HUMAN STRESS DETECTION IN AND THROUGH SLEEP BY USING AI

A PROJECT REPORT

Submitted by

SURENDHAR A	**211419104275**
THARUN A	**211219104290**
RAJARAJAN P	**211419104210**

in partial fulfilment for the awardof the

degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"HUMAN STRESS DETECTION IN AND THROUGH SLEEP USING AI"** is the bonafide work of "**SURENDHAR A (211419104275), THARUN A (211419104290), RAJARAJAN P (211419104210)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA M.E.,Ph.D.,	**Dr.S.BALAJI B.Tech,M.E.,Ph.D**
HEAD OF THE DEPARTMENT,	**ASSOCIATE PROFESSOR**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE	PANIMALAR ENGINEERING COLLEGE
NAZARATHPETTAI,	NAZARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above mentioned students were examined in the End Semester project viva-voice held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **SURENDHAR A (211419104275), THARUN A (211419104290), RAJARAJAN P (211419104210)** hereby declare that this project report titled **"HUMAN STRESS DETECTION IN AND THROUGH SLEEP USING AI",** under the guidance of **Dr. S. BALAJI, B.Tech ,M.E., Ph.D.**, is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

SURENDHAR A

THARUN A

RAJARAJAN P

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR, M.E., Ph.D**. and **Dr.SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities toundertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr.L.JABASHEELA., M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my project coordinator **Mr.M.MOHAN,M.E.,(Ph.D.)** and **Project Guide Dr.S.BALAJI B.Tech,M.E Ph.D,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

SURENDHAR A

THARUN A

RAJARAJAN P

ABSTRACT

Stress plays an integral role in influencing one's decision-making capability, attention span, learning, and problem-solving capacity. Stress detection and modeling have been active areas of research in the fields of psychology and computer science in recent times. Psychologists quantify stress using affective states, which is the experience of feeling the underlying emotional state. Most of the work in classifying human stress was achieved using user-dependent models, incapable of generalizing to a new user. This causes new user to spend a significant amount of their time in training the model to predict their affective states. There is an urgent need to treat basic mental health problems that prevail among children which may lead to complicated problems, if not treated at an early stage. Machine learning Techniques are currently well suited for analyzing medical data and diagnosing the problem. The attributes have been reduced by analysing Features over the full attribute data set. The accuracy over the selected attribute set on various machine learning algorithms have been compared.

TABLE OF CONTENTS

LIST OF FIGURES

1.1 INTRODUCTION

Stress plays an integral role in influencing one's decision-making capability, attention span, learning, and problem-solving capacity. Stress detection and modeling have been active areas of research in the fields of psychology and computer science in recent times. Psychologists quantify stress using affective states, which is the experience of feeling the underlying emotional state. Most of the work in classifying human stress was achieved using user-dependent models, incapable of generalizing to a new user. This causes new user to spend a significant amount of their time in training the model to predict their affective states. There is an urgent need to treat basic mental health problems that prevail among children which may lead to complicated problems, if not treated at an early stage. Machine learning Techniques are currently well suited for analyzing medical data and diagnosing the problem. The attributes have been reduced by analysing Features over the full attribute data set. The accuracy over the selected attribute set on various machine learning algorithms have been compared.

1.1.1 OBJECTIVES

The goal is to develop a machine learning model for stress detection Prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

1.2 Existing System:

A novel JCCB-FSC method to detect stress in decision-making. In particular, we used TSST and BART experimental paradigms to simulate the decision-making process of medical students under stress. Different from previous research that focused on simply grading stress in no specific situation, we aimed to detect stress in the decision-making process, and provide clues for stress detection in the decision-making situation of medical staff in the future. Stress is one of the contributing factors affecting decision-making. Therefore, early stress recognition is essential to improve clinicians' decision-making performance. Functional near-infrared spectroscopy (fNIRS) has shown great potential in detecting stress. However, the majority of previous studies only used fNIRS features at the individual level for classification without considering the correlations among channels corresponding to the brain, which may provide distinguishing features.

1.2.1 Disadvantages:

- This is a complex process.
- Performance metrics are not calculated.
- Deployment is not implemented.

2. LITERATURE REVIEW

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them

2.1 Review of Literature Survey

Title : Machine Learning-based Approach for Depression Detection in Twitter Using Content and Activity Features

Author: Hatoon AlSagri, Mourad Ykhlef

Year : 2020

Social media channels, such as Facebook, Twitter, and Instagram, have altered our world forever. People are now increasingly connected than ever and reveal a sort of digital persona. Although social media certainly has several remarkable features, the demerits are undeniable as well. Recent studies have indicated a correlation between high usage of social media sites and increased depression. The present study aims to exploit machine learning techniques for detecting a probable depressed Twitter user based on both, his/her network behavior and tweets. For this purpose, we trained and tested classifiers to distinguish whether a user is depressed or not using features extracted from his/her activities in the network and tweets. The results showed that the more features are used, the higher are the accuracy and F-measure scores in detecting depressed

users. This method is a data-driven, predictive approach for early detection of depression or other mental illnesses. This study's main contribution is the exploration part of the features and its impact on detecting the depression level..

Title : Predicting the Utilization of Mental Health Treatment with Various Machine Learning Algorithms

Author: MEERA SHARMA, SONOK MAHAPATRA

Year : 2020

In 2017, about 792 million people (more than 10% of the global population) lived their lives with a mental disorder [24]– 78 million of which committed suicide because of it. In these unprecedented times of COVID-19, mental health challenges have been even further exacerbated as home environments have been proven to be major sources of the creation and worsening of poor mental health. Additionally, proper diagnosis and treatment for people with mental health disorders remains underdeveloped in modern-day's society due to the widely ever-present public stigma attached to caring about mental health. Recently there have been attempts in the data science world to predict if a person is suicidal (and other diagnostic approaches) yet all face major setbacks. To begin, big data has many ethical issues related to privacy and reusability without permission—especially in regards to using feeds from social media. Additionally, people diagnosed with specific mental health conditions may not actually seek treatment, so data may be incorrect. In this research, we address both of these problems by using anonymous datasets to predict the answer to a different question—whether or not people are seeking mental health treatment. We also use a large variety of machine learning and deep learning classifiers and predictive models to predict with a high accuracy rate through statistical analysis.

Title : Prediction of Mental Disorder for employees in IT Industry

Author: Sandhya P, Mahek Kantesaria

Year : 2019

Mental health is nowadays a topic which is most frequently discussed when it comes to research but least frequently discussed when it comes to the personal life. The wellbeing of a person is the measure of mental health. The increasing use of technology will lead to a lifestyle of less physical work. Also, the constant pressure on an employee in any industry will make more vulnerable to mental disorder. These vulnerabilities consist of peer pressure, anxiety stress, depression, and many more. Here we have taken the dataset of the questionnaires which were asked to an IT industry employee. Based on their answers the result is derived. Here output will be that the person needs an attention or not. Different machine learning techniques are used to get the results. This prediction also tells us that it is very important for an IT employee to get the regular mental health check up to tract their health. The employers should have a medical service provided in their company and they should also give benefits for the affected employees.

Title : Prediction of Mental Health Problems Among Children Using Machine Learning Techniques

Author: Ms. Sumathi M.R, Dr. B. Poorna

Year : 2016

Early diagnosis of mental health problems helps the professionals to treat it at an earlier stage and improves the patients' quality of life. So, there is an urgent need to treat basic mental health problems that prevail among children which may lead to complicated problems, if not treated at an early stage. Machine learning Techniques are currently well suited for analyzing medical data and diagnosing the problem. This research has identified eight machine learning techniques and has compared their performances on different measures of accuracy in diagnosing five basic mental health problems. A data set consisting of sixty cases is collected for training and testing the performance of the techniques. Twenty-five attributes have been identified as important for diagnosing the problem from the documents. The attributes have been reduced by applying Feature

Selection algorithms over the full attribute data set. The accuracy over the full attribute set and selected attribute set on various machine learning techniques have been compared. It is evident from the results that the three classifiers viz., Multilayer Perceptron, Multiclass Classifier and LAD Tree produced more accurate results and there is only a slight difference between their performances over full attribute set and selected attribute set.

Title : Artificial Intelligence for Mental Health and Mental Illnesses: an Overview

Author: Sarah Graham, Colin Depp

Year : 2019

Purpose of Review Artificial intelligence (AI) technology holds both great promise to transform mental healthcare and potential pitfalls. This article provides an overview of AI and current applications in healthcare, a review of recent original research on AI specific to mental health, and a discussion of how AI can supplement clinical practice while considering its current limitations, areas needing additional research, and ethical implications regarding AI technology. Recent Findings We reviewed 28 studies of AI and mental health that used electronic health records (EHRs), mood rating scales, brain imaging data, novel monitoring systems (e.g., smartphone, video), and social media platforms to predict, classify, or subgroup mental health illnesses including depression, schizophrenia or other psychiatric illnesses, and suicide ideation and attempts. Collectively, these studies revealed high accuracies and provided excellent examples of AI's potential in mental healthcare, but most should be considered early proof-of-concept works demonstrating the potential of using machine learning (ML) algorithms to address mental health questions, and which types of algorithms yield the best performance. Summary As AI techniques continue to be refined and improved, it will be possible to help mental health practitioners re-define mental illnesses more objectively than currently done in the DSM-5, identify these illnesses at an earlier or prodromal stage when interventions may be more effective, and personalize treatments based on an individual's unique characteristics. However, caution is necessary in order to avoid over-interpreting preliminary results, and more work is required to bridge the gap between AI in mental health research and clinical care

3.1 DEVELOPMENT PROCESS

3.1.1Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammer bacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyse large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming**: Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning**: Natural Language Processing, Classification, Clustering.
- **Data Visualization**: Tableau, SAS, D3.js, Python, Java, R libraries.

3.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, and speech recognition and machine vision search engines, recommendation systems. Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology. Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered on particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move

and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity. As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

3.3 Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of common sense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

3.4 MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple

machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they "learn" about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too.

Fig 3.1 PROCESS OF MACHINE LEARNING

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already

labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

3.5 PROJECT REQUIREMENTS

General:

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
   A. Hardware requirements
   B. Software reqirements

3.5.1 FUNCTIONAL REQUIREMENTS:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn

3.5.2 NON-FUNCTIONAL REQUREMENTS:

Process of functional steps,

1. Problem define
2. Preparing data

3. Evaluating algorithms
4. Improving results
5. Prediction the result
6. Environmental Requirements

- **Software reqirements**

Operating System	: Windows 10 or later
Tool	: Anaconda with Jupyter Notebook

- **Hardware requirements:**

Processor	: Intel i3
Hard disk	: minimum 80 GB
RAM	: minimum 2 GB

3.6 PROPOSED SYSTEM:

Stress datas from different sources would be combined to form a generalized dataset. In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using machine learning algorithms are applied on the Training set and based on the test result accuracy, Test set prediction is done. For the prediction of the stress, ML prediction model is effective because it is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.

3.6.1 ADVANTAGES:

- Accuracy may be improvised.
- Performance metrics will be compared

3.7 SOFTWARE ANALYSIS

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data

processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system "Conda". The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the `conda build` command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

3.7.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

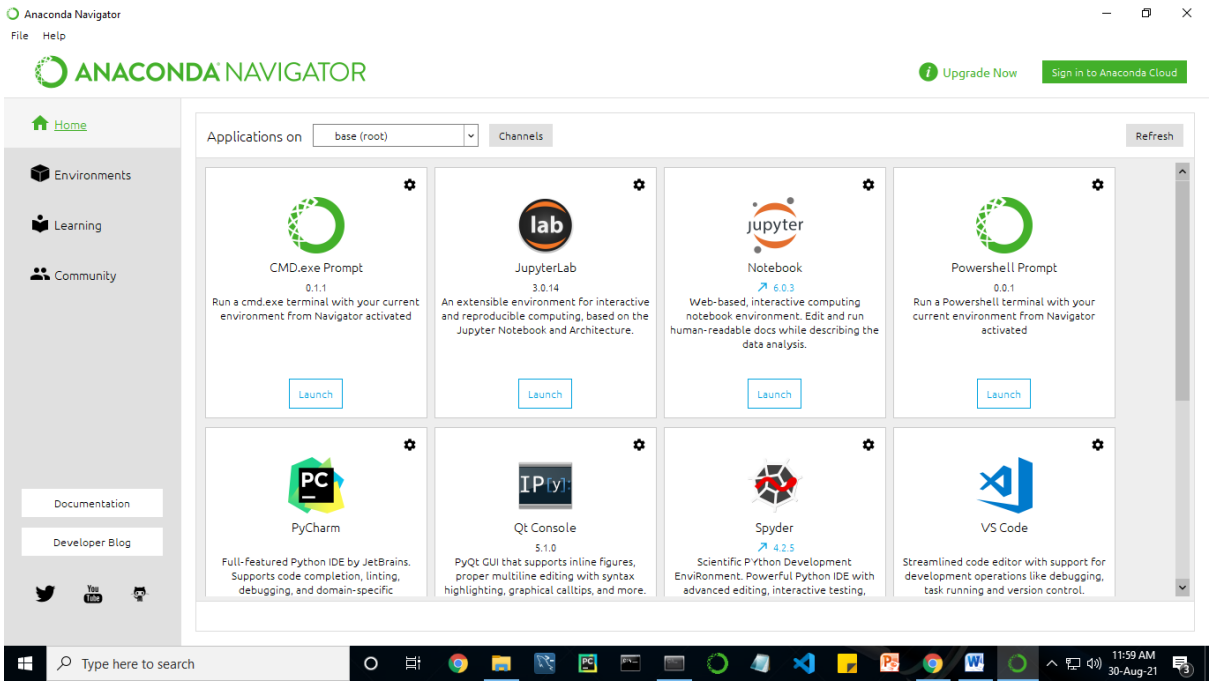


Fig 3.2 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with conda list on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. These softwares have great graphical user interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

3.7.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda

Running the Jupyter Notebook

Launching *Jupyter Notebook App:* The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): "jupyter notebook"

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy…) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- Launch the jupyter notebook app
- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- Click on the menu *Help -> User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing *shift + enter*.
- You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All.*

- To restart the kernel (i.e. the computational engine), click on the menu *Kernel -> Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser.

The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

Kernel: A notebook *kernel* is a "computational engine" that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results.

Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down.

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files.

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.

3.8 DESIGN ARCHITECTURE:

3.8.1 System Architecture:

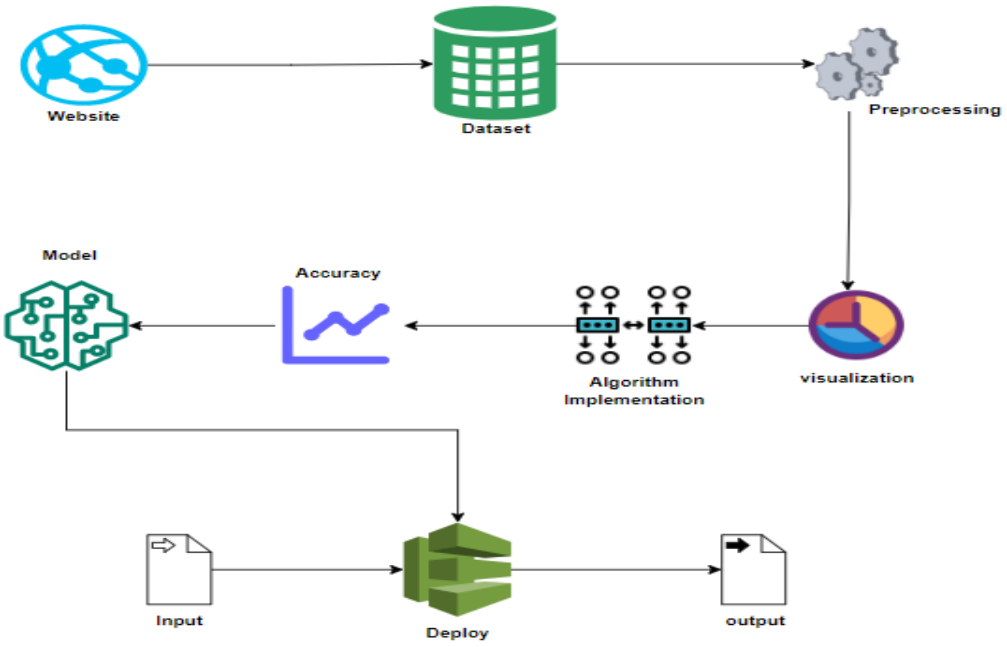


Fig 3.3 SYSTEM ARCHITECTURE

3.8.2 Work flow diagram

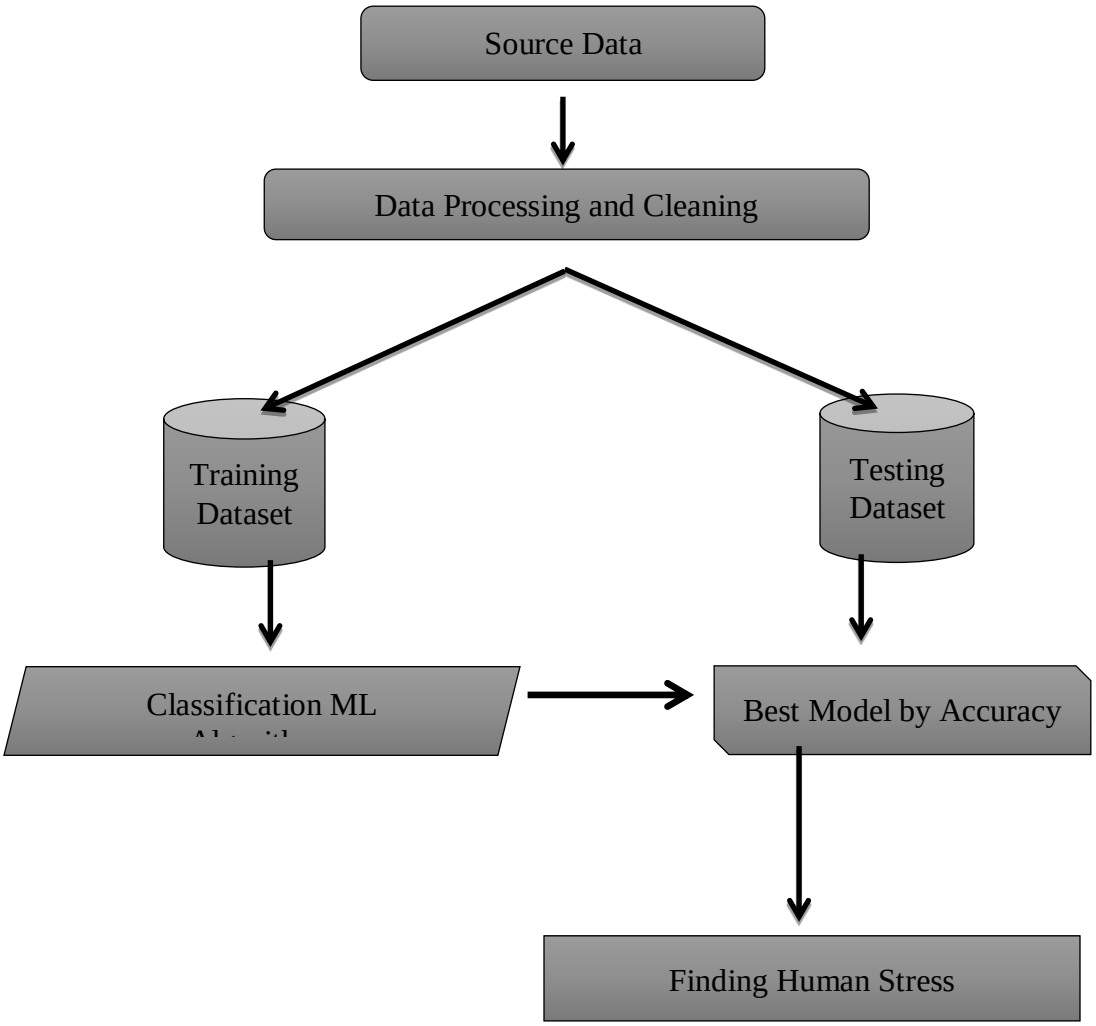


Fig 3.4 WORKFLOW DIAGRAM

3.9 LIST OF MODULES:

- Data Pre-processing
- Data Analysis of Visualization
- Implementing Algorithm 1
- Implementing Algorithm 2
- Implementing Algorithm 3
- Implementing Algorithm 4
- Deployment

3.9.1 MODULE DESCRIPTION:

3.9.1.1 Data Pre-processing:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

```
In [20]: 1 df.describe()
```

Out[20]:

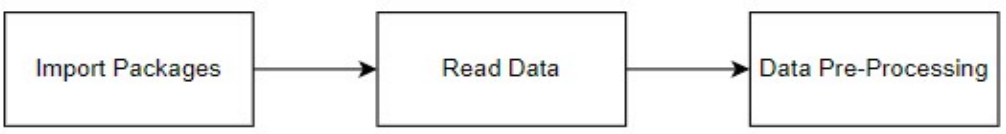
	sr	rr	t	lm	bo	rem	sr.1	hr	sl
count	630.000000	630.000000	630.00000	630.000000	630.000000	630.000000	630.000000	630.000000	630.000000
mean	71.600000	21.800000	92.80000	11.700000	90.900000	88.500000	3.700000	64.500000	2.000000
std	19.372833	3.966111	3.52969	4.299629	3.902483	11.893747	3.054572	9.915277	1.415337
min	45.000000	16.000000	85.00000	4.000000	82.000000	60.000000	0.000000	50.000000	0.000000
25%	52.500000	18.500000	90.50000	8.500000	88.500000	81.250000	0.500000	56.250000	1.000000
50%	70.000000	21.000000	93.00000	11.000000	91.000000	90.000000	3.500000	62.500000	2.000000
75%	91.250000	25.000000	95.50000	15.750000	94.250000	98.750000	6.500000	72.500000	3.000000
max	100.000000	30.000000	99.00000	19.000000	97.000000	105.000000	9.000000	85.000000	4.000000

```
In [21]: 1 df.corr()
```

Out[21]:

	sr	rr	t	lm	bo	rem	sr.1	hr	sl
sr	1.000000	0.976268	-0.902475	0.981078	-0.903140	0.950600	-0.920554	0.976268	0.975322
rr	0.976268	1.000000	-0.889237	0.991738	-0.889210	0.935572	-0.891855	1.000000	0.963516
t	-0.902475	-0.889237	1.000000	-0.896412	0.998108	-0.857299	0.954860	-0.889237	-0.962354
lm	0.981078	0.991738	-0.896412	1.000000	-0.898527	0.964703	-0.901102	0.991738	0.971071
bo	-0.903140	-0.889210	0.998108	-0.898527	1.000000	-0.862136	0.950189	-0.889210	-0.961092
rem	0.950600	0.935572	-0.857299	0.964703	-0.862136	1.000000	-0.893952	0.935572	0.951988
sr.1	-0.920554	-0.891855	0.954860	-0.901102	0.950189	-0.893952	1.000000	-0.891855	-0.973036
hr	0.976268	1.000000	-0.889237	0.991738	-0.889210	0.935572	-0.891855	1.000000	0.963516
sl	0.975322	0.963516	-0.962354	0.971071	-0.961092	0.951988	-0.973036	0.963516	1.000000

MODULE DIAGRAM



3.5 MODULE DIAGRAM

GIVEN INPUT EXPECTED OUTPUT

input : data

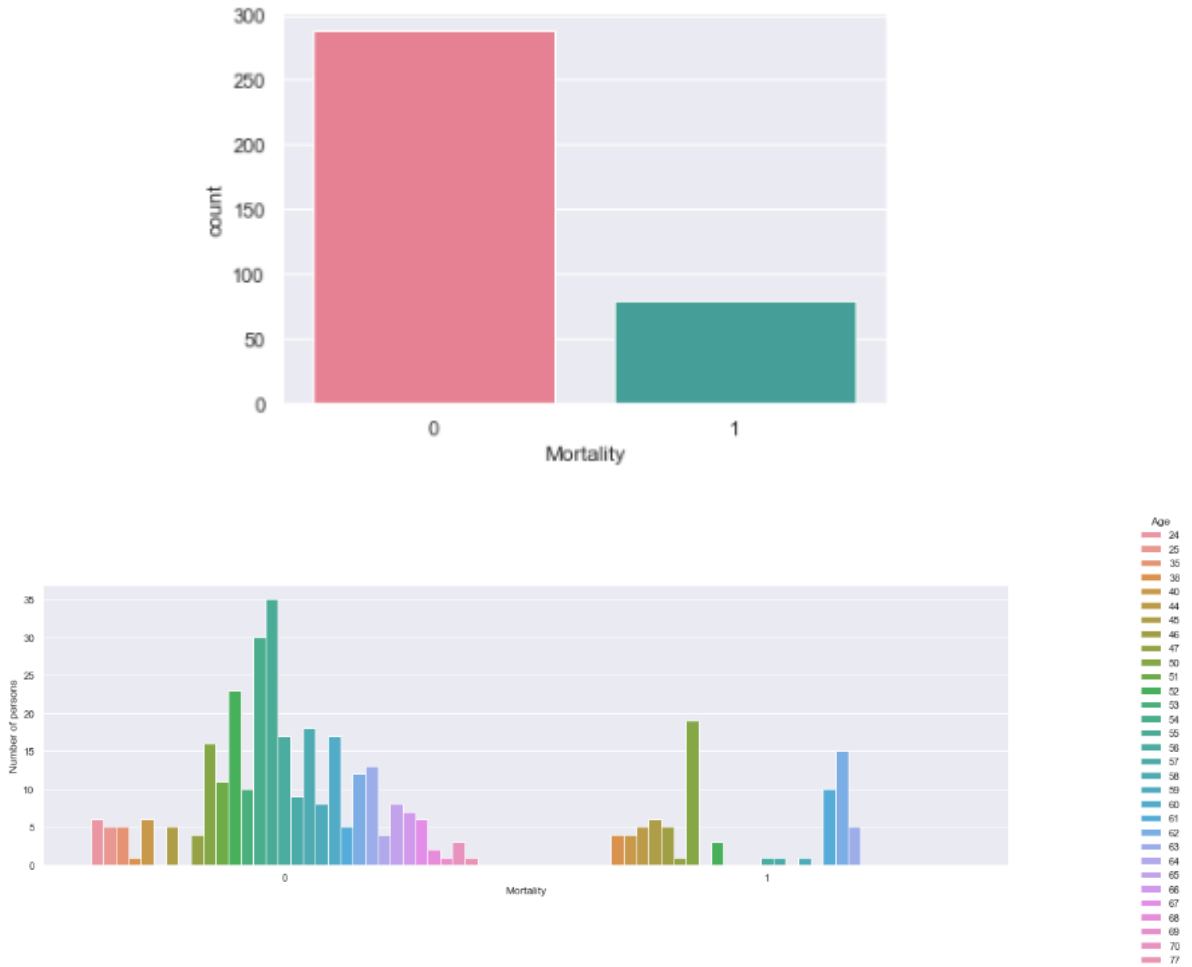
output : removing noisy data

3.9.1.2 Data visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.



MODULE DIAGRAM

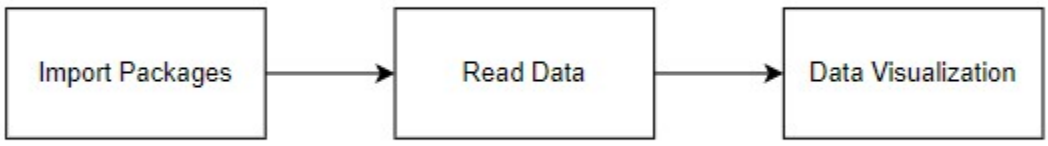


Fig 3.6 DATA VISUALISATION MODULE DIAGRAM

GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

Algorithm implementation:

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

Performance Metrics to calculate:

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

True Positive Rate(TPR) = TP / (TP + FN)

False Positive rate(FPR) = FP / (FP + TN)

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model

is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

Precision = TP / (TP + FP)

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labelled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

Recall = TP / (TP + FN)

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula:

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

The below 4 different algorithms are compared:

- SVM
- MLPClassifier
- Decision Tree Classifier
- AdaBoost

SVM:

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane.

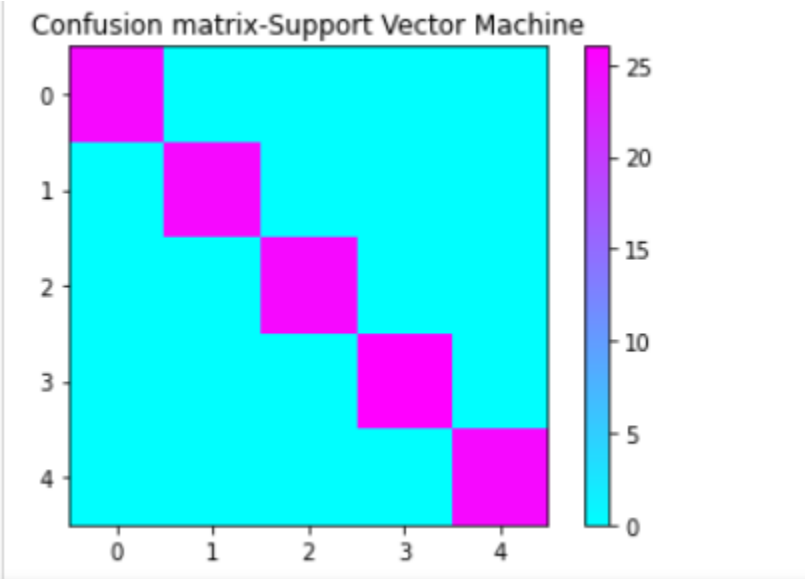
SVMs are used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages. This is one of the reasons we use SVMs in machine learning. It can handle both classification and regression on linear and non-linear data.

```
s = SVC()

s.fit(X_train,y_train)

predictS = s.predict(X_test)
```

```
print("")
print("Accuracy Result of Support Vector Machine is:",accuracy.mean() * 100)
svc=accuracy.mean() * 100
```



MODULE DIAGRAM

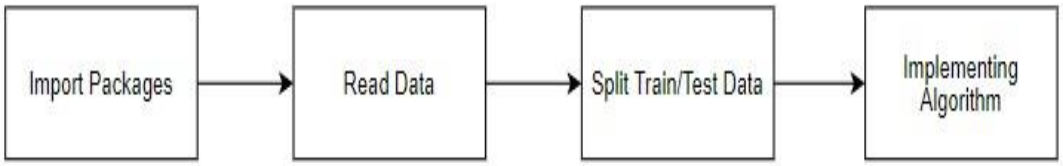


Fig 3.7 MODULE DIAGRAM FOR IMPLEMENTING ALGORITHM 1

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

3.9.1.3 MLPClasifier:

MLPClassifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of classification. One similarity though, with Scikit-Learn's other classification algorithms is that

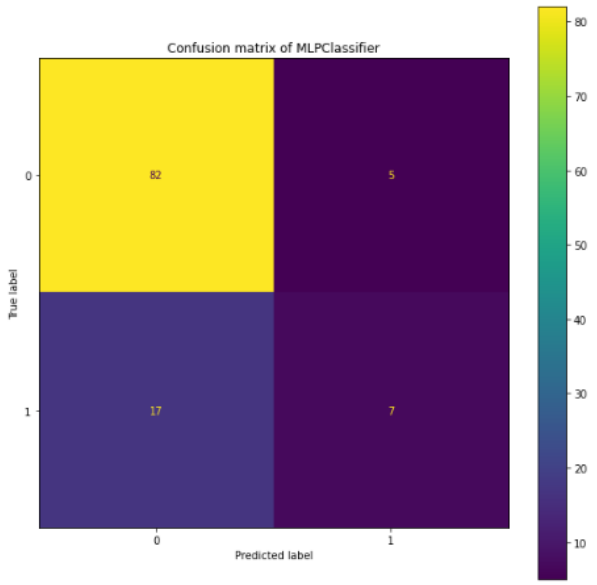
implementing MLPClassifier takes no more effort than implementing Support Vectors or Naive Bayes or any other classifiers from Scikit-Learn.

```
mlp = MLPClassifier()
mlp.fit(X_train,y_train)
predicted = mlp.predict(X_test)
```

Finding Accuracy

```
accuracy = accuracy_score(y_test,predicted)
print('Accuracy of MLPClassifier',accuracy*100)
```

```
Accuracy of MLPClassifier 80.18018018018019
```



MODULE DIAGRAM

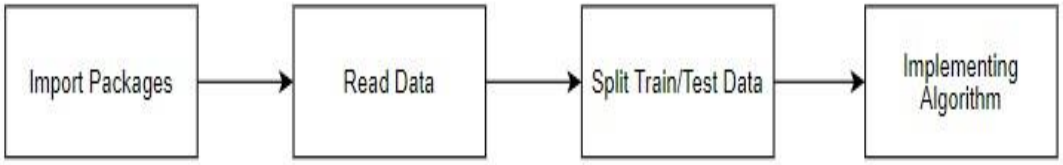


Fig 3.8 MODULE DIAGRAM FOR IMPLEMENTING ALGORITHM 2

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

3.9.1.4 Adaboost Classifier:

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem. The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. How does the AdaBoost algorithm work explain?

It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.

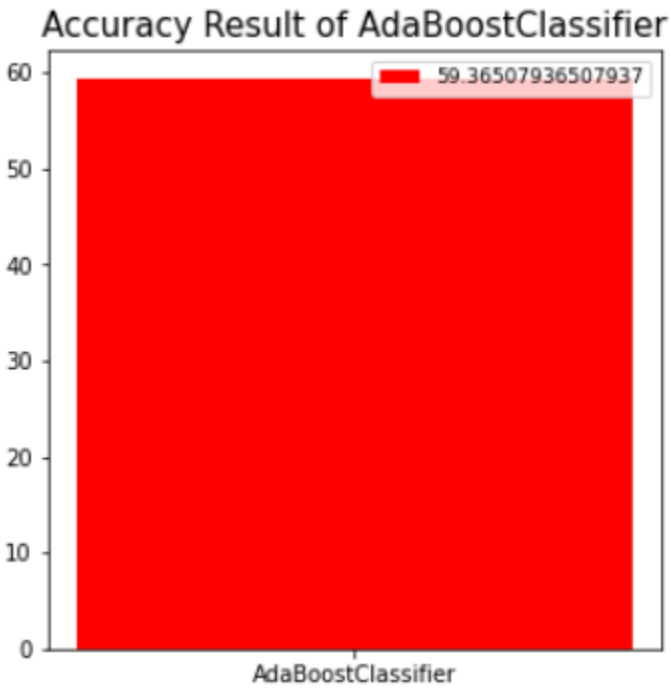
```
ADC = AdaBoostClassifier()

ADC.fit(X_train,y_train)

predictRF = ADC.predict(X_test)
```

```
accuracy = cross_val_score(ADC, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)


print("")
print("Accuracy Result of AdaBoostClassifier is:",accuracy.mean() * 100)
adc=accuracy.mean() * 100
```



MODULE DIAGRAM

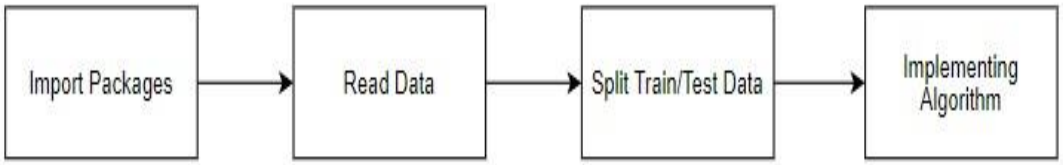


Fig 3.9 MODULE DIAGRAM FOR IMPLEMENTING ALGORITHM 3

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

3.9.1.5 Decision Tree Classifier:

Introduction Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves.

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

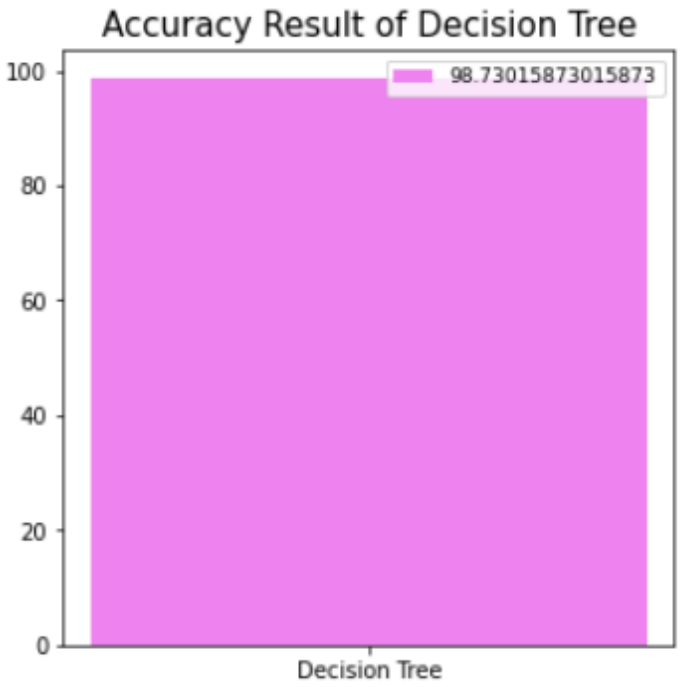
```
dt = DecisionTreeClassifier()

dt.fit(X_train,y_train)

predictDT = dt.predict(X_test)
```

```
accuracy = cross_val_score(dt, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)


print("")
print("Accuracy Result of Decision Tree is:",accuracy.mean() * 100)
DTC=accuracy.mean() * 100
```



MODULE DIAGRAM

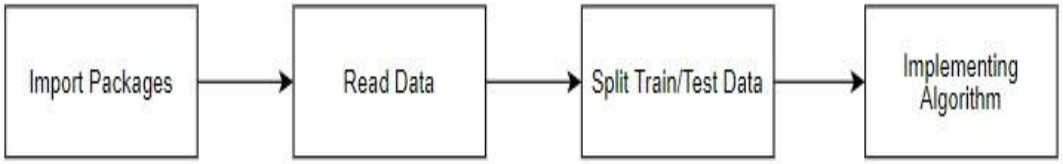


Fig 3.10 MODULE DIAGRAM FOR IMPLEMENTING ALGORITHM 4

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

3.10 Deployment:

3.10.1 Django (Web FrameWork) :

Django is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Django supports extensions that can add application features as if they were implemented in Django itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Django was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Django and related libraries passed to the newly formed Pallets project.

Django has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Django is part of the Pallets Projects, and based on several others of them.

Django is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Django is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity

among Python web developers. Let's take a closer look into Django, so-called "micro" framework for Python.

3.10.2 FEATURES:

Django was designed to be **easy to use and extend**. The idea behind Django is to build a solid foundation for web applications of different complexity. From then on you are free to **plug in any extensions** you think you need. Also you are free to build your own modules. Django is great for all kinds of projects. It's especially good for prototyping. Django depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Django as your web application framework if we have immensely powerful Django, Pyramid, and don't forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Django is pretty impressive too with it's:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus Django gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide how you will build your application.

- Django has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Django documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Django.
- It is super easy to deploy Django in production (Django is 100%_WSGI 1.0 compliant")
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Django is one of the most polished and feature-rich micro frameworks, available. Still young, Django has a thriving community, first-class extensions, and an **elegant API**. Django comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Django.

Django is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Django's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Django is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Django Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Django is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Django is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Django is a good start if you are getting into web development. Django is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Django is a web framework. This means Django provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Django:

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.

- Easy to develop and maintain applications.

Framework Django is a web framework from Python language. Django provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework Django still doesn't use the Model View Controller (MVC) method.

Django-RESTful is an extension for Django that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Django-Restful is a lightweight abstraction that works with the existing ORM/libraries. Django-RESTful encourages best practices with minimal setup.

Django Restful is an extension for Django that adds support for building REST APIs in Python using Django as the back-end. It encourages best practices and is very easy to set up. Django restful is very easy to pick up if you're already familiar with Django.

Django is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

Start Using an API

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.
4. The Django object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an __init__.py file inside) or a standard module (just a .py file).

For more information about resource loading, see open resource().

Usually you create a Django instance in your main module or in the __init__.py file of your package.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view_func.__name__.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
- **provide_automatic_options** (*Optional[bool]*) – Add the OPTIONS method and respond to OPTIONS requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.
  Return type -- None

After_Request(f)

- Register a function to run after each request to this object.
- The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

  If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use teardown_request() for that.

Parameters:

f (*Callable[[Response*], *Response]*)

eturn type

Callable[[Response], Response]

after_request_funcs: t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the after_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app_context()

Create an AppContext. Use as a with block to push the context, which will make current_app point at this application.

An application context is automatically pushed by RequestContext.push() when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app_context():

4. SYSTEM STUDY

4.1 Aim:

Stress Detection is one of the major factors in our healthcare domain. There are lot of patients who are actively present in the world. It is difficult to find the stress detection. So this project can easily find out the stress detection.

4.2 Objectives:

The goal is to develop a machine learning model for stress detection Prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

4.3 Scope of the Project

Here the scope of the project is that integration of patients stress detection with computer-based prediction could reduce errors and improve prediction outcome. This suggestion is promising as data modeling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of stress detection prediction.

4.4 Feasibility study:

4.4.1Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

4.4.2Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

4.4.3 Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

4.4.4Building the classification model

The prediction of Human Stress, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.**Construction of a Predictive Model**

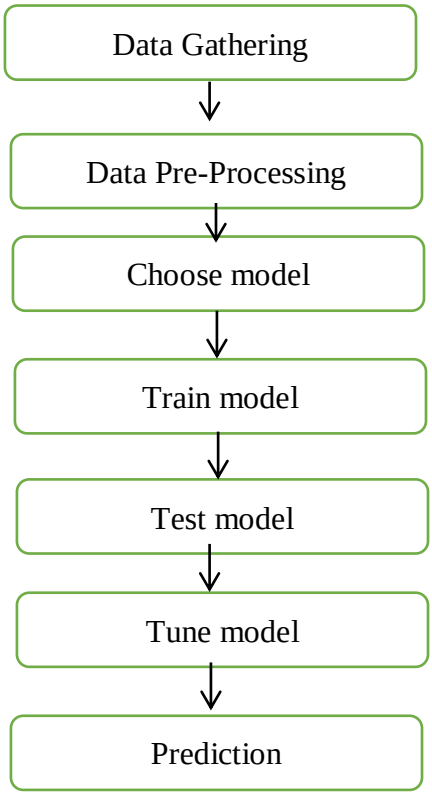


Fig 4.1 PREPROCESSING MODEL

5.UML DIAGRAMS

5.1 Use Case Diagram:

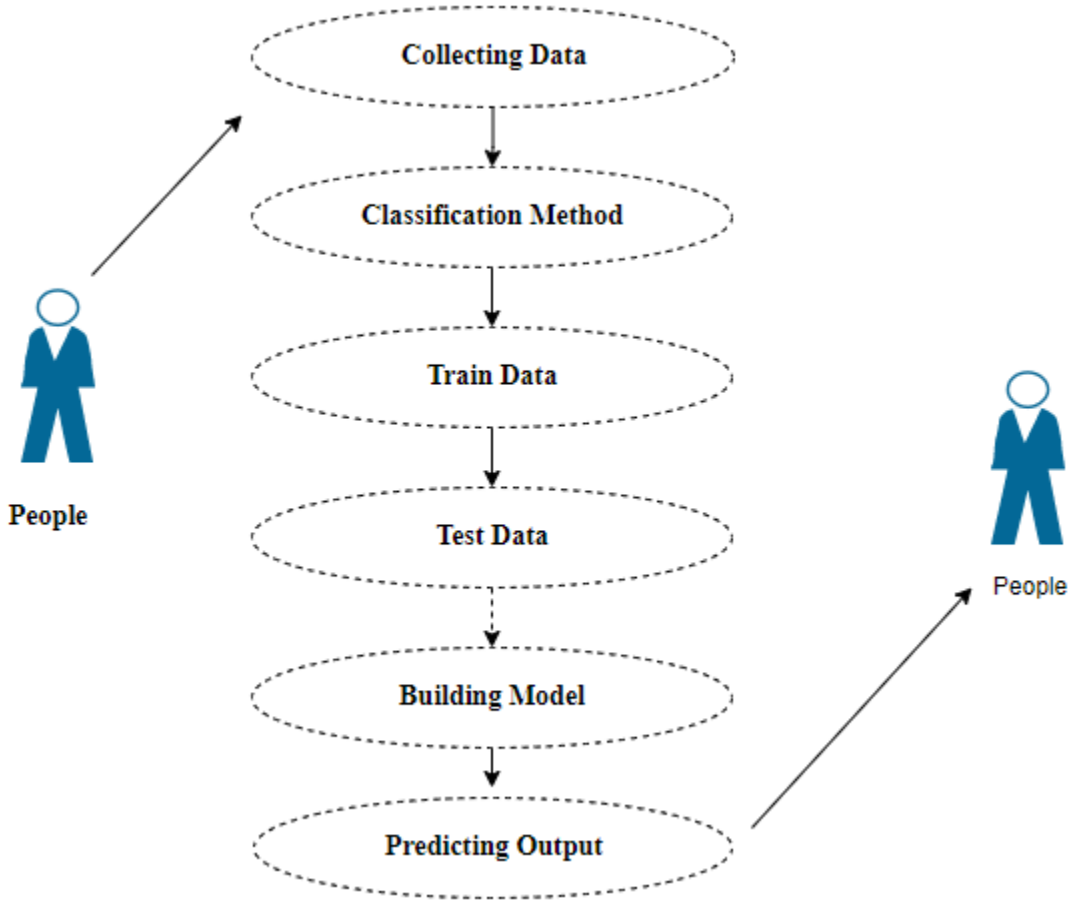


FIG 5.1 USECASE DIAGRAM

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

5.2 Class Diagram:

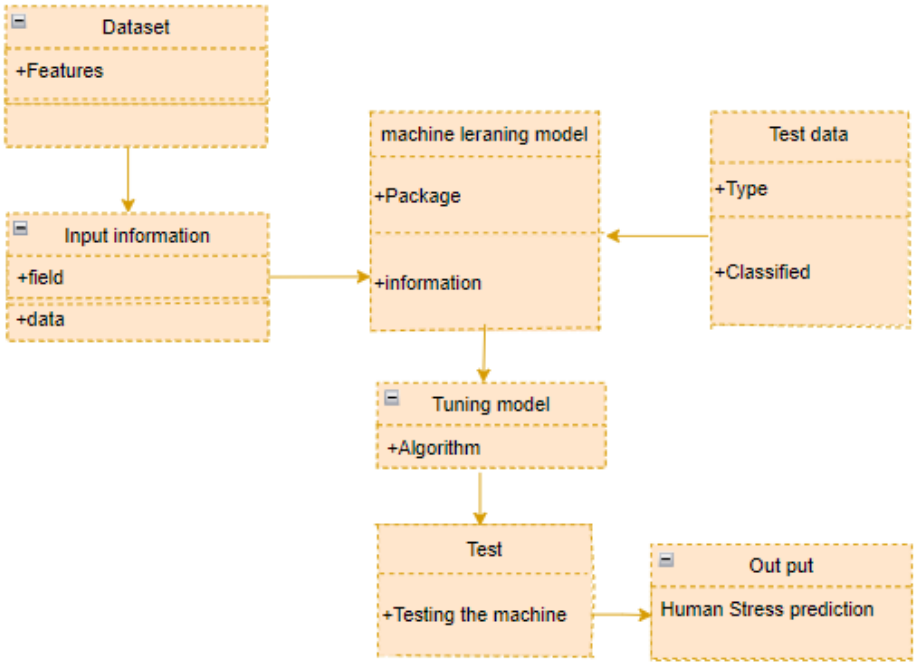


Fig 5.2 CLASS DIAGRAM

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

5.3 Activity Diagram:

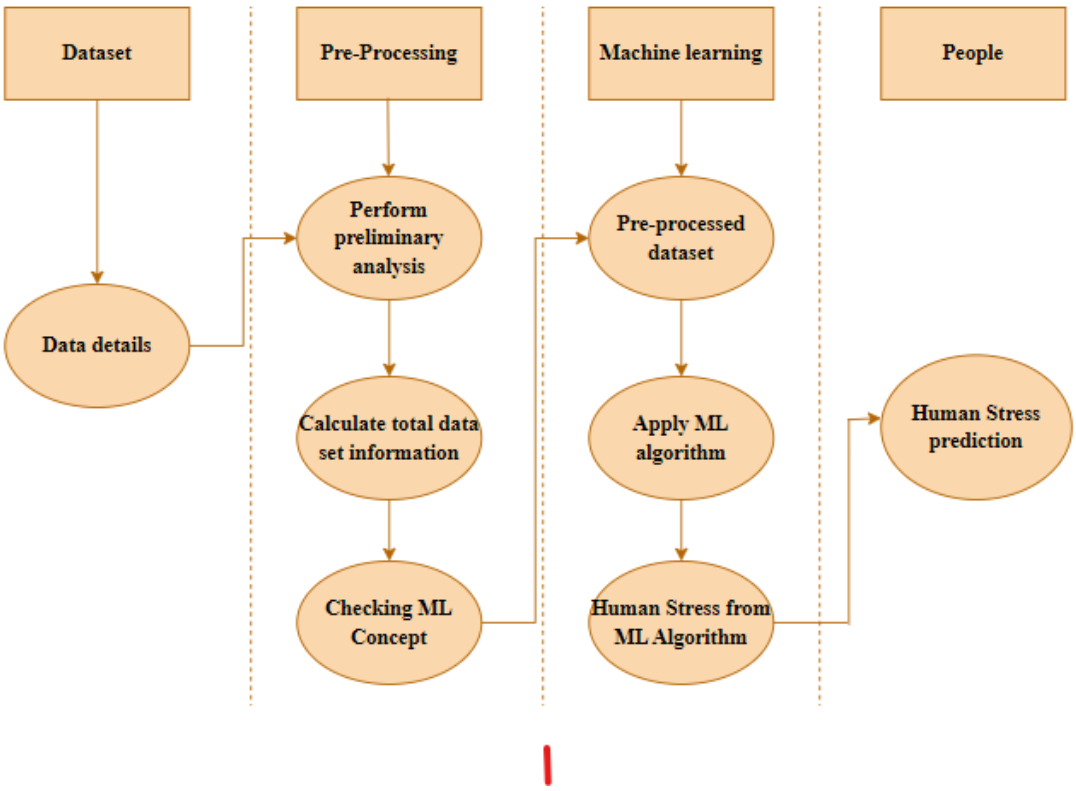


Fig 5.3 ACTIVITY DIAGRAM

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

5.4 Sequence Diagram:

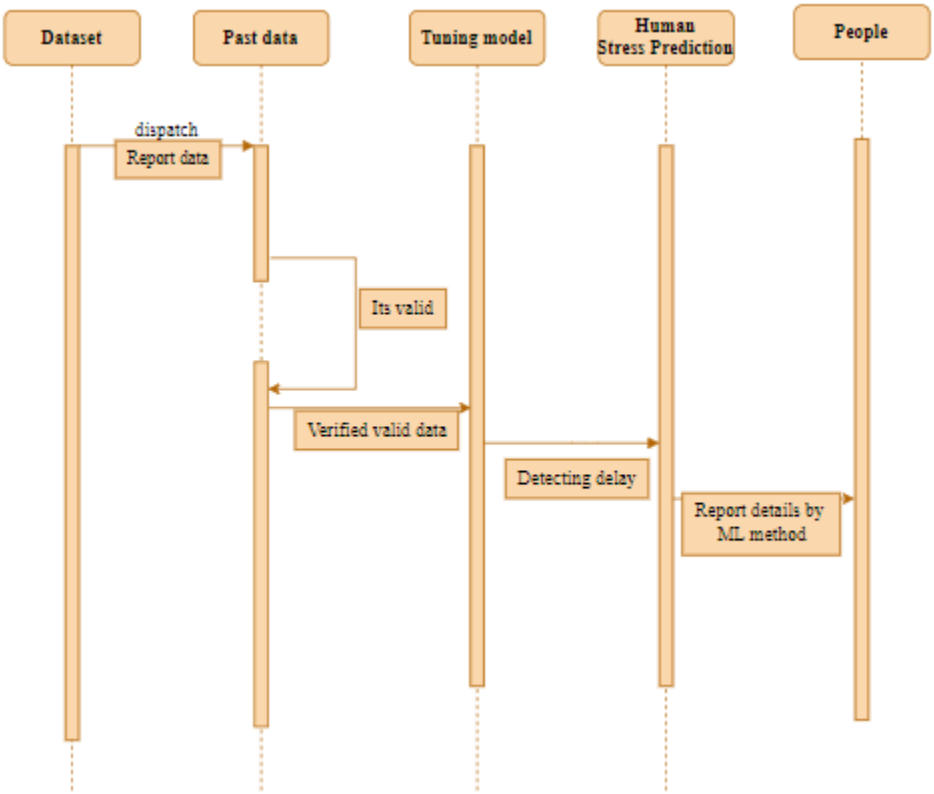


Fig 5.4 SEQUENCE DIAGRAM

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

5.5 Entity Relationship Diagram

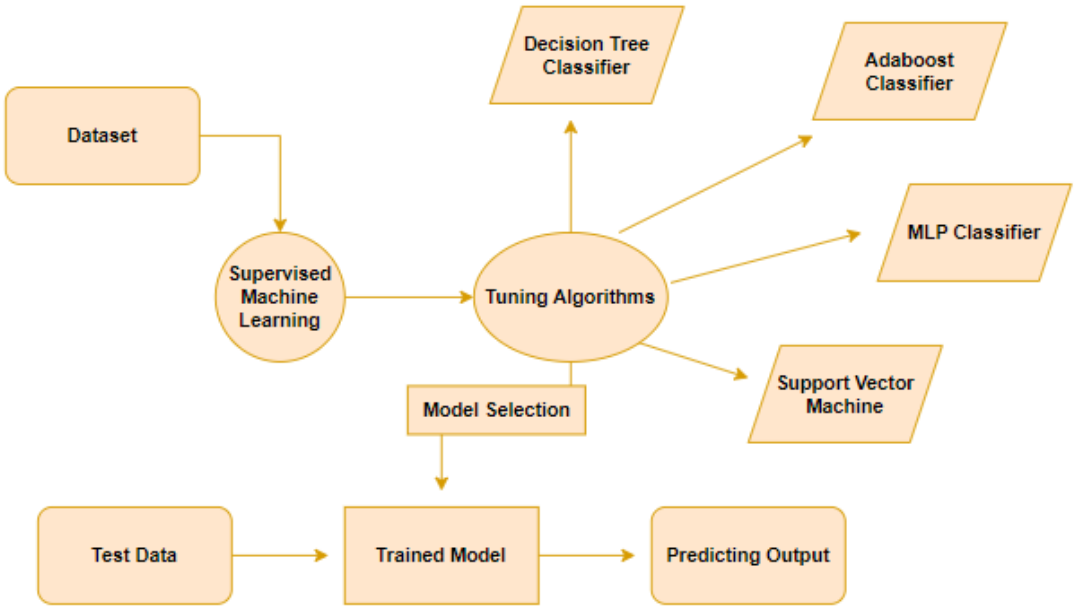


Fig 5.5 Entity Relationship Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

6. SOFTWARE DESCRIPTION

6.1PYTHON

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

Syntax and Semantics :

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation :

Main article: Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation

signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow:

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.

- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.[80][81] However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

6.2 HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

6.3 CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

6.3.1 Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document
- Example:

6.3.2 External CSS

- With the help of <link> tag in the head tag, we can apply styles
- Reference is added
- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document

Deploying the model predicting output

In this module the trained machine learning model is converted into pickle data format file (.pkl file) which is then deployed for providing better user interface and predicting the output of Human Stress.

7. TESTING

7.1 White Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internalperspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuittesting (ICT). While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this methodof test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques mentionedabove: control flow testing, data flow testing, branch testing, path testing, statementcoverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error freeenvironment by examining any fragile code. These White-box testing techniques arethe building blocks of white-box testing, whose essence is the careful testing of theapplication at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the codeis working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aidsin any defects that happen later on after the code is integrated with the rest of theapplication and therefore prevents any type of errors later on.
2. Integration testing. White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour.
3. Open environment through the use of white-box testing for any interactions ofinterfaces that are known to the programmer.

Regression testing. White-box testing during regression testing is the use ofrecycled white-box test cases at the unit and integration testing level.

7.2 Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into itsinternal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, butcan also dominate unit testing as well.

7.3 Test procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of how it does it. For instance, the testeris aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

Test cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from externaldescriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functiontests may also be used. The test designer selects both valid and invalid.

inputs and determines the correct output without any knowledge of the test object's internal structure.

7.4 Test design techniques

Decision table testing

All-pairs testing

State transition tables

Equivalence partitioning

Boundary value analysis

7.5 Test design techniques

Decision table testing

All-pairs testing

State transition tables

Equivalence partitioning

Boundary value analysis

7.6 Performance testing

In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

7.7 Unit testing

In computer programming, **unit testing** is a method by which individual unitsof source code, sets of one or more computer program modules together with a more

associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entiremodule, but is more commonly an individual function or procedure. In object- oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionallyby white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation.

Testing will not catch every error in the program, since it cannot evaluate every execution path in any but the most trivial programs. The same is true for unittesting. Additionally, unit testing by

definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system- level errors (such as functions performed across multiple units, or non-functional test areas such as performance). Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. In order to guarantee correct behaviour for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behaviour. Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs

7.8 Software testing

Software testing is a combinatorial problem. For example, every Boolean decision statement requires at least two tests: one with an outcome of "true" and onewith an outcome of "false". As a result, for every line of code written, programmersoften need 3 to 5 lines of test code. This obviously takes time and its investment maynot be worth the effort. There are also many problems that cannot easily be tested atall – for example those that are nondeterministic or involve multiple threads. In addition, code for a unit test is likely to be at least as buggy as the code it is testing.Fred Brooks in The Mythical Man-Month quotes: never take two chronometers to sea. Always take one or three. Meaning, if two chronometers contradict, how do youknow which one is correct. Another challenge related to writing the unit tests is thedifficulty of setting up realistic and useful tests. It is necessary to create relevant initial conditions so the part of the application being tested behaves like part of the complete system. If these initial conditions are not set correctly, the test will not beexercising the code in a realistic context, which diminishes the value and accuracy of unit test results. To obtain the intended benefits from unit testing, rigorous discipline is needed throughout the software development process. It is essential to keep careful records not only of the tests that have been performed, but also of all changes that have been made to the source code of this or any other unit in the software. Use of a version control system is essential. If a later version of the unit fails a particular test that it had previously passed, the version-control software can provide a list of the source code changes (if any) that have been applied to the unit since that time.

It is also essential to implement a sustainable process for ensuring that test case failures are reviewed daily and addressed immediately if such a processis not implemented and ingrained into the team's workflow, the application will evolve out of sync

7.8.1 Functional testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes what the system does.

Functional testing typically involves five steps. The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications
2. The determination of output based on the function's specifications
3. The execution of the test case
4. The comparison of actual and expected outputs

7.9 Testing types

7.9.1 Load testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business

critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

7.9.2 Stress testing

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

7.9.3 Soak testing

Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long periods of sustained activity are as good asor better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

7.9.4 Spike testing

Spike testing is done by suddenly increasing the number of or load generatedby, users by a very large amount and observing the behaviour of the system. The

goal is to determine whether performance will suffer, the system will fail, or it willbe able to handle dramatic changes in load.

7.9.5 Configuration testing

Rather than testing for performance from the perspective of load, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behaviour. A common examplewould be experimenting with different methods of load-balancing.

7.9.6 Isolation testing

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

7.9.7Integration testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual softwaremodules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system.

8. CODING

8.1 Module – 1

In [ ]:

```
import pandas as p
import numpy as n
```
In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```
In [ ]:

```
data=p.read_csv('human.csv')
```
In [ ]:

```
data.head()
```
In [ ]:

```
data.shape
```
In [ ]:

```
data.size
```
In [ ]:

```
data.isnull()
```
In [ ]:

```
data.columns
```
In [ ]:

```
data['sl'].unique()
```
In [ ]:

```
data[data["sl"] == 2]
```
In [ ]:

```
data[data["sl"] == 3]
```
In [ ]:

```
a = data["bo"].unique()
a.sort()
print(a)
```
In [ ]:

```
data["hr"].unique()
```
In [ ]:

```
data["sr"].unique()
```
In [ ]:

```
df = data.dropna()
In [ ]:

df
In [ ]:

df.isnull().sum()
In [ ]:

df.shape
In [ ]:

print("Maximum Age :",df["rem"].max())
print("Minimum Age :",df["rem"].min())
print("Mean of Age :","%.2f" % df["rem"].mean())
print("Median of Age :","%.2f" % df["rem"].median())
In [ ]:

df.describe()
In [ ]:

df.corr()
In [ ]:

df.info()
In [ ]:

p.crosstab(df["rem"], df["sl"])
In [ ]:

df.groupby(["sr.1","sl"]).groups
In [ ]:

df["sl"].value_counts()
In [ ]:

df["hr"].value_counts()
In [ ]:

p.Categorical(df["rr"]).describe()
In [ ]:

p.Categorical(df["sr.1"]).describe()
In [ ]:

df.duplicated()
```

8.2 Module - 2

In [ ]:

```
import pandas as p
import numpy as n
import seaborn as s
import matplotlib.pyplot as plt
```
In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```
In [ ]:

```
data=p.read_csv('human.csv')
```
In [ ]:

```
data
```
In [ ]:

```
df=data.dropna()
```
In [ ]:

```
df.columns
```
In [ ]:

```
p.crosstab(df.sl,df.rem)
```
In [ ]:

```
#Histogram
df['rem'].hist(figsize=(10,5), color='m', alpha=1)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('EYE MOVEMENT')
```
In [ ]:

```
plt.figure(figsize=(16,4))
plt.subplot(1,2,1)
plt.hist(df['rr'])
plt.title("RESPIRATION RATE")
plt.xlabel("X")
plt.ylabel("Y")
plt.subplot(1,2,2)
plt.hist(df['t'])
plt.xlabel('X')
plt.title('BODY TEMPERATURE')
plt.ylabel("Y")
plt.show()
```
In [ ]:

```
plt.figure(figsize=(9,6))
```

```
s.scatterplot(x=df['t'],y=df["rr"])
plt.show()
In [ ]:

plt.figure(figsize=(12,8))
plt.scatter(df["rem"],df["hr"],color="red")
In [ ]:

import seaborn as s
s.boxplot(df['sr'], color='c')
In [ ]:

import seaborn as s
s.boxplot(df['lm'], color='y')
In [ ]:

#Density Subplot
plt.figure(figsize= (18,12))
plt.subplot(3,3,1)
df["sl"].plot(kind='density')
plt.title("sl")
plt.subplot(3,3,2)
df["bo"].plot(kind='density')
plt.title("bo")
plt.subplot(3,3,3)
df["lm"].plot(kind='density')
plt.title("lm")
plt.subplot(3,3,4)
df["t"].plot(kind='density')
plt.title("t")
plt.subplot(3,3,5)
df["rr"].plot(kind='density')
plt.title("rr")
plt.subplot(3,3,6)
df["rem"].plot(kind='density')
plt.title("rem")
plt.subplot(3,3,7)
df["sr.1"].plot(kind='density')
plt.title("sr.1")
plt.subplot(3,3,8)
df["hr"].plot(kind='density')
plt.title("hr")
plt.subplot(3,3,9)
df["sr"].plot(kind='density')
plt.title("sr")

plt.show()
```

8.3 Module - 3

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=p.read_csv('human.csv')
data.head()
```

In [ ]:

```
data.columns
```

In [ ]:

```
df=data.dropna()
```

In [ ]:

```
df
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.tail()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df.info()
```

In [ ]:

```
df
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
```

```
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT        : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
In [ ]:

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1, stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
In [ ]:

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
In [ ]:

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score

s = SVC()

s.fit(X_train,y_train)

predictS = s.predict(X_test)

print("")
print('Classification report of Support Vector Machine Result is:')
print("")
print(classification_report(y_test,predictS))
print("")

cm=confusion_matrix(y_test,predictS)
print('Confusion Matrix result of Support Vector Machine is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(s, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
```

8.4 Module - 4

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```
In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```
In [ ]:

```
data=p.read_csv('human.csv')
```
In [ ]:

```
data.columns
```
In [ ]:

```
df=data.dropna()
```
In [ ]:

```
df
```
In [ ]:

```
df.columns
```
In [ ]:

```
df.head()
```
In [ ]:

```
df.shape
```
In [ ]:

```
df.info()
```
In [ ]:

```
df
```
In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```
In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
```

```
ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT        : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
In [ ]:

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1, stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
In [ ]:

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
In [ ]:

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score

ADC = AdaBoostClassifier()

ADC.fit(X_train,y_train)

predictRF = ADC.predict(X_test)

print("")
print('Classification report of AdaBoostClassifier Result is:')
print("")
print(classification_report(y_test,predictRF))
print("")

cm=confusion_matrix(y_test,predictRF)
print('Confusion Matrix result of AdaBoostClassifier is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(ADC, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
```

8.5 Module - 5

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```
In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```
In [ ]:

```
data=p.read_csv('human.csv')
```
In [ ]:

```
data.columns
```
In [ ]:

```
df=data.dropna()
```
In [ ]:

```
df
```
In [ ]:

```
df.columns
```
In [ ]:

```
df.head()
```
In [ ]:

```
df.shape
```
In [ ]:

```
df.info()
```
In [ ]:

```
df
```
In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```
In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
```

```
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT         : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
In [ ]:

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1, stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
In [ ]:

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
In [ ]:

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score

mlp = MLPClassifier()

mlp.fit(X_train,y_train)

predictLR = mlp.predict(X_test)

print("")
print('Classification report of MLPClassifier Result is:')
print("")
print(classification_report(y_test,predictLR))
print("")

cm=confusion_matrix(y_test,predictLR)
print('Confusion Matrix result of MLPClassifier is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(mlp, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

print("")
print("Accuracy Result of MLPClassifier is:",accuracy.mean() * 100)
mlp=accuracy.mean() * 100
```

In [ ]:

```
def Bar_Chart():
    import matplotlib.pyplot as plt
    data=[mlp]
    alg="MLPClassifier"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("m"))
    plt.title("Accuracy Result of MLPClassifier",fontsize=15)
    plt.legend(b,data,fontsize=9)
Bar_Chart()
```

In [ ]:

```
TP = cm[0][0]
FP = cm[1][0]
FN = cm[0][1]
TN = cm[1][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-MLPClassifier', cmap=plt.cm.autumn):
    plt.imshow(cm, interpolation='hermite', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predictLR)
print('Confusion matrix-MLPClassifier:')
```

8.6 Module - 6

In [ ]:
```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
data=p.read_csv('human.csv')
```

In [ ]:
```
data.columns
```

In [ ]:
```
df=data.dropna()
df = df.rename({'sr.1':'srh'},axis=1)
```

In [ ]:
```
df
```

In [ ]:
```
df.columns
```

In [ ]:
```
df.head()
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.info()
```

In [ ]:
```
df
```

In [ ]:
```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```

In [ ]:
```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT        : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1, stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
```

In [ ]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

dt = DecisionTreeClassifier()

dt.fit(X_train,y_train)

predictDT = dt.predict(X_test)

print("")
print('Classification report of Decision Tree Result is:')
print("")
print(classification_report(y_test,predictDT))
print("")

cm=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of Decision Tree is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(dt, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)


print("")
print("Accuracy Result of Decision Tree is:",accuracy.mean() * 100)
DTC=accuracy.mean() * 100
```

In [ ]:

```
def Bar_Chart():
    import matplotlib.pyplot as plt
    data=[DTC]
```

```
    alg="Decision Tree"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("Violet"))
    plt.title("Accuracy Result of Decision Tree",fontsize=15)
    plt.legend(b,data,fontsize=9)
Bar_Chart()
```

In [ ]:

```
TP = cm[0][0]
FP = cm[1][0]
FN = cm[0][1]
TN = cm[1][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-Decision Tree', cmap=plt.cm.BuPu):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predictDT)
print('Confusion matrix-Decision Tree:')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

```
import joblib

joblib.dump(dt,'model.pkl')
```

Deploy

```
from django.shortcuts import render
```

```
from django.shortcuts import render, redirect
import numpy as np
import joblib

model = joblib.load('C:/Users/SURENDAR A/Music/PROJECT/CODE/deploy/app/model.pkl')

# Create your views here.
def home(request):
    return render(request, "index.html")

def predict(request):
    if request.method == "POST":
        int_features = [x for x in request.POST.values()]
        int_features = int_features[1:]
        print(int_features)
        final_features = [np.array(int_features, dtype=object)]
        print(final_features)
        prediction = model.predict(final_features)
        print(prediction)
        output = prediction[0]
        print(f'output{output}')
        if output == 0:
            return render(request, 'index.html', {"prediction_text":"VERY LESS STRESS AFFECTED"})
```

9. SCREENSHOT:

Fig 9.1 LOGIN PAGE

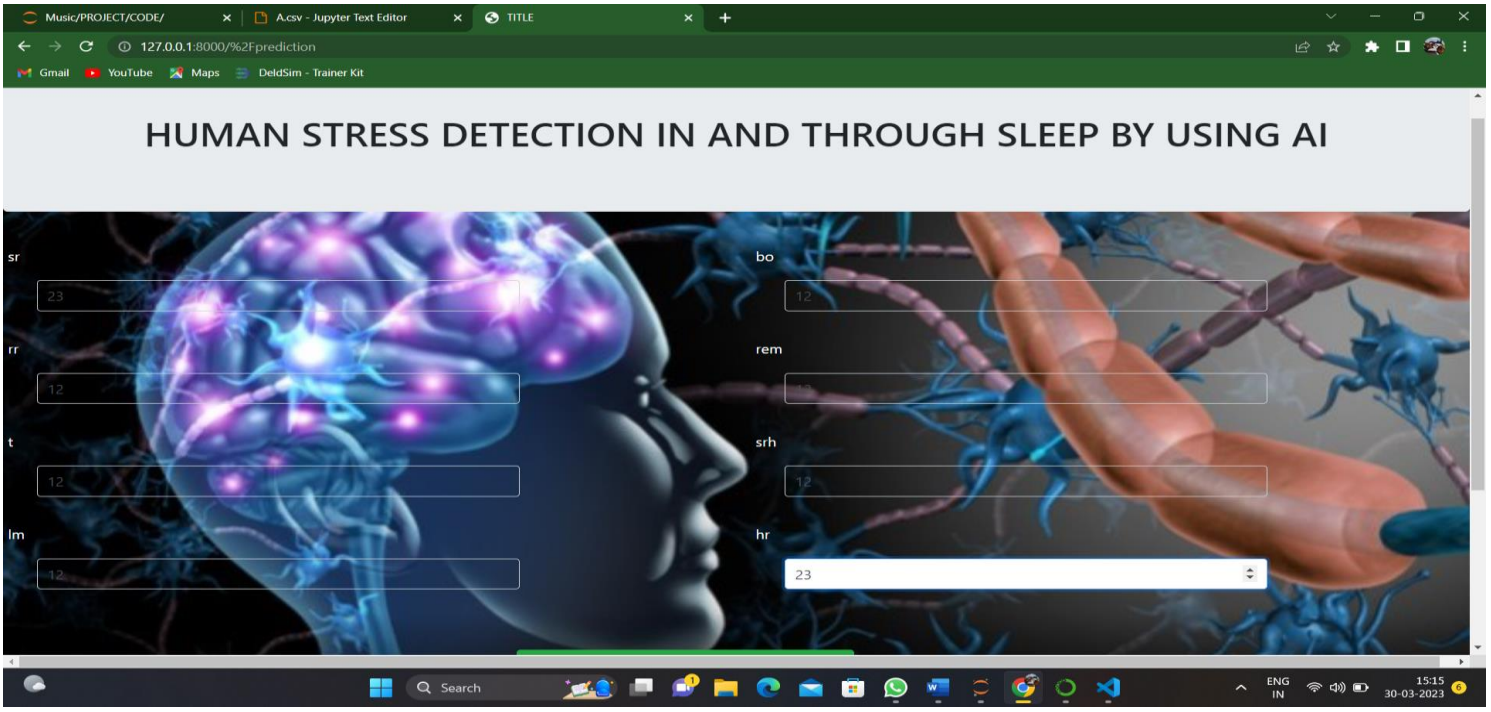


Fig 9.2 STRESS DETECTION PAGE

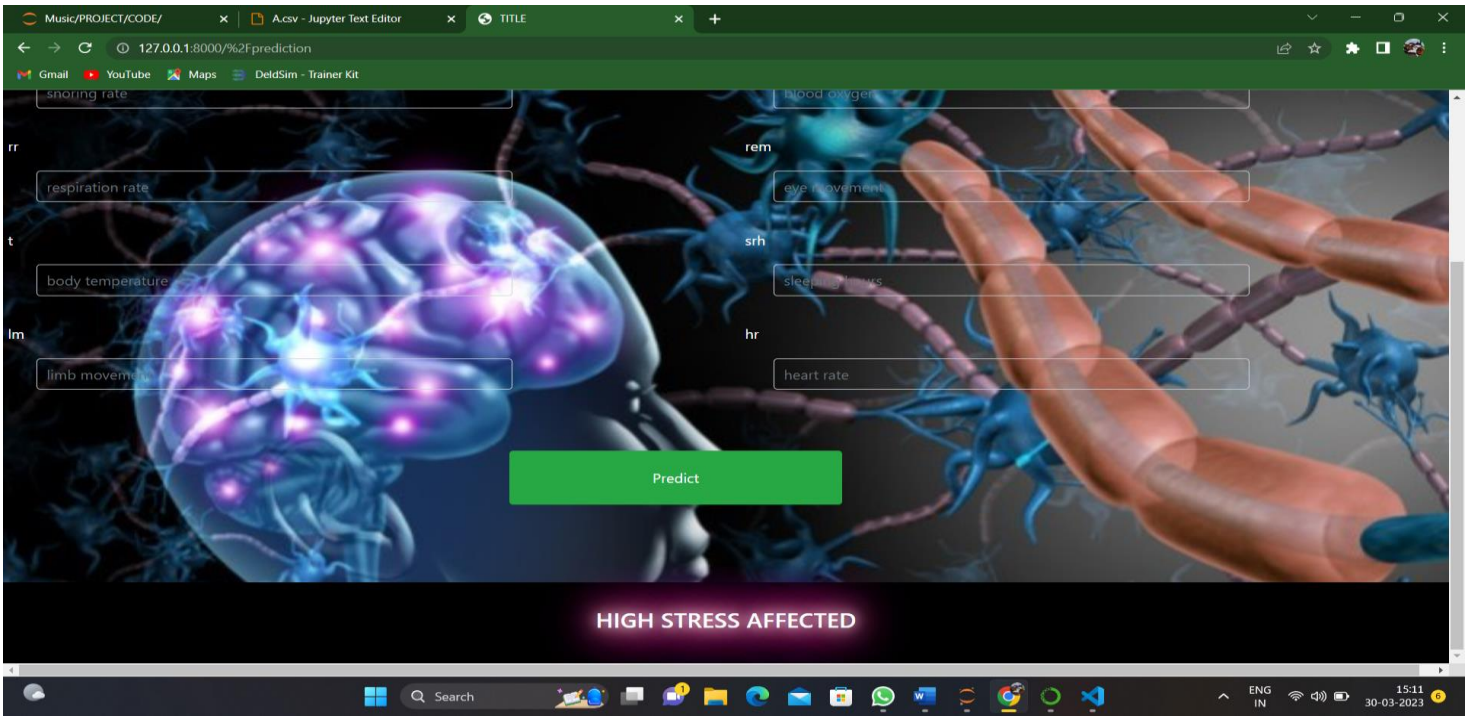


Fig 9.3 RESULTANT PAGE

10. Conclusion

10.1 Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the Human Stress of the patient.

10.2 Future Work:

- Deploying the project in the cloud.
- To optimize the work to implement in the IOT system.

REFERENCES

- Prediction of mental disorder for employees in IT industry (Sandhya P.Mahekkanesaria)(International journal of innovative technology and exploring engineering IJITEE)
- Machine learning -based approach for depression detection (Hatoon Alsagri,Mourad Yklef)
- Prediction the utilization of mental health treatment with various machine learning algorithms.(Meera sharma,sonak Mahapatra)
- Artificial intelligence for mental health and mental illness(Sarah graham,colin deep)
- K. Sengupta, "Stress Detection: A Predictive Analysis," 2021 Asian Conference on Innovation in Technology (ASIANCON), PUNE, India, 2021, pp. 1-6, doi: 10.1109/ASIANCON51346.2021.9544609.

- J. G. Jayawickrama and R. A. H. M. Rupasingha, "Ensemble Learning Approach to Human Stress Detection Based on Behaviours During the Sleep," 2022 4th International Conference on Advancements in Computing (ICAC), Colombo, Sri Lanka, 2022, pp. 132-137, doi: 10.1109/ICAC57685.2022.10025175.

- G. Azar, C. Gloster, N. El-Bathy, S. Yu, R. H. Neela and I. Alothman, "Intelligent data mining and machine learning for mental health diagnosis using genetic algorithm," 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, 2015, pp. 201-206.

- B. Saha, T. Nguyen, D. Phung and S. Venkatesh, "A Framework for Classifying Online Mental Health-Related Communities With an Interest in Depression," in IEEE Journal of Biomedical and Health Informatics, vol. 20, no. 4, pp. 1008-1015, July 2016.

- T. Simms, C. Ramstedt, M. Rich, M. Richards, T. Martinez and C. Giraud-Carrier, "Detecting Cognitive Distortions Through Machine Learning Text Analytics," 2017 IEEE International Conference on Healthcare Informatics (ICHI), Park City, UT, 2017, pp. 508-512.

- A. R. Subhani, W. Mumtaz, M. N. B. M. Saad, N. Kamel and A. S. Malik, "Machine Learning Framework for the Detection of Mental Stress at Multiple Levels," in IEEE Access, vol. 5, pp. 13545-13556, 2017.

- R. M. Khalil and A. Al-Jumaily, "Machine learning based prediction of depression among type 2 diabetic patients," 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, 2017, pp. 1-5.

- Srividya, M & Subramaniam, Mohanavalli & Natarajan, Bhalaji. (2018). Behavioral Modeling for Mental Health using Machine Learning Algorithms. Journal of Medical Systems. 42. 88. 10.1007/s10916-018-0934-5.

- Ashar Khan, Mohd Shahid Husain & Anam Khan. (2018). Analysis of Mental State of Users using Social Media to predict Depression! A Survey. International Journal of Advanced Research in Computer Science, II International Conference on "Advancement in Computer Engineering and Information Technology" Volume 9, Special Issue No. 2, pp. 100-106.

- Megat S'adan, Megat Ahmad Haziq & Pampouchidou, Anastasia & Meriaudeau, Fabrice. (2018). Deep Learning Techniques for Depression Assessment. 10.1109/ICIAS.2018.8540634.

- Sumathi, Ms & B, Dr. (2016). Prediction of Mental Health Problems Among Children Using Machine Learning Techniques. International Journal of Advanced Computer Science and Applications. 7. 10.14569/IJACSA.2016.070176.

- M. Deshpande and V. Rao, "Depression detection using emotion artificial intelligence," 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, 2017, pp. 858-862.

- **https://ieeexplore.ieee.org/document/958880**