data. *Intelligent Automation And Soft Computing(Autosoft)*, 2019, 25(3):547–559.
- ▶ Razavi S M, Taghipour-Gorjikolaie M, Mehrshad N. Multimodal biometric identification system based on finger-veins using hybrid rank–decision-level fusion technique[J]. *IEEE Transactions on Electrical & Electronic Engineering*, 2017, 12(5):728-735.
- ▶ S. Katsigiannis and N. Ramzan, "Dreamer: A database for emotion recognition through eeg and ecg signals from wireless low-cost off-the-shelf devices," *IEEE journal of biomedical and health informatics*, vol. 22, no. 1, pp. 98–107, 2018.
- ▶ Y. Li, W. Zheng, Z. Cui, Y. Zong, and S. Ge, "Eeg emotion recognition based on graph regularized sparse linear regression," *Neural Processing Letters*, pp. 1–17, 2018.
- ▶ P. Song and W. Zheng, "Feature selection based transfer subspace learning for speech emotion recognition," *IEEE Transactions on Affective Computing*, 2018.
- ▶ W. Zheng, "Multichannel eeg-based emotion recognition via group sparse canonical correlation analysis," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 3, pp. 281–290, 2017.

APPENDIX

A.SAMPLE CODE

```
%load_ext tensorboard
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import re
from scipy.io.arff import loadarff
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import model_selection, svm
from sklearn.svm import SVR
from sklearn.metrics import auc, accuracy_score, confusion_matrix, mean_squared_
error
from sklearn.model_selection import cross_val_score, cross_val_predict, GridSearch
CV, StratifiedKFold, KFold, RandomizedSearchCV, train_test_split
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
import xgboost as xgb
import tensorflow as tf
for device in gpu_devices:
    tf.config.experimental.set_memory_growth(device, True)
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras import backend as K
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalMaxPool1D
from tensorflow.keras.layers import Conv1D, MaxPooling1D
from imblearn.over_sampling import RandomOverSampler
```

Using TensorFlow backend.

```
from sklearn.naive_bayes import GaussianNB
from skmultilearn.ensemble import RakeID
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
from sklearn.utils import class_weight
from skmultilearn.model_selection import iterative_train_test_split
from sklearn.metrics import multilabel_confusion_matrix, ConfusionMatrixDisplay, f1_score
from tensorflow.keras.layers import Conv1D, MaxPooling1D, BatchNormalization
from sklearn import metrics

raw_data = loadarff('emotions.arff')
df = pd.DataFrame(raw_data[0]).astype(float)
df.head()
df.info()
df.describe()

labels = ['amazed-suprised', 'happy-pleased', 'relaxing-calm', 'quiet-still', 'sad-lonely', 'angry-aggressive']
feats_df = df.drop(columns = labels)
labels_df = df[labels]
label_d = {'amazed-suprised' : 0, 'happy-pleased': 0, 'relaxing-calm' : 0, 'quiet-still' : 0, 'sad-lonely': 0, 'angry-aggressive' : 0}
for col in labels_df.columns:
    label_d[col] += len(labels_df[labels_df[col] == 1])

plt.figure(figsize = (16,8))
plt.bar(range(len(label_d)), list(label_d.values()), align='center')
plt.xticks(range(len(label_d)), list(label_d.keys()))
plt.show()

def transform_multiclass(orig_df):
    df = orig_df.copy()
    classe = []
    for i in range(len(df)):
        classe.append("")
```

```
for i in range(len(df)):
```

```

if df['amazed-surprised'][i] == 1:
    classe[i] = classe[i] + 'surprised-'
if df['happy-pleased'][i] == 1:
    classe[i] = classe[i] + 'happy-'
if df['relaxing-calm'][i] == 1:
    classe[i] = classe[i] + 'relaxing-'
if df['quiet-still'][i] == 1:
    classe[i] = classe[i] + 'still-'
if df['sad-lonely'][i] == 1:
    classe[i] = classe[i] + 'lonely-'
if df['angry-aggressive'][i] == 1:
    classe[i] = classe[i] + 'angry-'

df['Class'] = classe
df.drop(['angry-aggressive','amazed-surprised','happy-pleased','relaxing-calm','quiet-
still','sad-lonely'],axis=1,inplace = True)
return df
class_label_df = transform_multiclass(labels_df)
class_label_df['Class'].unique()
plt.figure(figsize = (16,8))
sns.countplot(y = class_label_df.Class)
plt.show()
class_label_df.Class.value_counts()
df.corr().style.background_gradient(cmap='coolwarm')
df.duplicated().unique()
df.isnull().sum().unique()
corr_matrix = feats_df.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
to_drop = [column for column in upper.columns if any(upper[column] > 0.8)]
df.drop(df[to_drop], axis=1,inplace = True)
df['Class'] = class_label_df['Class']
df.head()
class_df = df.drop(labels,1)

```

```
print(class_df.shape)
```

```

class_df.head()
classes = class_label_df.Class.value_counts().index
class_series = class_label_df['Class'].value_counts()
outliers = []
for label in classes:
    if class_series[label] < 5:
        outliers.append(label)
outliers
for outlier in outliers:
    class_df = class_df[class_df['Class'] != outlier]

print(class_df.shape)
(577, 54)
X = class_df.drop('Class',1)
y = class_df['Class']
print(X.shape, y.shape)
(577, 53) (577,)
ros = RandomOverSampler()
X_over, y_over = ros.fit_resample(X, y)
print(X_over.shape)
print(y_over.shape)
(1539, 53)
(1539,)
y_over.value_counts()
X_train, X_test, y_train, y_test = train_test_split(X_over,y_over,test_size = 0.2, random_state = 42)
train_df = X_train.join(y_train)
test_df = X_test.join(y_test)
test_df.shape
test_df.duplicated().sum()
test_df.drop_duplicates(inplace = True)
test_df.shape
test_df.shape

```

```

le = LabelEncoder()
X_train = train_df.drop('Class',1).values
X_train = preprocessing.scale(X_train)
y_train = le.fit_transform(train_df['Class'])
X_test = test_df.drop('Class',1).values
X_test = preprocessing.scale(X_test)
y_test = le.fit_transform(test_df['Class'])
X_train
classifier = RandomForestClassifier(n_estimators = 100,n_jobs = -1, verbose = 0, random_state = 30)
classifier.fit(X_train, y_train)
RandomForestClassifier(n_jobs=-1, random_state=30)
y_pred = classifier.predict(X_test)
cm = metrics.confusion_matrix(y_test, y_pred)
f1 = round(f1_score(y_test,y_pred, average = 'micro'),2)
accuracy = round(accuracy_score(y_test,y_pred),2)
print(f'Accuracy: {accuracy*100}%\n')
print(f'F1 Score: {f1*100}%\n')
print(f'Confusion Matrix:\n{cm}')
classifier = OneVsRestClassifier(xgb.XGBClassifier(n_jobs = -1))
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
cm = metrics.confusion_matrix(y_test, y_pred)
f1 = round(f1_score(y_test,y_pred, average = 'micro'),3)
accuracy = round(accuracy_score(y_test,y_pred),3)
print(f'Accuracy: {accuracy*100}%\n')
print(f'F1 Score: {f1*100}%\n')
print(f'Confusion Matrix:\n{cm}')
BATCH_SIZE = 64
EPOCHS = 100
dense_layers = [1]
conv_layers = [2]
layer_sizes = [128]

```

`X_train_n = X_train`

```

y_train_n = y_train
X_test_n = X_test
y_test_n = y_test
X_train_n = preprocessing.scale(X_train_n)
X_test_n = preprocessing.scale(X_test_n)
X_train_n = np.reshape(X_train_n, (X_train_n.shape[0], X_train_n.shape[1], 1))
X_test_n = np.reshape(X_test_n, (X_test_n.shape[0], X_test_n.shape[1], 1))
sw = class_weight.compute_sample_weight('balanced', y_train_n)
for dense_layer in dense_layers:
    for layer_size in layer_sizes:
        for conv_layer in conv_layers:
            NAME = f'emotions_cnn-dense-{dense_layer}-layer{layer_size}-conv{conv_la
            yer}-time-{int(time.time())}'
            tb_callback = TensorBoard(log_dir = f'logs/{NAME}')
            model2 = Sequential()
            if conv_layer > 0:
                model2.add(Conv1D(layer_size, 3,padding='valid', input_shape=(53,1),acti
                vation='tanh'))
                model2.add(BatchNormalization())
                model2.add(MaxPooling1D(pool_size=(4)))
                model2.add(Dropout(0.2))
            for l in range(conv_layer - 1):
                model2.add(Conv1D(layer_size, 3,padding='valid',activation='tanh'))
                model2.add(BatchNormalization())
                model2.add(MaxPooling1D(pool_size=(2)))
                model2.add(Dropout(0.2))
            else:
                model2.add(Flatten())
                model2.add(Dense(layer_size, input_shape = (53,) ,activation='tanh'))

            model2.add(Dropout(0.2))
            model2.add(Flatten())
            for l in range(dense_layer):

```

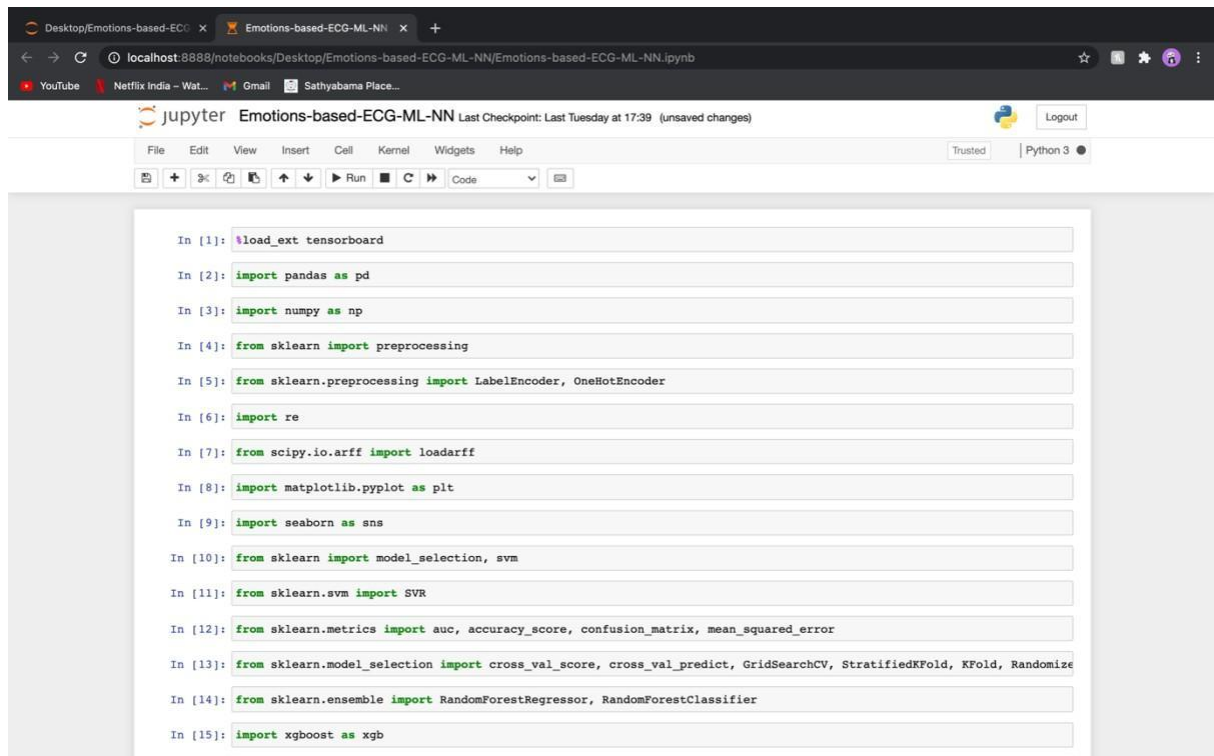
```
model2.add(Dense(layer_size, activation='relu'))
```

```

        model2.add(BatchNormalization())
        model2.add(Dropout(0.1))
        model2.add(Dropout(0.3))
        model2.add(Dense(20, activation="softmax"))
model2.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=[
'accuracy'])
history = model2.fit(X_train_n, y_train_n,
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,
                    verbose= 0,
                    validation_data=(X_test_n, y_test_n),
                    )
score = model2.evaluate(X_test_n, y_test_n, verbose=1)
print(f'Loss: {score[0]}\nAccuracy: { round(score[1], 2)*100}%')
y_pred = np.argmax(model2.predict(X_test_n), axis = 1)
cm = metrics.confusion_matrix(y_test_n, y_pred)
f1_score1 = round(f1_score(y_test_n,y_pred,average='micro'),3)
print(f'F1 Score: {f1_score1*100}%\n')
print(f'Confusion Matrix:\n{cm}')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model train vs validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
plt.show()

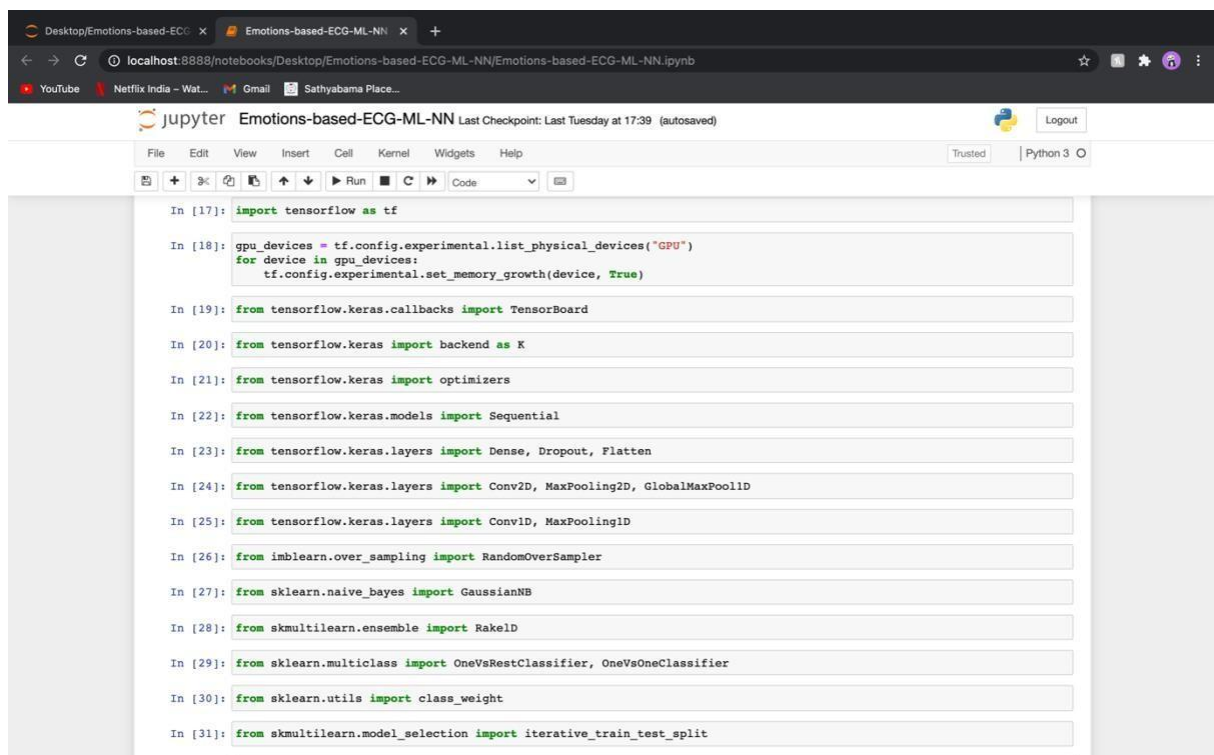
```

B.SCREENSHOTS



This screenshot shows the first 15 input cells of a Jupyter Notebook titled "Emotions-based-ECG-ML-NN". The notebook is running on a local host at 8888. The imports include:

```
In [1]: %load_ext tensorboard
In [2]: import pandas as pd
In [3]: import numpy as np
In [4]: from sklearn import preprocessing
In [5]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
In [6]: import re
In [7]: from scipy.io.arff import loadarff
In [8]: import matplotlib.pyplot as plt
In [9]: import seaborn as sns
In [10]: from sklearn import model_selection, svm
In [11]: from sklearn.svm import SVR
In [12]: from sklearn.metrics import auc, accuracy_score, confusion_matrix, mean_squared_error
In [13]: from sklearn.model_selection import cross_val_score, cross_val_predict, GridSearchCV, StratifiedKFold, KFold, Randomize
In [14]: from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
In [15]: import xgboost as xgb
```



This screenshot shows the next 16 input cells of the Jupyter Notebook. The imports include:

```
In [17]: import tensorflow as tf
In [18]: gpu_devices = tf.config.experimental.list_physical_devices("GPU")
for device in gpu_devices:
    tf.config.experimental.set_memory_growth(device, True)
In [19]: from tensorflow.keras.callbacks import TensorBoard
In [20]: from tensorflow.keras import backend as K
In [21]: from tensorflow.keras import optimizers
In [22]: from tensorflow.keras.models import Sequential
In [23]: from tensorflow.keras.layers import Dense, Dropout, Flatten
In [24]: from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalMaxPool1D
In [25]: from tensorflow.keras.layers import Conv1D, MaxPooling1D
In [26]: from imblearn.over_sampling import RandomOverSampler
In [27]: from sklearn.naive_bayes import GaussianNB
In [28]: from skmultilearn.ensemble import RakelD
In [29]: from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
In [30]: from sklearn.utils import class_weight
In [31]: from skmultilearn.model_selection import iterative_train_test_split
```

Desktop/Emotions-based-ECG x Emotions-based-ECG-ML-NN x +

localhost:8888/notebooks/Desktop/Emotions-based-ECG-ML-NN/Emotions-based-ECG-ML-NN.ipynb

jupyter Emotions-based-ECG-ML-NN Last Checkpoint: Last Tuesday at 17:39 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

In [30]: from sklearn.utils import class_weight

In [31]: from sklearn.model_selection import iterative_train_test_split

In [32]: from sklearn.metrics import multilabel_confusion_matrix, ConfusionMatrixDisplay, f1_score

In [33]: from tensorflow.keras.layers import Conv1D, MaxPooling1D, BatchNormalization

In [34]: from sklearn import metrics

In [35]: raw_data = loadarff('emotions.arff')

In [36]: df = pd.DataFrame(raw_data[0]).astype(float)

In [37]: df.head()

```

Out[37]:

	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	Mean_Acc1298_Mean_Mem40_MFCC_0	Mean_A
0	0.034741	0.089665	0.091225	-73.302422	
1	0.081374	0.272747	0.085733	-62.584437	
2	0.110545	0.273567	0.084410	-65.235325	
3	0.042481	0.199281	0.093447	-80.305152	
4	0.074550	0.140880	0.079789	-93.697749	

5 rows x 78 columns

```

In [38]: df.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 593 entries, 0 to 592
Data columns (total 78 columns):
Column Non-Null Count Dtype

Desktop/Emotions-based-ECG x Emotions-based-ECG-ML-NN x +

localhost:8888/notebooks/Desktop/Emotions-based-ECG-ML-NN/Emotions-based-ECG-ML-NN.ipynb

jupyter Emotions-based-ECG-ML-NN Last Checkpoint: Last Tuesday at 17:39 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

In [43]: label_d = {'amazed-surprised': 0, 'happy-pleased': 0, 'relaxing-calm': 0, 'quiet-still': 0, 'sad-lonely': 0, 'angry-aggressive': 0}

In [44]: for col in labels_df.columns:
    label_d[col] += len(labels_df[labels_df[col] == 1])

In [45]: plt.figure(figsize = (16,8))
plt.bar(range(len(label_d)), list(label_d.values()), align='center')
plt.xticks(range(len(label_d)), list(label_d.keys()))
plt.show()

```

Emotion	Count
amazed-surprised	175
happy-pleased	165
relaxing-calm	265
quiet-still	150
sad-lonely	170
angry-aggressive	190

