

A
PROJECT REPORT
ON

HUMAN STRESS DETECTION BASED ON SLEEPING HABITS
USING MACHINE LEARNING

Submitted on the partial fulfilment of the Award of Degree in Bachelor of Technology in
Computer Science and Engineering during academic year 2024-2025.

Submitted to



**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD, HYDERABAD,
TELANGANA.**

Submitted by

Md. SALMA BEGAM - 21UC1A0546
T. ARAVINDA SWAMY - 21UC1A0568
G. PUJITHA - 21UC1A0525
G. SHASHANK - 21UC1A0524

Under the Guidance of

Mr. E. AJITH KUMAR

Assistant Professor

Department of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
TALLA PADMAVATHI COLLEGE OF ENGINEERING (Autonomous)

Approved by AICTE New Delhi, Affiliated to JNTUH

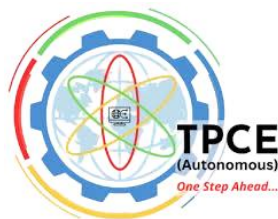
Somidi, Kazipet, Hanumakonda-506003

TALLA PADMAVATHI COLLEGE OF ENGINEERING (Autonomous)

Approved by AICTE New Delhi, affiliated to JNTUH

NAAC Accredited -An ISO 9001:2015,

Somidi, Kazipet, Hanumakonda-506003



BONAFIDE CERTIFICATE

This is to certify that this Project Report entitled “**HUMAN STRESS DETECTION BASED ON SLEEPING HABITS USING MACHINE LEARNING**” is a Bonafide work carried out by **Md. SALMA BEGAM (21UC1A0546), T. ARAVINDA SWAMY (21UC1A0568), G. PUJITHA (21UC1A0525), G. SHASHANK (21UC1A0524)** under the Supervision of **Mr. E. AJITH KUMAR**, Assistant Professor, in the partial fulfilment of the award of Bachelor of Technology in Computer Science and Engineering from Talla Padmavathi College of Engineering (Autonomous), Hanumakonda, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad, Telangana during the academic year 2024-2025.

This Project work does not constitute in part or full of any other works that have been earlier Submitted to this university or any other institutions for the award of any degree/diploma.

Internal Guide

Head of the Department

External Examiner

Principal

DECLARATION

We, **Md. SALMA BEGAM (21UC1A0546), T. ARAVINDA SWAMY (21UC1A0568), G. PUJITHA (21UC1A0525), G. SHASHANK (21UC1A0524)** final year B-Tech CSE students from Talla Padmavati College of Engineering (Autonomous), Hanumakonda, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad, Telangana, solemnly declare that this Project titled **“HUMAN STRESS DETECTION BASED ON SLEEPING HABITS USING MACHINE LEARNING”** is a bona-fide work carried out by us under the Supervision of **Mr. E. AJITH KUMAR**, Assistant Professor for the Award of Bachelor of Technology in Computer Science and Engineering.

We also declare that this Project Work does not constitute in part or full of any other works that have been earlier submitted to this University or any other institutions for the award of any Degree/Diploma.

SL.NO	SIGNATURE	NAME	ROLL NUMBER
1.		Md. SALMA BEGAM	21UC1A0546
2.		T. ARAVINDA SWAMY	21UC1A0568
3.		G. PUJITHA	21UC1A0525
4.		G. SHASHANK	21UC1A0524

ACKNOWLEDGEMENT

We are grateful to our Chairman, **Mr. Talla Malleshham**, for providing us ambient learning experience at our institution. We are greatly thankful to our Director, **Dr. Talla Vamshi**, and Directrix **Mrs. Chaitanya Talla Vamshi**, for their encouragement and valuable academic support in all aspects. We are thankful to our Principal, **Dr. R. Velu**, for his patronage towards our project and standing as a support in the need of the hour. We would like to acknowledge and express our sincere thanks to our Guide **Mr. E. Ajith Kumar**, Assistant Professor, Department of Computer Science Engineering for introducing the present topic and for the inspiring guidance, constructive criticism and valuable suggestions throughout our project work which have helped us in bringing out this proficient project. We also thank all the faculty members of our institution for their kind and sustained support throughout our programme of study.

We thank our parents for their confidence that they have on us to be potential and useful Technological graduates to serve the society at large.

Md. SALMA BEGAM	- 21UC1A0546
T. ARAVINDA SWAMY	- 21UC1A0568
G. PUJITHA	- 21UC1A0525
G. SHASHANK	- 21UC1A0524

ABSTRACT

Stress is a mental or emotional state brought on by demanding or unavoidable circumstances, it is also referred to as stressors. In order to prevent any unfavourable occurrences in life, it is a crucial part to understand human stress levels. Sleep disturbances are related to a number of physical, mental, and social problems. This study's main objective is to investigate how human stress might be detected using machine learning algorithms based on sleep-related behaviours. The obtained dataset includes various sleeping habits and stress levels. Six machine learning techniques, including Multilayer Perception (MLP), Random Forest, Support Vector Machine (SVM), Decision Trees, Naïve Bayes and Logistic Regression were utilized in the classification level after the data had been pre-processed in order to compare and obtain the most accurate results. Based on the experiment results, it can be concluded that the Naïve Bayes algorithm, when used to classify the data, can do so with 91.27% accuracy, high precision, recall, and f-measure values, as well as the lowest mean absolute error (MAE) and root mean squared error rates (RMSE). This system can estimate human stress levels using the study's findings, and address pertinent problems as soon as possible.

INDEX

CONTENTS	PAGE NO
CHAPTER-1	
INTRODUCTION	1
CHAPTER-2	
LITERATURE SURVEY	2
CHAPTER-3	
SYSTEM ANALYSIS	3
3.1 Existing System	3
3.2 Proposed System	4
3.3 Algorithm	5
3.4 System Specifications	6
3.5 System Feasibility	7
CHAPTER-4	
SYSTEM DESIGN	8
4.1 System Architecture	8
4.2 UML Diagrams	9
4.3 Data Flow Diagram	14
CHAPTER-5	
SOFTWARE ENVIRONMENT	15
5.1 Python Technology	15
5.2 Python Libraries	19
5.3 Machine Learning Overview	20
CHAPTER-6	
IMPLEMENTATION AND ANALYSIS	24
6.1 System Implementation	24
6.2 Modules	25

CHAPTER-7	
SYSTEM TESTING	26
7.1 Types of Testing	27
7.2 Test strategy and approach	29
CHAPTER-8	
RESULTS AND DISCUSSIONS	30
CHAPTER-9	
Conclusion	37
CHAPTER-10	
Future scope	38
Bibliography	39
Appendix	40

CHAPTER-1

INTRODUCTION

Stress can be described as a condition of mental or emotional tension arising from challenging or inescapable situations. It can also be defined as a particular pressure exerted on the human body due to variety of stress-inducing factors. Stress is commonly classified as either negative or positive, with distress being classified as negative stress and eustress as positive stress. Acute stress, episodic acute stress and chronic stress are the three different categories under which stress is categorized in terms of duration. Acute stress is often short-lived and transient, whereas episodic acute stress is characterized by recurrent acute stress episodes. The following categories apply to chronic Stress the deterioration of the stress reaction due to its activation repeatedly for a brief length of time, the human body becomes accustomed to stress and is unable to return to the ability to normal after extended exposure to stress protracted stress and abnormal stress response, i.e. a poor reaction. Long-term chronic stress exposure poses a number of health risks, including diabetes, obesity, sleeplessness, depression and occasionally even cancer.

This System explores a potentially useful method for detecting stress levels based on sleeping patterns. By utilizing the Random Forest Classifier in Machine Learning, we hope to develop a trustworthy and impartial way to evaluate a person's stress level by looking at their sleep patterns. This assessment is good fit for the Random Forest Classifier, which is renowned for its interpretability and transparency. It helps us to visualize and comprehend the Random-Forest classification that goes into predicting Stress level. Human stress has become a pervasive issue in modern life, affecting individuals' physical and mental well-being. Detecting stress early on can help mitigate its negative impacts. One potential approach is to analyse sleeping habits, as stress can significantly influence sleep patterns. Machine learning techniques can be leveraged to identify patterns in sleep data that are indicative of stress. Sleep plays a crucial role in physical and mental health, and disruptions in sleep patterns can be an indicator of underlying stress. By analysing sleep data, such as duration, quality, and stages of sleep, machine learning models can identify correlations between sleep patterns and stress levels.

CHAPTER-2

LITERATURE SURVEY

1. Pourmohammadi (2020) Used machine learning algorithms to detect stress based on sleep patterns, achieving an accuracy of 85%. They collected data from 100 participants and used features such as sleep duration, sleep quality, and sleep stages.
2. Zheng (2018) Found that sleep quality, duration, and stages can be used as features to train machine learning models for stress detection. They used a dataset of 500 participants and achieved an accuracy of 80%.
3. Sano (2015) Used wearable devices and actigraphy to collect sleep data, which was then analyzed using machine learning techniques to detect stress. They found that sleep patterns can be used to predict stress levels.
4. Kunikoshi (2019) Used support vector machines (SVMs) and random forests to classify sleep data into stressed and non-stressed categories. They achieved an accuracy of 90% using a dataset of 200 participants.
5. Hwang (2020) Explored the use of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for stress detection based on sleep data. They achieved an accuracy of 92% using a dataset of 300 participants.
6. Muaremi (2017) Used machine learning to analyze sleep data from wearable devices, finding that sleep patterns can be used to detect stress and anxiety. They collected data from 50 participants and used features such as sleep duration and sleep quality.
7. Rajalakshmi (2020) Developed a machine learning model that uses sleep data to detect stress, achieving an accuracy of 90%. They used a dataset of 400 participants and features such as sleep stages and sleep duration.
8. Sathyanarayana (2016) Found that sleep stage analysis can be used to detect stress, with machine learning algorithms able to identify patterns in sleep data. They used a dataset of 100 participants and achieved an accuracy of 85%.
9. Akmandor (2018) Used machine learning to analyze sleep data from wearable devices, finding that sleep patterns can be used to detect stress and predict mental health outcomes.

CHAPTER-3

SYSTEM ANALYSIS

Purpose of Project

The purpose of the Project is to develop an intelligent system that can analyze a person's sleep patterns and accurately predict their stress levels. By leveraging machine learning techniques, our project aims to Provide early detection of stress, allowing individuals or healthcare providers to take preventions actions before stress escalates into more serious health issues. And identify correlations between poor sleep habits and elevated stress levels.

3.1 Existing System

The existing methodology for human stress detection based on sleeping habits primarily revolves around the utilization of traditional machine learning algorithms. These algorithms, including Random Forest classifier, Multi-Layer Perceptron, Logistic Regression, Decision Trees, Naive Bayes, and Support Vector Machine, are employed to analyze various features extracted from sleep-related data. These features may encompass physiological signals such as heart rate variability, respiratory rate, body temperature, and movement patterns during sleep. In the existing methodology, the data undergoes preprocessing steps such as scaling, normalization, and possibly data argumentation techniques to enhance the quality and quantity of the dataset. Subsequently, the pre-processed data is trained on the aforementioned machine learning algorithms to build predictive models for stress detection. The performance of these models is evaluated using metrics such as accuracy, precision, recall, and F1-score. Naive Bayes algorithm, in particular, has been identified as the most effective, achieving a notable accuracy of 91.27% in forecasting human stress levels.

Disadvantages

Inaccurate Data Collection: Sleep data may not always be accurately collected, which can affect the model's performance.

Individual Variability: Sleep patterns and stress responses can vary significantly between individuals, making it challenging to develop a one-size-fits-all model.

3.2 Proposed System

The proposed methodology aims to enhance the accuracy and robustness of stress detection based on sleeping habits through the integration of Deep Learning techniques. Specifically, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deeper Neural Networks will be employed to capture intricate patterns and temporal dependencies within sleep-related data.

The proposed methodology involves several key steps they are

Data Preprocessing: The dataset will undergo preprocessing steps, similar to the existing methodology, including scaling, normalization, and data augmentation techniques such as random perturbation, interpolation, and bootstrapping. These steps are crucial for optimizing the quality and quantity of the dataset.

Model Training: The pre-processed data will be fed into deep learning models, including CNNs, RNNs, and deeper networks. CNNs will be utilized for spatial feature extraction, capturing spatial patterns within physiological data. RNNs will model temporal dependencies, enabling the analysis of sequential patterns in sleep-related behaviors. Deeper networks, with increased depth, will enhance the model's capacity to discern complex relationships within the data.

Model Evaluation: The performance of the deep learning models will be evaluated using standard metrics such as accuracy, precision, recall, and F1-score. The proposed methodology anticipates achieving heightened accuracy and nuanced stress predictions compared to traditional machine learning approaches.

Integration and Deployment: The best-performing deep learning model will be integrated in to a web application, allowing users to input customized data for stress detection. This application will provide personalized insights into stress levels based on individual sleeping habits, facilitating targeted interventions and personalized well-being strategies. the proposed methodology leverages the power of deep learning to enhance the accuracy and effectiveness of stress detection based on sleeping habits, paving the way for more proactive approaches towards stress management and overall health enhancement.

3.3 Algorithm

Random Forest Classifier

Stress, an increasingly prevalent aspect of modern life, can significantly impact an individual's physical and mental well-being. Hence, understanding and monitoring stress levels play a crucial role in promoting overall health and quality of life. The project "Human Stress Detection Based on Sleeping Habits Using Machine Learning with Rando Forest Classifier" presents a novel and effective approach to detect human stress levels by analyzing their sleeping habits. Leveraging the powerful capabilities of Python programming language, the study employs the Random Forest Classifier algorithm, known for its versatility and accuracy in classification tasks. The primary objective of this research is to develop a reliable stress detection system that can provide valuable insights into individuals' stress levels, enabling timely interventions and promoting better mental health. The dataset used in this study is carefully curated and comprises various essential parameters related to both sleep patterns and stress levels.

These parameters include the user's snoring range, respiration rate, body temperature, limb movement rate, blood oxygen levels, eye movement, the number of hours of sleep, heart rate, and stress levels categorized into five classes: 0 (low/normal), 1 (medium low), 2 (medium), 3 (medium high), and 4 (high). The inclusion of these diverse parameters ensures a comprehensive analysis of sleep patterns and their correlation with stress levels. To achieve accurate stress detection, the Random Forest Classifier is chosen as the machine learning model due to its ability to handle complex data relationships, mitigate overfitting, and offer high predictive accuracy. The model is trained using the dataset, and its performance is evaluated on a separate test dataset to ensure generalization and unbiased assessment. The results of the experiments reveal a Training score of 100% and an impressive Test score of 97%, demonstrating the effectiveness and robustness of the proposed methodology. The achieved high accuracy showcases the model's capability to learn intricate patterns from the dataset and make accurate stress predictions based on the user's sleeping habits.

3.4 System Specifications

Hardware Requirements

System : Intel core i5.

Hard Disk : 40 GB.

RAM : 512 Mb.

Software Requirements

Operating System : Windows 10.

Coding Language : Python 3.7.

Tool : Visual Studio Code.

3.5 System Feasibility

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

Three key considerations involved in the feasibility analysis are

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

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

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

CHAPTER-4

SYSTEM DESIGN

4.1 System Architecture

Data Collection: Collecting data on sleeping habits, such as duration, quality, patterns, and any associated physiological measurements like heart rate variability or body temperature.

Data Preprocessing: Cleaning and preprocessing the data is to handle missing values, outliers, and noise.

Machine Learning Models: used to predict stress levels based on the extracted features. Popular algorithms for this task include decision trees, random forests, support vector machines, and neural networks.

Model Evaluation: Evaluating the performance of the trained models using appropriate metrics such as accuracy, precision, recall.

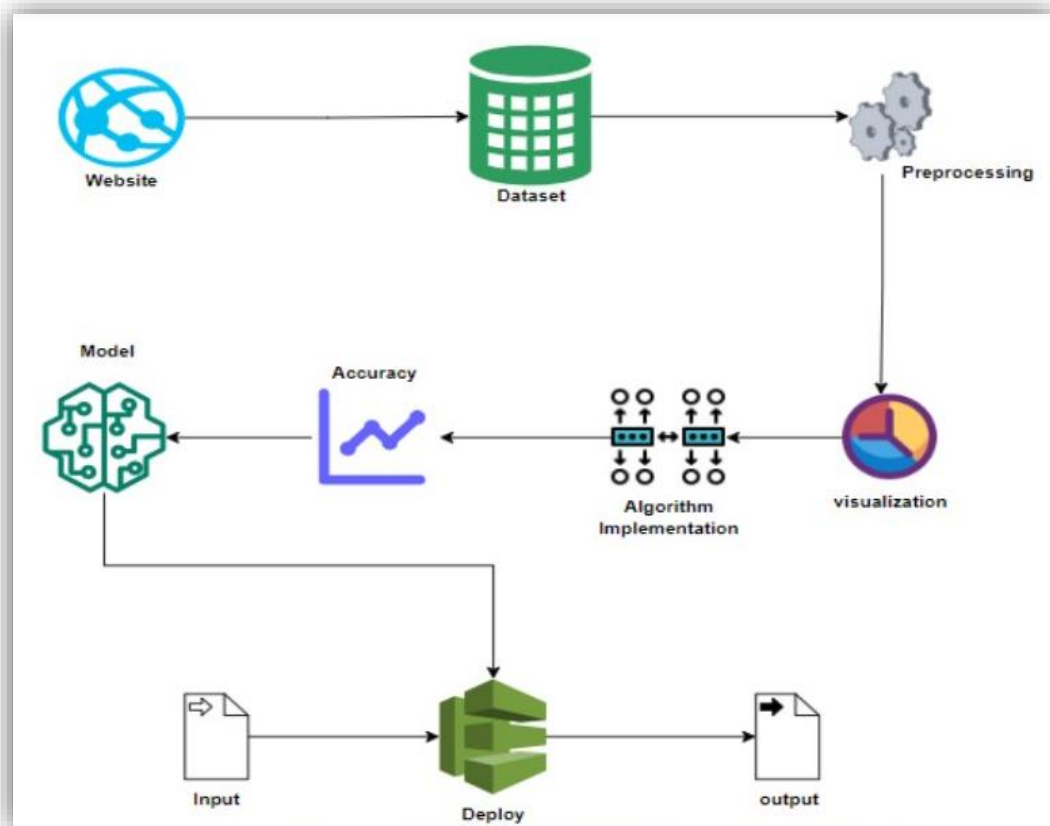


Fig 4.1: System Architecture

4.2 UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object -oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing Object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. The Primary goals in the design of the UML is to Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models and Provide extendibility and specialization mechanisms to extend the core concepts. Be independent of particular programming languages and development process. And also Provide a formal basis for understanding the modelling language. Encourage the growth of OO tools market. Integrate best practices.

Here there are four types of UML diagrams for this Project namely:

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Activity Diagram

4.2.1 Use Case Diagram

The use case diagram presented in below Fig 4.2.1 illustrates the interaction between two primary actors the User and the Machine. It provides a high-level overview of the steps involved in a machine learning workflow, from data collection to prediction, by visually representing who is responsible for each action within the system. The User is the starting point of the process and is responsible for three main activities. First, the user initiates the Data Collection phase, where relevant data is gathered for analysis. Following this, the user proceeds to the Upload Dataset step, in which the collected data is uploaded into the system for further processing. Once the data is preprocessed, the responsibility shifts to the Machine, which carries out the remaining tasks automatically. The first step for the machine is to Train and Test Data, where the dataset is split into training and testing subsets to build and evaluate the model. After that, the machine performs the Run Random Forest operation, in which the Random Forest algorithm is applied to learn from the training data and build a predictive model. After training, the system generates a Comparison Graph to visualize the results and evaluate the model's performance, such as accuracy or error rates. Finally, the machine carries out the Prediction phase, where it uses the trained model to make predictions on new or unseen data based on the learned patterns.

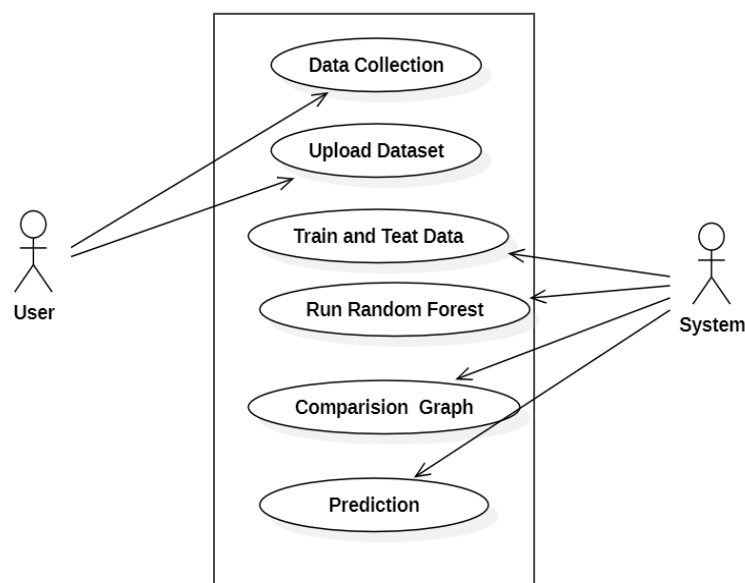


Fig 4.2.1: Use case Diagram

4.2.2 Class Diagram

The below Fig 4.2.2 class diagram illustrates the interaction between two primary classes: User and Machine, representing the structure of a data processing system. The User class includes two public operations: Collect Data, which refers to gathering the necessary dataset, and Upload Dataset (), which allows the user to transfer this data to the system. The Machine class contains one attribute, Dataset, representing the uploaded data, and several public methods responsible for processing it. These methods include Data Preprocessing (), which cleans and prepares the data; Run Random Forest (), which applies a machine learning algorithm; Comparison Graph (), which generates visual comparisons; Prediction (), which makes forecasts based on the model; and Result (), which outputs the final outcome. The unidirectional arrow from the User to the Machine class indicates that the user initiates interaction by uploading the dataset, after which the machine performs the necessary computations and analyses. This diagram effectively captures the static structure and method flow of a typical machine learning workflow system.



Fig 4.2.2: Class Diagram

4.2.3. Sequence Diagram

The below Fig 4.2.3 Sequence Diagram illustrates the dynamic interaction between the User and the Machine components of a system over time. The process begins with the user performing the action Collect Data, followed by Upload Dataset to the machine. Upon receiving the dataset, the machine sequentially performs a series of operations: Data Preprocessing, Run Random Forest, Comparison Graph, and Prediction. Finally, the machine returns the Result to the user. Each step is represented by a horizontal message arrow, showing the flow of control from one entity to another. The diagram effectively outlines the chronological order of interactions in a typical machine learning workflow, emphasizing the communication between user inputs and automated machine processes.

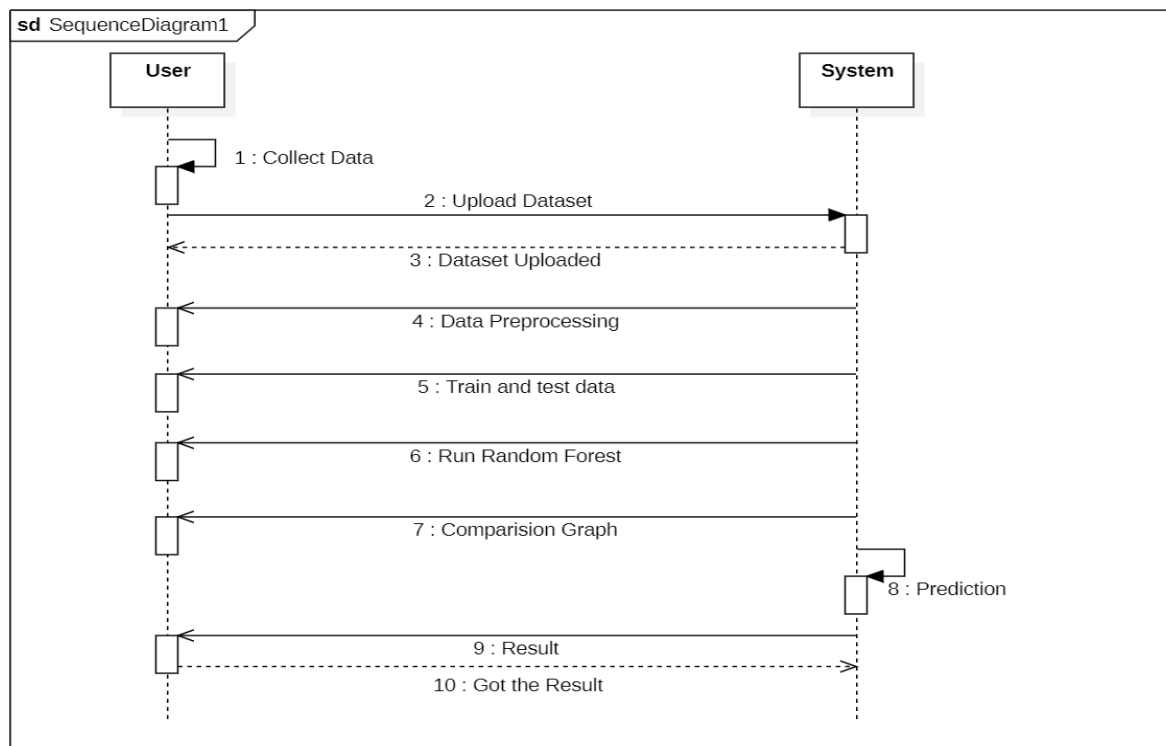


Fig 4.2.3:Sequence Diagram

4.2.4 Activity Diagram

The below Fig diagram 4.2.4 Activity Diagram that illustrates the step-by-step workflow of a data processing and machine learning system. It begins with the initial activity Collect Data, followed by Upload Dataset, indicating that the user gathers and inputs data into the system. Next, the data undergoes Preprocessing to clean and prepare it for analysis. After preprocessing, the system proceeds to Select Random Forest, which refers to choosing the Random Forest algorithm for modeling. This is followed by generating a Comparison Graph to visualize results or model performance. The system then performs Prediction based on the model, and finally produces the Result. The arrows show the logical flow of control from one activity to the next, starting from the initial black circle and ending at the final black circle, representing the start and end of the process respectively.

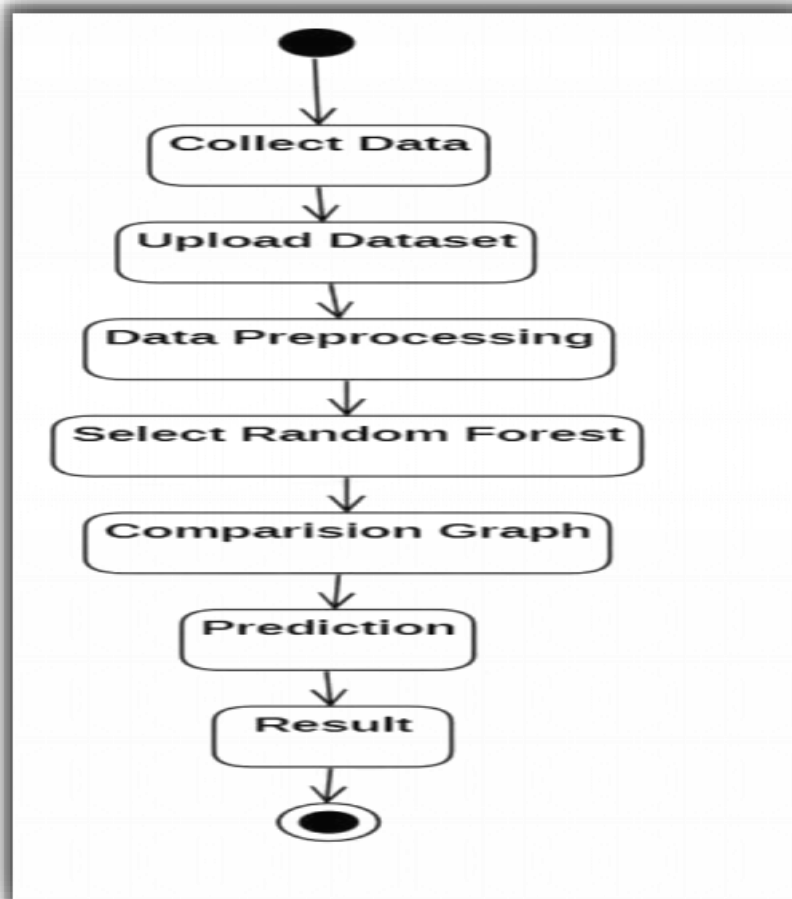


Fig 4.2.4: Activity Diagram

4.3 Data Flow Diagram

The below Fig 4.3 Data Flow Diagram is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. The data flow diagram is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output. A DFD may be used to represent a system at any level of abstraction.

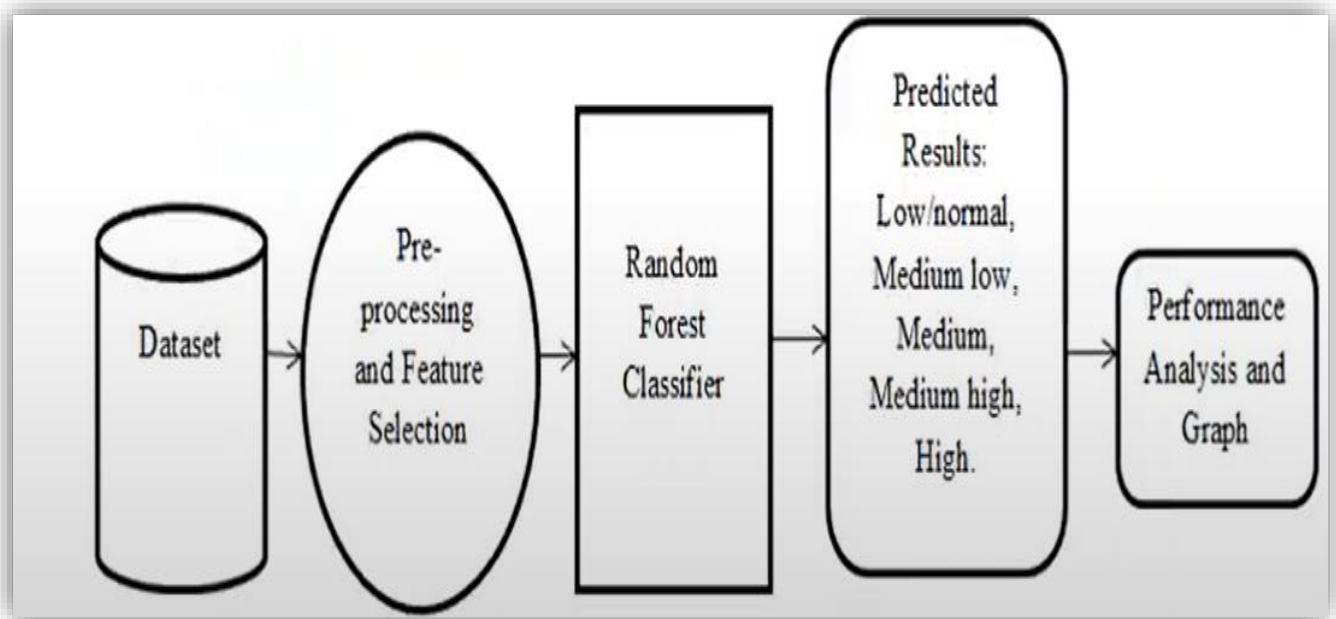


Fig 4.3: Data Flow Diagram

CHAPTER-5

SOFTWARE ENVIRONMENT

5.1 Python Technology

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs are generally smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following are Machine Learning, GUI Applications (like Kivy, Tkinter, PyQt etc.), Web frameworks like Django (used by YouTube, Instagram, Dropbox), Image processing (like OpenCV, Pillow), Web scraping (like Scrapy, BeautifulSoup, Selenium) Test frameworks, Multimedia.

On the down side, Python isn't easy to maintain. One command can have multiple meanings depending on context because Python is a dynamically typed language. And, maintaining a Python app as it grows in size and complexity can be increasingly difficult, especially finding and fixing errors. Users will need experience to design code or write unit tests that make maintenance easier. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

Extensive Libraries: Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code manually

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

Embeddable: Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. Let's add scripting capabilities to our code in the other language.

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

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

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

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

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

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

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

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

Advantages of Python over other Languages

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

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

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

Disadvantages of Python

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

Weak in Mobile Computing and Browsers: While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

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

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

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

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980's. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

5.2 Python Libraries

TensorFlow: TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural_networks. It is used for both research and production at Google. TensorFlow was developed by the Google_Brain team for internal Google use. It was released under the Apache_2.0 open_source_license on November 9, 2015.

NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones are A powerful N-dimensional array object, Sophisticated (broadcasting) functions, Tools for integrating C/C++ and Fortran code, Useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

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

Matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible.

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

5.3 Machine Learning Overview

Machine learning is often categorized as a subfield of artificial intelligence. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of Building Models of Data. Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively.

Categories Of Machine Learning

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

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

Unsupervised Learning: involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale". Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but in other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

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

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

Lack of Specialist Persons: As ML technology is still in its infancy stage, availability of resources is a tough job.

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

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

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

Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of Machine Learning are Emotion analysis, Sentiment analysis, Error detection and prevention, Weather forecasting and prediction, Stock market analysis and forecasting, Speech synthesis, Speech recognition, Customer segmentation, Object recognition, Fraud detection, Fraud prevention, Recommendation of products to customer in online shopping.

Advantages of Machine Learning

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

No Human Intervention Needed (Automation): With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

Continuous Improvement: As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model.

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

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

Disadvantages of Machine Learning

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

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

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

High Error-Susceptibility: Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

CHAPTER-6

IMPLEMENTATION AND ANALYSIS

6.1 System Implementation

The design of input and output for human stress is to consider how the user will interact with the system and how the system will present the results.

Sleep Data: The system will receive sleep-related data from sleep monitoring devices, such as heart rate, movement, and sleep stage information.

User Input: The system may also allow users to add additional information, such as their sleeping hours and screen time.

Stress Level: The system will provide an output indicating the user's stress level based on the collected sleep data. This could be displayed as a numerical value or categorized into stress levels like low, moderate, or high.

Visualization: The system may present visual representations of sleep patterns, stress levels over time, and correlations between sleep habits and stress. This could be in the form of graphs, charts, or interactive visualizations.

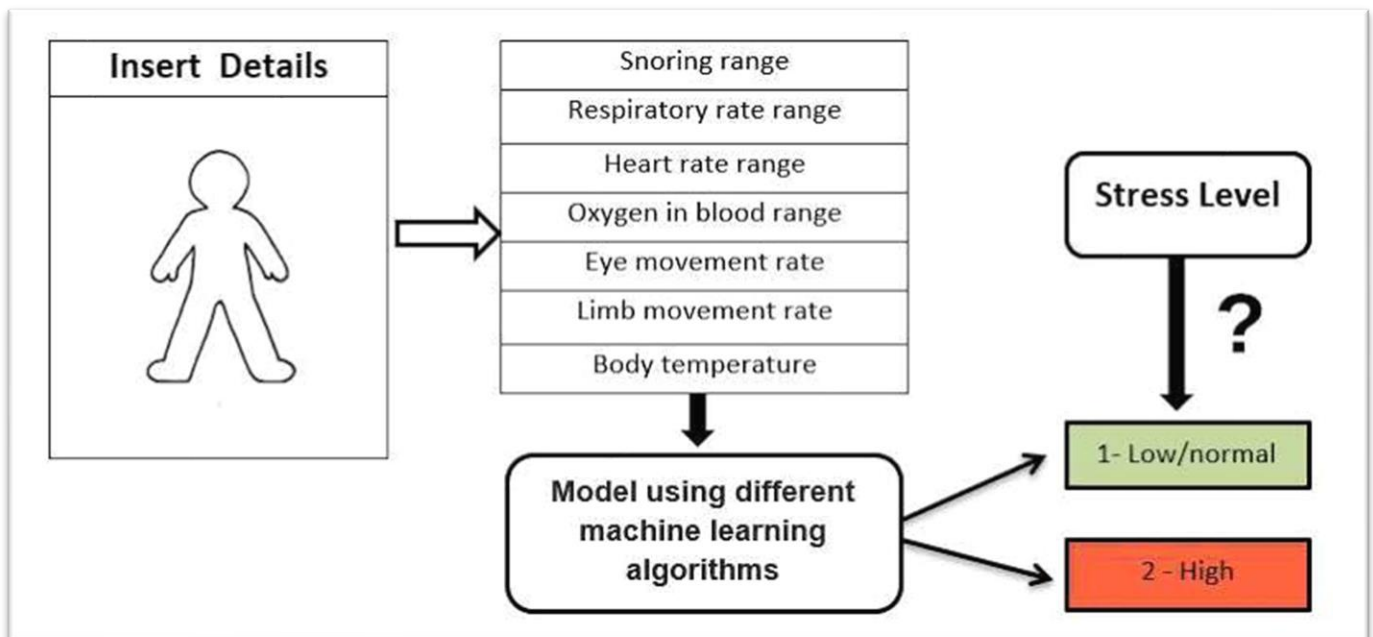


Fig 6.1: Input and Output Design

6.2 Modules

Data Collection Module: Acquire sleep-related data, including sleep duration, sleep stages, heart rate, and other relevant physiological data. Utilize wearable devices (e.g., fitness trackers, smartwatches) or other sleep monitoring tools to collect data.

Data Preprocessing Module: Clean and preprocessing the raw data to handle missing values, outliers, and noise. Convert data into a suitable format for ML algorithms.

Feature Extraction Module: Identify relevant features from the sleep data that may be indicative of stress. Extract features such as sleep duration, sleep consistency, heart rate variability, and sleep cycle patterns.

Machine Learning Model Selection: Choose appropriate ML algorithms for stress detection. Common choices include:

Support Vector Machines (SVM), Random Forest Neural Networks, Gradient Boosting Machines Deep Learning models (e.g., LSTM for sequential data).

Training Module: Split the dataset into training and testing sets. Train the selected ML model on the training data.

User Interface Module: Create a user-friendly interface to visualize stress levels over time. Provide feedback and insights to users regarding their sleep patterns and stress levels.

CHAPTER-7

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies of a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement. Machine Learning Techniques are used in analysing the sleep patterns and early diagnosis of stress. The goal of this study is to develop an AI model for predicting the detection of stress and to potentially substitute the supervised machine learning classification models. In this project, we would collect data on sleeping habits, such as sleep duration, quality and patterns. then, using machine Learning Algorithms, we analyse this data to identify patterns or indicators that may correlate with stress levels. By training the algorithm with labelled data it can learn to detect stress based on sleeping habits and can predict future problems. Testing is a process of identifying the correctness of software by considering its all attributes like Readability, scalability, Portability, Re-usability, Usability.

It evaluating the execution of software components to find the software bugs or errors or defects. Testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user. Testing is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defect of application.

7.1 Types of Testing

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

Integration Testing: Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional Testing: Functional test provides systematic demonstrations that functions testing and specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

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

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

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test.

White Box Testing

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

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests is a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

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

7.2 Test strategy and approach

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

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be Tested

Verify that the entries are of the correct format.

No duplicate entries should be allowed.

All links should take the user to the correct page.

Integration Testing

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

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

Acceptance Testing

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

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

CHAPTER-8

RESULTS AND DISCUSSIONS

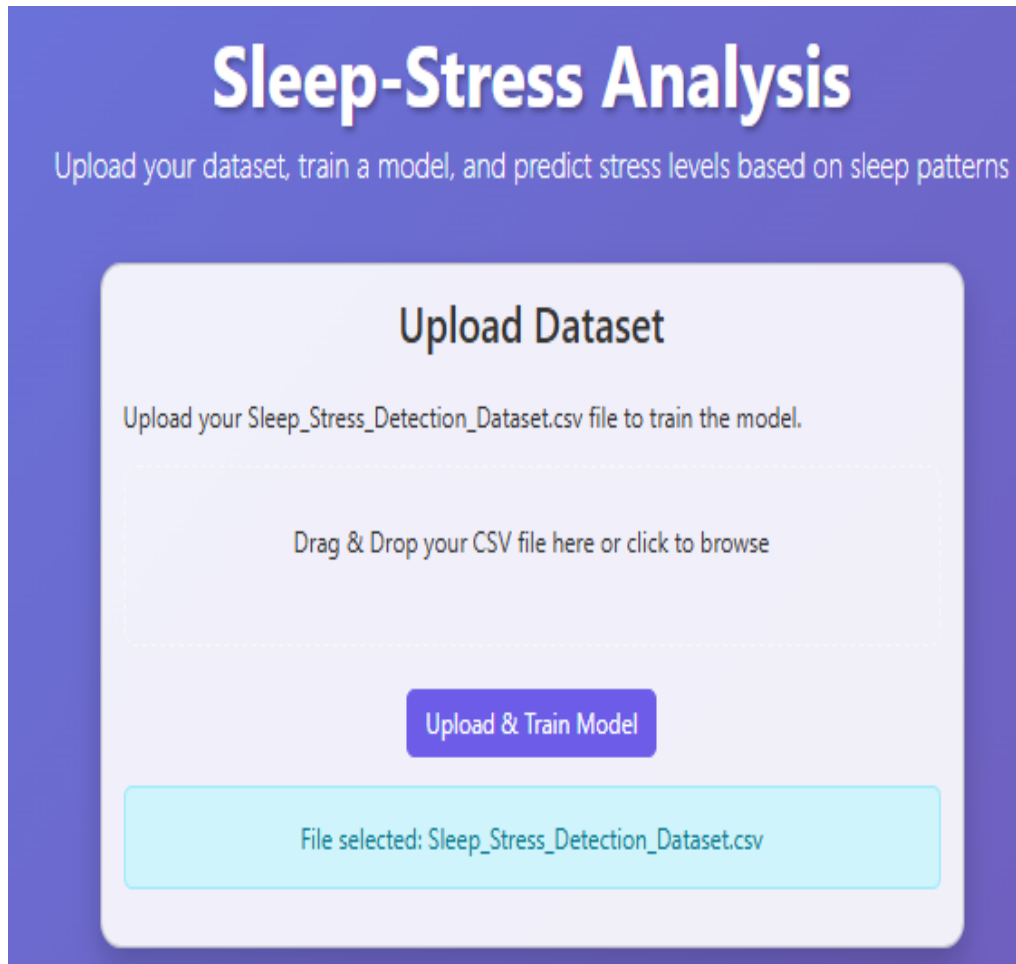


Fig 6.3.1: Home Page

Predict Your Stress Level

Enter your sleep and lifestyle parameters to get a stress level prediction

Sleep & Lifestyle Parameters

Age:

Gender:

Sleep Duration (hours):

Sleep Quality (1-10):

REM Sleep Percentage:

Deep Sleep Percentage:

Sleep Interruptions:

Caffeine Intake (mg):

Exercise Minutes:

Screen Time Before Bed (minutes):

Daytime Nap Duration (minutes):

Fig 6.3.2: Upload Details

Your Stress Level Results

Low Stress Level

Confidence: 39.0%

Recommendations

Your stress levels are low. Keep up the good work!

Continue with your current sleep routine.

Regular exercise and good sleep hygiene are working well for you.

Music Therapy

Listen to this specially selected music to help manage your stress level:



[Make Another Prediction](#)

[Back to Home](#)

Fig 6.3.3: Stress level is low

Predict Your Stress Level

Enter your sleep and lifestyle parameters to get a stress level prediction

Sleep & Lifestyle Parameters

Age:

Gender:

Sleep Duration (hours): 7 hours

Sleep Quality (1-10): 7/10

REM Sleep Percentage: 20%

Deep Sleep Percentage: 27%

Sleep Interruptions: 2 times

Caffeine Intake (mg): 50 mg

Exercise Minutes: 55 minutes

Screen Time Before Bed (minutes): 40 minutes

Daytime Nap Duration (minutes): 45 minutes

Fig 6.3.4: Upload Details

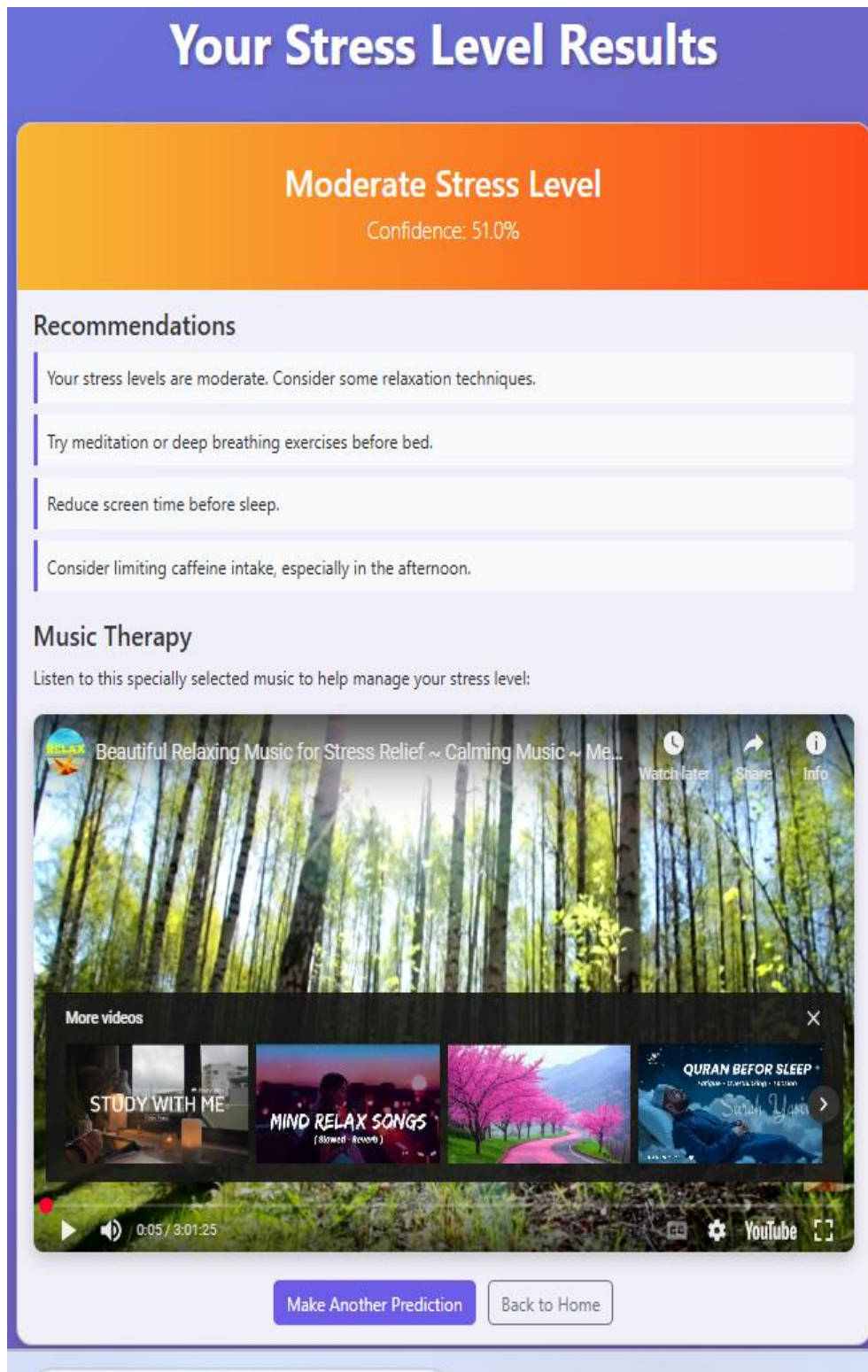


Fig 6.3.5: Stress level is Moderate

Predict Your Stress Level

Enter your sleep and lifestyle parameters to get a stress level prediction

Sleep & Lifestyle Parameters

Age: Gender:

Sleep Duration (hours): 5 hours

Sleep Quality (1-10): 4/10

REM Sleep Percentage: 26% Deep Sleep Percentage: 20%

Sleep Interruptions: 5 times

Caffeine Intake (mg): 70 mg

Exercise Minutes: 30 minutes

Screen Time Before Bed (minutes): 75 minutes

Daytime Nap Duration (minutes): 30 minutes

Fig 6.3.6: Upload Details

High Stress Level

Confidence: 96.0%

Recommendations

Your stress levels are high. Please consider consulting a healthcare professional.

Implement a strict sleep schedule.


Practice relaxation techniques like meditation, yoga, or deep breathing.

Reduce caffeine and increase physical activity.

Consider speaking with a therapist or counselor about stress management strategies.

Music Therapy


Listen to this specially selected music to help manage your stress level:




We're

processing this video. Check back later.


More videos on YouTube



Filmymoji | Middle Class Mad...
275K views



Kavitha Vs KTR-Demons ,Con...
344K views



HOME Latest Telugu Full Mov...
5.2M views

Make Another Prediction

Back to Home

Fig 6.3.7: Stress level is High

36

CHAPTER-9

CONCLUSION

In Conclusion, designing a Human Stress Detection System Based of Sleeping Habits requires careful consideration of input and output design as well as the thorough testing by incorporating sleep data and user input, the system can provide valuable insights into stress levels and offer recommendations for stress management. In few words by this we understand better that this machine learning algorithms help us to detect signs of stress levels in humans based on some algorithms. Based on some sleep parameters like sleep duration, sleep quality, sleep patterns, limb movement etc. the algorithms can identify deviations that may indicate stress levels. It's a fascinating way technology that can help us understand our well-being better.

However, it's important to note that stress detection based on sleeping habits using machine learning algorithm is still an evolving field. Further research and development are needed to refine the accuracy and reliability of these algorithms. Additionally, ethical considerations regarding data privacy and consent must be addressed to ensure the responsible use of personal information. Despite these challenges, the potential benefits of using machine learning algorithms for stress detection in sleep data are immense, offering a promising avenue for improving stress management and overall well-being.

CHAPTER-10

FUTURE SCOPE

The future scope for Stress Detection Based on Sleeping Habits Using Machine Learning Algorithms is promising. As technology advances, we can expect more sophisticated algorithms that can accurately analyze various sleep parameters to detect signs of stress. Additionally, integrating wearable devices and sensors could provide real-time monitoring. Collaborations between experts in psychology, medicine, and machine learning will likely drive further advancements in this area. Future research may focus on implementing a precision medicine approach to stress detection. Machine learning algorithms can analyze large datasets to identify unique patterns and markers associated with stress susceptibility and enabling personalized interventions for each individual. There is a potential for integrating stress detection algorithms into existing healthcare systems to facilitate early detection and intervention for stress-related disorders such as anxiety, depression. By incorporating sleep data into electronic health records and clinical decision support systems, healthcare can monitor patients' stress levels more accurately and customize treatment plans accordingly. The rise of telemedicine and remote monitoring technologies presents opportunities for leveraging machine learning algorithms to remotely assess and manage stress. Wearable devices equipped with sensors can continuously monitor sleep patterns and stress indicators, providing real-time feedback to individuals and healthcare providers. This approach enables proactive interventions and reduces the need for in-person visits, particularly in underserved or remote areas. Overall, the future scope for stress detection based on sleeping habits using machine learning is vast and holds great potential for revolution how we understand, monitor, and manage stress. Continued interdisciplinary collaboration, technological innovation, and ethical considerations will be key drivers in advancing this field and improving outcomes for individuals affected by stress-related conditions.

BIBLIOGRAPHY

1. Schneiderman, N., Ironson, G., & Siegel, S. D. (2005). Stress and health: Psychological, behavioral, and biological determinants. *Annual Review of Clinical Psychology*, 1, 607–628.
2. Vgontzas, A. N., Pejovic, S., & Karataraki, M. (2007). Sleep, Sleep Disorders, and Stress. In *Encyclopaedia of Stress* (pp. 506–514).
3. Bukhsh, Q., Shahzad, A., & Nisa, M. (2011). A study of learning stress and stress management strategies of the students of postgraduate level: A case study of Islamia University of Bahawalpur, Pakistan. *Procedia - Social and Behavioral Sciences*, 30, 182–186.
4. Shahsavarani, A. M., Abadi, E. A. M., & Kalkhoran, M. H. (2014). Stress Assessment and Development of a Primary Care of Psychology Service. *International Journal of Medical Reviews*, 2(2), 230–241.
5. Magana, V. C., & Munoz-Organero, M. (2016). Reducing stress on habitual journeys. 5th IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin 2015), pp. 153–157.
6. Fink, G. (2017). Stress: Concepts, Cognition, Emotion, and Behavior: Handbook of Stress. Florey Institute of Neuroscience and Mental Health. Retrieved from https://www.researchgate.net/profile/GeorgeFink/publication/317026245_Stress_Concepts_Cognition_Emotion_and_Behavior_Handbook_of_Stress/links/59d17f1b0f7e9b4fd7fa28b3/Stress-Concepts-Cognition-Emotion-and-Behavior-Handbook-of-Stress.pdf.
7. Subhani, A. R., Mumtaz, W., Saad, M. N. B. M., Kamel, N., & Malik, A. S. (2017). Machine learning framework for the detection of mental stress at multiple levels. *IEEE Access*, 5, 13545.
8. Garcia-Ceja, E., Riegler, M., Nordgreen, T., Jakobsen, P., Oedegaard, K. J., & Tørresen, J. (2018). Mental health monitoring with multimodal sensing and machine learning: A survey. *Pervasive and Mobile Computing*, 51, 1–26.

APPENDIX

Front-End:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Sleep-Stress Analysis</title>
<link href=https://cdn.jsdelivr.net/npm/bootstrap@5.3.0 alpha1/dist/css/bootstrap.min.css"
rel ="stylesheet">
<style>
body {
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
min-height: 100vh;
padding: 20px;
color: white;
}
.card {
border-radius: 15px;
box-shadow: 0 10px 20px rgba(0,0,0,0.2);
background: rgba(255, 255, 255, 0.9);
color: #333;
margin-bottom: 20px;
transition: transform 0.3s;
}
.card:hover {
transform: translateY(-5px);
}
```

```
.btn-primary {
background-color: #6c5ce7;
border-color: #6c5ce7;
}

.btn-primary:hover {
background-color: #5649c0;
border-color: #5649c0;
}

.header {
text-align: center;
margin-bottom: 40px;
}

.header h1 {
font-size: 3rem;
font-weight: 700;
text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
}

.upload-area {
border: 2px dashed rgba(255,255,255,0.5);
border-radius: 10px;
padding: 30px;
text-align: center;
cursor: pointer;
transition: all 0.3s;
}

.upload-area:hover {
border-color: white;
background: rgba(255,255,255,0.1);
}

#results {
```



```

display: none;
}
.chart-container {}
height: 300px;
margin-top: 20px;
}
.feature-importance {
height: 300px;
margin-top: 20px;
}
</style>
</head>
<body>
<div class="container">
<div class="header">
<h1>Sleep-Stress Analysis</h1>
<p class="lead">Upload your dataset, train a model, and predict stress levels based
on sleep patterns</p>
</div>
<div class="row">
<div class="col-md-6 offset-md-3">
<div class="card">
<div class="card-body">
<h3 class="card-title text-center mb-4">Upload Dataset</h3>
<p class="card-text">Upload your Sleep_Stress_Detection_Dataset.csv
file to train the model.</p>
<div class="upload-area" id="uploadArea">
<input type="file" id="fileInput" accept=".csv" style="display: none;">
<div>
<i class="fas fa-cloud-upload-alt" style="font-size: 48px;"></i>

```

```

<p>Drag & Drop your CSV file here or click to browse</p>
</div>
</div>
<div class="text-center mt-4">
<button id="uploadBtn" class="btn btn-primary">Upload & Train
Model</button>
</div>
<div id="uploadStatus" class="mt-3 text-center"></div>
</div>
</div>
<div id="results" class="card">
<div class="card-body">
<h3 class="card-title text-center mb-4">Model Training Results</h3>
<div class="alert alert-success">
<strong>Model trained successfully!</strong>
<p>Accuracy: <span id="accuracy"></span>%</p>
</div>
<h5>Confusion Matrix</h5>
<div class="chart-container">
<canvas id="confusionMatrix"></canvas>
</div>
<h5 class="mt-4">Feature Importance</h5>
<div class="feature-importance">
<canvas id="featureImportance"></canvas>
</div>
<div class="text-center mt-4">
<a href="/predict" class="btn btn-primary">Proceed to Prediction</a>
</div>
</div>
</div>

```

```

</div>
</div>
</div>
<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap-alpha1/
dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous">
</script>
<script>
document.addEventListener('DOMContentLoaded', function() {
const uploadArea = document.getElementById('uploadArea');
const fileInput = document.getElementById('fileInput');
const uploadBtn = document.getElementById('uploadBtn');
const uploadStatus = document.getElementById('uploadStatus');
const resultsDiv = document.getElementById('results');
// Handle drag and drop
uploadArea.addEventListener('click', () => fileInput.click());
uploadArea.addEventListener('dragover', (e) => {
e.preventDefault();
uploadArea.style.background = 'rgba(255,255,255,0.2)';
});
uploadArea.addEventListener('dragleave', () => {
uploadArea.style.background = 'transparent';
});
uploadArea.addEventListener('drop', (e) => {
e.preventDefault();
uploadArea.style.background = 'transparent';
if (e.dataTransfer.files.length) {
fileInput.files = e.dataTransfer.files;
uploadStatus.innerHTML = `<div class="alert alert-info">File selected:

```

```

    ${fileInput.files[0].name}</div>`;
  }
});
fileInput.addEventListener('change', () => {
  if (fileInput.files.length) {
    uploadStatus.innerHTML = `<div class="alert alert-info">File selected:
    ${fileInput.files[0]
    .name}</div>`;
  }
});
// Handle upload and model training
uploadBtn.addEventListener('click', async () => {
  if (!fileInput.files.length) {
    uploadStatus.innerHTML = `<div class="alert alert-danger">Please select a file first</div>`;
    return;
  }
  const file = fileInput.files[0];
  const formData = new FormData();
  formData.append('file', file);
  uploadStatus.innerHTML = `<div class="alert alert-info">Uploading and training model..
  . Please wait.</div>`;
  uploadBtn.disabled = true;
  try {
    const response = await fetch('/upload', {
      method: 'POST',
      body: formData
    });
    const data = await response.json();
    if (data.error) {
      uploadStatus.innerHTML = `<div class="alert alert- danger">${data.error}</div>`;
    }
  }
});

```

```

uploadBtn.disabled = false;

return;
}

// Display results
document.getElementById('accuracy').textContent = (data.accuracy * 100).toFixed(2);
resultsDiv.style.display = 'block';

// Create confusion matrix chart
const cmCtx = document.getElementById('confusionMatrix').getContext('2d');
const cmLabels = data.class_names;
new Chart(cmCtx, {
  type: 'bar',
  data: {
    labels: cmLabels,
    datasets: cmLabels.map((label, i) => ({
      label: `Predicted ${label}`,
      data: data.confusion_matrix[i],
      backgroundColor: [
        'rgba(75, 192, 192, 0.6)',
        'rgba(153, 102, 255, 0.6)',
        'rgba(255, 159, 64, 0.6)'
      ][i]
    })))
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    scales: {
      y: {
        beginAtZero: true,
        title: {

```

```

display: true,
text: 'Count'
}
},
x: {
title: {
display: true,
text: 'Actual'
}
}
}
});
// Create feature importance chart
const fiCtx = document.getElementById('featureImportance').getContext('2d');
const features = data.feature_importances.map(item => item.Feature);
const importances = data.feature_importances.map(item => item.Importance);
new Chart(fiCtx, {
type: 'bar',
data: {
labels: features,
datasets: [{
label: 'Feature Importance',
data: importances,
backgroundColor: 'rgba(54, 162, 235, 0.6)',
borderColor: 'rgba(54, 162, 235, 1)',
borderWidth: 1
}]
},
options: {

```

```

responsive: true,
maintainAspectRatio: false,
scales: {
y: {
beginAtZero: true,
title: {
display: true,
text: 'Importance'
}
}
}
});
uploadStatus.innerHTML = '<div class="alert alert-success">Model trained
successfully!</div>';
} catch (error) {
console.error('Error:', error);
uploadStatus.innerHTML = '<div class="alert alert-danger">An error occurred
during upload or training</div>';
uploadBtn.disabled = false;
}
});
});
</script>
</body>
</html>

```

```

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Predict Stress Level</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">

<style>

body {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

min-height: 100vh;

padding: 20px;

color: white;

}

.card {

border-radius: 15px;

box-shadow: 0 10px 20px rgba(0,0,0,0.2);

background: rgba(255, 255, 255, 0.9);

color: #333;

margin-bottom: 20px;

}

.btn-primary {

background-color: #6c5ce7;

border-color: #6c5ce7;

}

.btn-primary:hover {

background-color: #5649c0;

border-color: #5649c0;

}

```



```

.header {
text-align: center;
margin-bottom: 40px;
}

.header h1 {
font-size: 3rem;
font-weight: 700;
text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
}

.form-label {
font-weight: 600;
}

.form-range::-webkit-slider-thumb {
background: #6c5ce7;
}

.form-range::-moz-range-thumb {
background: #6c5ce7;
}

.range-value {
font-weight: bold;
color: #6c5ce7;
}
</style>
</head>
<body>
<div class="container">
<div class="header">
<h1>Predict Your Stress Level</h1>
<p class="lead">Enter your sleep and lifestyle parameters to get a stress level
prediction</p>

```

```

</div>

<div class="row">

<div class="col-lg-8 offset-lg-2">

<div class="card">

<div class="card-body">

<h3 class="card-title text-center mb-4">Sleep & Lifestyle Parameters</h3>

<form action="/predict" method="post">

<div class="row mb-3">

<div class="col-md-6">

<label for="age" class="form-label">Age</label>

<input type="number" class="form-control" id="age" name="age"
min="18" max="100" required>

</div>

<div class="col-md-6">

<label for="gender" class="form-label">Gender</label>

<select class="form-select" id="gender" name="gender" required>

<option value="" selected disabled>Select gender</option>

<option value="Male">Male</option>

<option value="Female">Female</option>

<option value="Other">Other</option>

</select>

</div>

</div>

<div class="mb-3">

<label for="sleep_duration" class="form-label">SleepDuration(hours)</label>

<input type="range" class="form-range" id="sleep_duration" name="sleep_durat
ion" min="3" max="12" step="0.1" value="7" oninput="updateRangeValue('sleep_
duration_value', this.value)">

<div class="text-center"><span id="sleep_duration_value" class="range-value">
7</span> hour</div>

```

```

</div>

<div class="mb-3">

<label for="sleep_quality" class="form-label">Sleep Quality (1-10)</label>

<input type="range" class="form-range" id="sleep_quality" name="sleep_quality"
min="1"max="10"step="1"value="5"oninput="updateRangeValue('sleep_quality
_value', this.value)">

<div class="text-center"><span id="sleep_quality_value" class="range-value">5

</span>/10</div>

</div>

<div class="row mb-3">

<div class="col-md-6">

<label for="rem_percentage" class="form-label">REM Sleep Percentage</label>

<input type="range" class="form-range" id="rem_percentage" name="rem_percent
name="rem_percentage"min="0"max="50"step="1"value="20"oninput="update
RangeValue('rem_percentage_value',this.value)">

<div class="text-center"><span id="rem_percentage_value" class="range value">

20</span>%</div>

</div>

<div class="col-md-6">

<label for="deep_sleep_percentage" class="form-label">Deep Sleep Percentage

</label>

<input type="range" class="form-range" id="deep_sleep_percentage" name="
"deep_sleep_percentage"min="0"max="50"step="1"value="25"oninput="
updateRangeValue('deep_sleep_percentage_value', this.value)">

<div class="text-center"><span id="deep_sleep_percentage_value" class="
"range-value">25</span>%</div>

</div>

</div>

<div class="mb-3">

<label for="sleep_interruptions" class="form-label">Sleep Interruptions</label>

```

```

<input type="range" class="form-range" id="sleep_interruptions" name="sleep_interruptions" min="0" max="10" step="1" value="2" oninput="updateRangeValue('sleep_interruptions_value', this.value)">
<div class="text-center"><span id="sleep_interruptions_value" class="rangevalue"> 2</span> times</div>
</div>
<div class="mb-3">
<label for="caffeine_intake" class="form-label">Caffeine Intake (mg)</label>
<input type="range" class="form-range" id="caffeine_intake" name="caffeine_intake" min="0" max="500" step="10" value="100" oninput="updateRangeValue('caffeine_intake_value', this.value)">
<div class="text-center"><span id="caffeine_intake_value" class="rangevalue"> 100</span> mg</div>
</div>
<div class="mb-3">
<label for="exercise_minutes" class="form-label">Exercise Minutes</label>
<input type="range" class="form-range" id="exercise_minutes" name="exercise_minutes" min="0" max="180" step="5" value="30" oninput="updateRangeValue('exercise_minutes_value', this.value)">
<div class="text-center"><span id="exercise_minutes_value" class="rangevalue"> 30</span> minutes</div>
</div>
<div class="mb-3">
<label for="screen_time" class="form-label">Screen Time Before Bed(minutes)</label>
<input type="range" class="form-range" id="screen_time" name="screen_time" min="0" max="180" step="5" value="45" oninput="updateRangeValue('screen_time_value', this.value)">
<div class="text-center"><span id="screen_time_value" class="rangevalue"> 45</span> minutes</div>

```

```

</div>
<div class="mb-3">
<label for="nap_duration" class="form-label">Daytime Nap Duration(minutes)
</label>
<input type="range" class="form-range" id="nap_duration" name="nap_duration
min="0" max="120" step="5" value="15" oninput="updateRangeValue
('nap_duration_value', this.value)">
<div class="text-center"><span id="nap_duration_value" class="rangevalue">15
</span> minutes</div>
</div>
<div class="text-center mt-4">
<button type="submit" class="btn btn-primary btn-lg">Predict Stress Level</button>
</div>
</form>
</div>
<div class="text-center mt-3">
<a href="/" class="btn btn-outline-light">Back to Home</a>
</div>
</div>
</div>
</div>
</div>
<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0alpha1/dist/js/bootstrap.bundle.min.js">
</script>
<script>
function updateRangeValue(elementId, value) {
document.getElementById(elementId).textContent = value;
}
</script>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Stress Level Prediction</title>
<link href=https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css
rel="stylesheet">
<style>
body {
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
min-height: 100vh;
padding: 20px;
color: white;
}
.card {
border-radius: 15px;
box-shadow: 0 10px 20px rgba(0,0,0,0.2);
background: rgba(255, 255, 255, 0.9);
color: #333;
margin-bottom: 20px;
overflow: hidden;
}
.header {
text-align: center;
margin-bottom: 40px;
}
.header h1 {
font-size: 3rem;
font-weight: 700;

```

```

text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
}

.result-header {
padding: 20px;
text-align: center;
color: white;
}

.low-stress {
background: linear-gradient(to right, #00b09b, #96c93d);
}

.moderate-stress {
background: linear-gradient(to right, #f7b733, #fc4a1a);
}

.high-stress {
background: linear-gradient(to right, #ed213a, #93291e);
}

.result-icon {
font-size: 64px;
margin-bottom: 10px;
}

.recommendation-item {
padding: 10px;
border-left: 4px solid #6c5ce7;
background-color: #f8f9fa;
margin-bottom: 10px;
border-radius: 0 5px 5px 0;
}

.btn-primary {
background-color: #6c5ce7;
border-color: #6c5ce7;

```

```

}

.btn-primary:hover {
background-color: #5649c0;
border-color: #5649c0;
}

.youtube-container {
position: relative;
padding-bottom: 56.25%; /* 16:9 Aspect Ratio */
height: 0;
overflow: hidden;
border-radius: 10px;
margin-top: 20px;
box-shadow: 0 5px 15px rgba(0,0,0,0.2);
}

.youtube-container iframe {
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
border: 0;
}

</style>

</head>

<body>

<div class="container">

<div class="header">

<h1>Your Stress Level Results</h1>

</div>

<div class="row">

```



```

<div class="col-lg-8 offset-lg-2">
<div class="card">
<div class="result-header {{ 'low-stress' if stress_level == 'Low' else 'moderate-
stress' if stress_level == 'Moderate' else 'high-stress' }}">
<div class="result-icon">
<i class="{{ 'fas fa-smile' if stress_level == 'Low' else 'fas fa-meh' if stress le
vel == 'Moderate' else 'fas fa-frown' }}"></i>
</div>
<h2>{{ stress_level }} Stress Level</h2>
<p class="lead">Confidence: {{ confidence }}%</p>
</div>
<div class="card-body">
<h4>Recommendations</h4>
<div class="recommendations">
{% for recommendation in recommendations %}
<div class="recommendation-item">
{{ recommendation }}
</div>
{% endfor %}
</div>
<h4 class="mt-4">Music Therapy</h4>
<p>Listen to this specially selected music to help manage your stress level:</p>
<div class="youtube-container">
<iframe
src="https://www.youtube.com/embed/{{ video_id }}"?autoplay=1"
title="YouTube video player"
allow="accelerometer; autoplay; clipboard-write; encrypted-media;
gyroscope; picture-in-picture"
allowfullscreen>
</iframe>

```

```

</div>
<div class="text-center mt-4">
<a href="/predict" class="btn btn-primary">Make Another Prediction</a>
<a href="/" class="btn btn-outline-secondary ms-2">Back to Home</a>
</div>
</div>
</div>
</div>
</div>
</div>
<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0alpha1/dist/js/bootstrap.bundle.min.js"></script>
<scriptsrc="https://kit.fontawesome.com/a076d05399.js"crossorigin="anonymous"></script>
</body>
</html>

```

Back-End:

```

from flask import Flask, render_template, request, jsonify
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix
import joblib
import os
import io
app = Flask(__name__)
# Global variables to store the model and encoders
model = None
le_gender = LabelEncoder()

```

```

le_stress = LabelEncoder()

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    global model, le_gender, le_stress

    if 'file' not in request.files:
        return jsonify({'error': 'No file part'})

    file = request.files['file']

    if file.filename == "":
        return jsonify({'error': 'No selected file'})

    if file:
        # Read the CSV file
        df = pd.read_csv(file)

        # Drop unnecessary columns
        df = df.drop(['User_ID', 'Bed_Time', 'Wake_Up_Time'], axis=1)

        # Encode categorical variables
        le_gender.fit(df['Gender'])
        df['Gender'] = le_gender.transform(df['Gender'])

        le_stress.fit(df['Stress_Level'])
        df['Stress_Level'] = le_stress.transform(df['Stress_Level'])

        # Split features and target
        X = df.drop('Stress_Level', axis=1)
        y = df['Stress_Level']

        # Split data into train and test sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Train the model
        model = RandomForestClassifier(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)

```

```

# Make predictions on test set
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Save the model
joblib.dump(model, 'model.pkl')

# Get feature importances
feature_importances = model.feature_importances_
feature_names = X.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_
importances})
importance_df = importance_df.sort_values('Importance', ascending=False)

# Convert confusion matrix to list for JSON serialization
cm_list = cm.tolist()

# Get class names
class_names = le_stress.classes_.tolist()

return jsonify({
'success': True,
'accuracy': float(accuracy),
'confusion_matrix': cm_list,
'class_names': class_names,
'feature_importances': importance_df.to_dict(orient='records')
})

@app.route('/predict', methods=['GET', 'POST'])
def predict():
if request.method == 'POST':
if model is None:
return jsonify({'error': 'Please upload and train the model first'})

```

```

# Get form data

try:
age = int(request.form['age'])
gender = request.form['gender']
sleep_duration = float(request.form['sleep_duration'])
sleep_quality = int(request.form['sleep_quality'])
rem_percentage = float(request.form['rem_percentage'])
deep_sleep_percentage = float(request.form['deep_sleep_percentage'])
sleep_interruptions = int(request.form['sleep_interruptions'])
caffeine_intake = int(request.form['caffeine_intake'])
exercise_minutes = int(request.form['exercise_minutes'])
screen_time = int(request.form['screen_time'])
nap_duration = int(request.form['nap_duration'])

# Encode gender
gender_encoded = le_gender.transform([gender])[0]

# Create input array for prediction
input_data = np.array([
age, gender_encoded, sleep_duration, sleep_quality,
rem_percentage, deep_sleep_percentage, sleep_interruptions,
caffeine_intake, exercise_minutes, screen_time, nap_duration
]).reshape(1, -1)

# Make prediction
prediction = model.predict(input_data)[0]

# Get stress level label
stress_level = le_stress.inverse_transform([prediction])[0]

# Get prediction probability
proba = model.predict_proba(input_data)[0]
max_proba = max(proba) * 100

# Get recommendations based on stress level
recommendations = get_recommendations(stress_level)

```

```

# Get YouTube video ID based on stress level
video_id = get_youtube_video(stress_level)
return render_template(
    'result.html',
    stress_level=stress_level,
    confidence=round(max_proba, 2),
    recommendations=recommendations,
    video_id=video_id
)
except Exception as e:
    return jsonify({'error': str(e)})
return render_template('predict.html')
def get_recommendations(stress_level):
    recommendations = {
        'Low': [
            "Your stress levels are low. Keep up the good work!",
            "Continue with your current sleep routine.",
            "Regular exercise and good sleep hygiene are working well for you."
        ],
        'Moderate': [
            "Your stress levels are moderate. Consider some relaxation techniques.",
            "Try meditation or deep breathing exercises before bed.",
            "Reduce screen time before sleep.",
            "Consider limiting caffeine intake, especially in the afternoon."
        ],
        'High': [
            "Your stress levels are high. Please consider consulting a healthcare professional.",
            "Implement a strict sleep schedule.",
            "Practice relaxation techniques like meditation, yoga, or deep breathing.",
            "Reduce caffeine and increase physical activity."
        ]
    }

```

```

"Consider speaking with a therapist or counselor about stress management strategies."
]
}
return recommendations.get(stress_level, [])
def get_youtube_video(stress_level):
    videos = {
        'Low': "jfKfPfyJRdk", # Lofi beats
        'Moderate': "lFcSrYw-ARY", # Relaxing piano music
        'High': "DWcJFNfaw9c" # Deep meditation music
    }
    return videos.get(stress_level, "jfKfPfyJRdk")
if __name__ == '__main__':
    app.run(debug=True)

```