

# **DEVELOPING A FRAMEWORK FOR SLEEPING STAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK ALGORITHM USING EEG DATA**

**A PROJECT REPORT**

*Submitted by*

**AKSHAYA E**

**422519205002**

**KAUSHIKA P**

**422519205018**

**SHOBANA R**

**422519205039**

*in partial fulfilment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**



**UNIVERSITY COLLEGE OF ENGINEERING VILLUPURAM**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY 2023**

# **ANNA UNIVERSITY: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**DEVELOPING A FRAMEWORK FOR SLEEPING STAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK ALGORITHM USING EEG DATA**” is the bonafide work of “**AKSHAYA E(422519205002), KAUSHIKA P(422519205018), SHOBANA R (422519205039)**” who carried out the project work under my supervision in our campus.

**SIGNATURE**

**Dr.P.SEENUVASAN M.E, Ph.D.,**

**HEAD OF THE DEPARTMENT,**

Information Technology,

University College of Engineering Villupuram,

Kakuppam, Villupuram – 605103.

**SIGNATURE**

**Mrs.R.GEETHA M.E.,**

**SUPERVISOR,**

Teaching Fellow,

InformationTechnology,

University College of Engineering Villupuram,

Kakuppam, Villupuram – 605103.

Submitted for IT8811 project viva voice examination held on .....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **ACKNOWLEDGEMENT**

We are grateful to our Dean **Dr.R. Senthil M.E. Ph.D.**, for his constant encouragement to do this project and also during the entire course period.

We much thankful to **Dr.P. SEENUVASAN M.E. Ph.D., Head of the Department**, Department of Information Technology for giving us the opportunity to do project and for providing constant support and academic guidance.

We would like to thank our project coordinator, **Dr. E. Kavitha M.Tech., Ph.D.**, Department of Information Technology for her constant guidance and encouragement in bringing out this project successful.

We express our sincere thanks to our project internal guide **Mrs. R. GEETHA M.E.**, Department of Information Technology for her valuable suggestion and guidance for the development and completion of this project.

Finally, we express our heartfelt thanks to our friends, staff members and many well-wishers for helping in successful completion of our project.

Above all, we thank our parents who are responsible for what we are today.

**AKSHAYA E**

**KAUSHIKA P**

**SHOBANA R**

## **ABSTRACT**

Maintaining proper health and mental stability is critical for overall health and well-being. Despite a good deal of research investment, sleep quality continues to be a crucial public challenge. Nowadays, people of all age groups are affected by improper sleep quality. Poor sleep can lead to a variety of neurological disorders. Sleep disorders are common in all subsets of the population, independently of gender. This public health challenge greatly affects quality of life in terms of both physical and mental health. Insomnia, parasomnias, sleep-related breathing difficulties, hypersomnia, bruxism, narcolepsy, and circadian rhythm disorders are some common examples of sleep-related disorders. Some of these disorders can be treated with proper analysis of early symptoms; in such cases, adequate sleep quality is essential for the patient's recovery. Artificial intelligence has several applications in sleep medicine including sleep and respiratory event scoring in the sleep laboratory, diagnosing and managing sleep disorders, and population health.. Artificial intelligence is a powerful tool in healthcare that may improve patient care, enhance diagnostic abilities, and augment the management of sleep disorders. However, there is a need to regulate and standardize existing machine learning algorithms and deep learning algorithm prior to its inclusion in the sleep clinic. In this project, we can develop the framework for sleeping stage classification for both subjective data and ECG data and classify the data using Convolutional neural network algorithm to analyse the multiple sleeping stages.

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>iii</b>
	<b>LIST OF FIGURES</b>	<b>vii</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>ix</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 DEEP LEARNING	1
	1.2 APPLICATIONS OF DEEP LEARNING	2
	1.2.1 Deep Learning Works	2
	1.2.2 Predictive Analytics	5
	1.3 MACHINE LEARNING	7
	1.3.1 Supervised Learning	8
	1.3.2 Unsupervised Learning	9
	1.4 ARTIFICIAL INTELLIGENCE	9
	1.5 REINFORCEMENT LEARNING	10
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>13</b>
	2.1 INTRODUCTION	13
	2.2 RELATED WORKS	13
<b>3</b>	<b>EXISTING WORK</b>	<b>23</b>
	3.1 EXISTING WORK	23

	3.2 DISADVANTAGES OF EXISTING WORK	23
<b>4</b>	<b>PROPOSED SYSTEM</b>	<b>24</b>
	4.1 PROPOSED WORK	24
	4.2 MODULES	25
	4.2.1 Datasets Acquisition	25
	4.2.2 Preprocessing	26
	4.2.3 Feature Extraction	27
	4.2.4 Classification	28
	4.2.5 Prediction	29
	4.3 ALGORITHM	30
	4.3.1 Convolution Neural Network	30
	4.3.2 Applications	31
	4.3.3 Advantages	31
<b>5</b>	<b>IMPLEMENTATION</b>	<b>32</b>
	5.1 DATASETS	32
	5.2 HARDWARE REQUIREMENTS	34
	5.3 SOFTWARE REQUIREMENTS	34
	5.4 FEATURES OF PYTHON	34
	5.5 TRENDFLOW LIBRARIES IN PYTHON	39

	5.6 USECASE DIAGRAM	42
<b>6</b>	<b>RESULTS AND ANALYSIS</b>	<b>47</b>
<b>7</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>48</b>
	7.1 CONCLUSION	48
	7.2 FUTURE ENHANCEMENT	48
	<b>APPENDIX</b>	<b>64</b>
	<b>SOURCE CODE</b>	<b>64</b>
	<b>REFERENCES</b>	<b>80</b>

## LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	Architecture Diagram	25
4.2	Datasets Acquisition	26
4.3	Preprocessing	27
4.4	Features Extraction	28
4.5	Prediction	29
4.6	Convolutional Neural Network Framework	30
5.1	Different Stages Of Sleep	33
5.2	Activity Diagram	42
5.3	Usecase Diagram	43
5.4	Sequence Diagram	44
5.5	Class Diagram	45
5.6	Collaboration Diagram	46
7.1	Home Page	49
7.2	Counter Plot	51
7.3	Overall Class Data	52
7.4	Stage 1 Not Sleep Data	52
7.5	Stage 2 Not Sleep Data	53
7.6	Stage 3 Not Sleep Data	53
7.7	Stage 4 Not Sleep Data	54
7.8	Stage 1 Sleep Data	54
7.9	Stage 2 Sleep Data	55



7.10	Stage 3 Sleep Data	55
7.11	Stage 4 Sleep Data	56
7.12	Stage 5 Sleep Data	56
7.13	Scatter Matrix For Sleep And Non Sleep	57
7.14	Scatter Matrix For Sleep	57
7.15	Scatter Matrix For Non-Sleep	57
7.16	Prediction Figure	60
7.17	Stage Classification	60
7.18	Precaution Details	60
7.19	Eeg Based Evaluation	61
7.20	Overall Eeg Plot	61
7.21	Annotations Of Multiple Sleeping Stage	62
7.22	Alice And Bob Sleeping Stage	62

## **LIST OF ABBREVIATIONS**

<b>S.NO</b>	<b>ABBREVIATION</b>	<b>EXPLANATION</b>
1	ECU	Electronic Control Unit
2	CNN	Convolutional Neural Network
3	SVM	Support Vector Machine
4	DNA	Deoxyribonucleic Acid
5	AI	Artificial Intelligence
6	API	Application Programming Interface
7	CPU	Central Processing Unit
8	FPGA	Field Programmable Gate Array
9	DDPG	Deep Deterministic Policy Gradient
10	EEG	Electroencephalogram
11	EDF	European Data Format
12	CSV	Comma -Separated Values
13	ECG	Electrocardiogram

# **CHAPTER 1**

## **1. INTRODUCTION**

### **1.1 DEEP LEARNING**

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful. Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

## 1.2 APPLICATIONS OF DEEP LEARNING

Deep learning applications are used in industries from automated driving to medical devices.

**Automated Driving:** Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

**Aerospace and Defense:** Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

**Medical Research:** Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.

**Industrial Automation:** Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

**Electronics:** Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

### 1.2.1 Deep Learning Works

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.

The term deep usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150. Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

One of the most popular types of deep neural networks is known as convolutional neural networks (CNN or ConvNet). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not pretrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

CNNs learn to detect different features of an image using tens or hundreds of hidden layers. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer could learn how to detect edges, and the last learns how to detect more complex shapes specifically catered to the shape of the object we are trying to recognize.

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs end-to-end learning where a network is given raw

data and a task to perform, such as classification, and it learns how to do this automatically.

Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network. A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

The three most common ways people use deep learning to perform object classification are:

- **Training from Scratch**

To train a deep network from scratch, you gather a very large labeled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have a large number of output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train.

- **Transfer Learning**

Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours.

- **Feature Extraction**

A slightly less common, more specialized approach to deep learning is to use the network as a feature extractor. Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during the training process. These features can then be used as input to a machine learning model such as support vector machines (SVM).

### **1.2.2 Predictive Analysis**

Predictive analytics uses historical data to predict future events. Typically, historical data is used to build a mathematical model that captures important trends. That predictive model is then used on current data to predict what will happen next, or to suggest actions to take for optimal outcomes. Predictive analytics has received a lot of attention in recent years due to advances in supporting technology, particularly in the areas of big data and machine learning.

Predictive analytics is often discussed in the context of big data, Engineering data, for example, comes from sensors, instruments, and connected systems out in the world. Business system data at a company might include transaction data, sales results, customer complaints, and marketing information. Increasingly, businesses make data-driven decisions based on this valuable trove of information. To extract value from big data, businesses apply algorithms to large data sets using tools such as Hadoop and Spark. The data sources might consist of transactional databases, equipment log files, images, video, audio, sensor, or other types of data. Innovation often comes from combining data from several sources.

With all this data, tools are necessary to extract insights and trends. Machine learning techniques are used to find patterns in data and to build models that predict future outcomes. A variety of machine learning algorithms are available, including linear and nonlinear regression, neural networks, support vector machines, decision trees, and other algorithms.

Predictive analytics helps teams in industries as diverse as finance, healthcare, pharmaceuticals, automotive, aerospace, and manufacturing.

- **Automotive** – Breaking new ground with autonomous vehicle
  - Companies developing driver assistance technology and new autonomous vehicles use predictive analytics to analyze sensor data from connected vehicles and to build driver assistance algorithms.
- **Aerospace** – Monitoring aircraft engine health
  - To improve aircraft up-time and reduce maintenance costs, an engine manufacturer created a real-time analytics application to predict subsystem performance for oil, fuel, liftoff, mechanical health, and controls.
- **Energy Production** – Forecasting electricity price and demand
  - Sophisticated forecasting apps use models that monitor plant availability, historical trends, seasonality, and weather.
- **Financial Services** – Developing credit risk models
  - Financial institutions use machine learning techniques and quantitative tools to predict credit risk.
- **Industrial Automation and Machinery** – Predicting machine failures
  - A plastic and thin film producer saves 50,000 Euros monthly using a health monitoring and predictive maintenance application that reduces downtime and minimizes waste.
- **Medical Devices** – Using pattern-detection algorithms to spot asthma and copd
  - An asthma management device records and analyzes patients' breathing sounds and provides instant feedback via a smart phone app to help patients manage asthma and copd.

Predictive analytics is the process of using data analytics to make predictions based on data. This process uses data along with analysis, statistics, and machine learning techniques to create a predictive model for forecasting future events.



The term predictive analytics describes the application of a statistical or machine learning technique to create a quantitative prediction about the future. Frequently, supervised machine learning techniques are used to predict a future value. Predictive analytics starts with a business goal to use data to reduce waste, save time, or cut costs. The process harnesses heterogeneous, often massive, data sets into models that can generate clear, actionable outcomes to support achieving that goal, such as less material waste, less stocked inventory, and manufactured product that meets specifications.

### 1.3 MACHINE LEARNING

Machine learning is a data analytics technique that teaches computers to do what comes naturally to humans and animals learn from experience. Machine learning algorithms use computational methods to learn information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases. Deep learning is a specialized form of machine learning. With the rise in big data, machine learning has become a key technique for solving problems in areas, such as

- **Computational finance** for credit scoring and algorithmic trading
- **Image processing and computer vision** for face recognition, motion detection, and object detection
- **Computational biology** for tumour detection, drug discovery, and DNA sequencing
- **Energy production** for price and load forecasting
- **Automotive, aerospace, and manufacturing** for predictive maintenance
- **Natural language processing** for voice recognition applications

Machine learning uses two types of techniques: Supervised learning, which trains a model on known input and output data so that it can predict future outputs,

and unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

### **1.3.1 Supervised Learning**

Supervised machine learning builds a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data. Use supervised learning if you have known data for the output you are trying to predict. Supervised learning uses classification and regression techniques to develop predictive models.

- **Classification techniques**

Predict discrete responses for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring. Use classification if your data can be tagged, categorized, or separated into specific groups or classes. For example, applications for hand-writing recognition use classification to recognize letters and numbers. In image processing and computer vision, unsupervised pattern recognition techniques are used for object detection and image segmentation. Common algorithms for performing classification include support vector machine(SVM), boosted and bagged decision trees, k-nearest neighbour, Naive Bayes, discriminant analysis, logistic regression, and neural networks.

- **Regression techniques**

Predict continuous responses for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading. Use regression techniques if you are

working with a data range or if the nature of your response is a real number, such as temperature or the time until failure for a piece of equipment.

common regression algorithms include linear model, nonlinear model, regularization, stepwiseregression, boosted and bagged decision trees, neural networks, and adaptive neuro-fuzzy learning.

### 1.3.2 Unsupervised Learning

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labeled responses.

- **Clustering** is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Applications for cluster analysis include gene sequence analysis, market research, and object recognition.

## 1.4 ARTIFICIAL INTELLIGENCE

Artificial intelligence or AI, is a simulation of intelligent human behavior. It's a computer or system designed to perceive its environment, understand its behaviors, and take action. Consider self-driving cars AI-driven systems like these integrate AI algorithms, such as machine learning and deep learning, into complex environments that enable automation. Taking raw data and making it useful for an accurate, efficient, and meaningful model is a critical step. In fact, it represents most of your AI effort.

Data preparation requires domain expertise, such as experience in speech and audio signals, navigation and sensor fusion, image and video processing, and radar and lidar. Engineers in these fields are best suited to determine what the critical features of the data are, which are unimportant, and what rare events to consider.

AI also involves prodigious amounts of data. Yet labelling data and images is tedious and time-consuming. Sometimes, you don't have enough data, especially for

safety-critical systems. Generating accurate synthetic data can improve your data sets. In both cases, automation is critical to meeting deadlines.

**Deployment:** AI models need to be deployed to CPUs, GPUs, and/or FPGAs in your final product, whether part of an embedded or edge device, enterprise system, or cloud. AI models running on the embedded or edge device provide the quick results needed in the field, while AI models running in enterprise systems and the cloud provide results from data collected across many devices. Frequently, AI models are deployed to a combination of these systems. The deployment process is accelerated when you generate code from your models and target your devices. Using code generation optimization techniques and hardware-optimized libraries, you can tune the code to fit the low power profile required by embedded and edge devices or the high-performance needs of enterprise systems and the cloud.

## 1.5 REINFORCEMENT LEARNING

In control systems that benefit from learning based on cumulative reward, reinforcement learning is an ideal technique. Let's you train policies using DQN, A2C, DDPG, and other reinforcement learning algorithms. You can use these policies to implement controllers and decision-making algorithms for complex systems such as robots and autonomous systems. You can implement the policies using deep neural networks, polynomials, or lookup tables. Reinforcement learning is a type of machine learning technique where a computer agent learns to perform a task through repeated trial and error interactions with a dynamic environment. This learning approach enables the agent to make a series of decisions that maximize a reward metric for the task without human intervention and without being explicitly programmed to achieve the task. AI programs trained with reinforcement learning beat human players in board games like Go and chess, as well as video games. While reinforcement learning is by no means a new concept, recent progress in deep learning and computing power made it possible to achieve some remarkable results in the area of artificial intelligence.

Reinforcement learning is a branch of machine learning (Figure1). Unlike unsupervised and supervised machine learning, reinforcement learning does not rely on a static dataset, but operates in a dynamic environment and learns from collected experiences. Data points, or experiences, are collected during training through trial-and-error interactions between the environment and a software agent. This aspect of reinforcement learning is important, because it alleviates the need for data collection, preprocessing, and labeling before training, otherwise necessary in supervised and unsupervised learning. Practically, this means that, given the right incentive, a reinforcement learning model can start learning a behavior on its own, without (human) supervision.

Deep learning spans all three types of machine learning; reinforcement learning and deep learning are not mutually exclusive. Complex reinforcement learning problems often rely on deep neural networks, a field known as deep reinforcement learning

Deep neural networks trained with reinforcement learning can encode complex behaviours. This allows an alternative approach to applications that are otherwise intractable or more challenging to tackle with more traditional methods. For example, in autonomous driving, a neural network can replace the driver and decide how to turn the steering wheel by simultaneously looking at multiple sensors such as camera frames and lidar measurements. Without neural networks, the problem would normally be broken down in smaller pieces like extracting features from camera frames, filtering the lidar measurements, fusing the sensor outputs, and making driving decisions based on sensor inputs. While reinforcement learning as an approach is still under evaluation for production systems, some industrial applications are good candidates for this technology.

**Advanced controls:** Controlling nonlinear systems is a challenging problem that is often addressed by linearizing the system at different operating points. Reinforcement learning can be applied directly to the nonlinear system.

**Automated driving:** Making driving decisions based on camera input is an area where reinforcement learning is suitable considering the success of deep neural networks in image applications.

**Robotics:** Reinforcement learning can help with applications like robotic grasping, such as teaching a robotic arm how to manipulate a variety of objects for pick-and-place applications. Other robotics applications include human-robot and robot-robot collaboration.

**Scheduling:** Scheduling problems appear in many scenarios including traffic light control and coordinating resources on the factory floor towards some objective. Reinforcement learning is a good alternative to evolutionary methods to solve these combinatorial optimization problems.

**Calibration:** Applications that involve manual calibration of parameters, such as electronic control unit (ECU) calibration, may be good candidates for reinforcement learning.

## **CHAPTER 2**

### **2. LITERATURE SURVEY**

#### **2.1 TITLE: DETECTION OF INSOMNIA USING ELECTRO-CARDIOGRAPHY AND ELECTROMYOGRAPHY**

**AUTHOR: ASRA MALIK**

**YEAR:2021**

**DESCRIPTION:** Insomnia is a disorder of sleep in which a person constantly complains about insufficient sleep despite having an adequate amount of time to sleep. It is considered as one of the most common psychological illnesses with an approximate severity of about 10%. Frustration is the key feature of insomnia. The quality and quantity of sleep remain in trouble while falling asleep, sustaining sleep, or rising in the early morning. Insomnia is also associated with several other sleep-wake, emotional, or medical conditions which rises attention for its separate treatment. Increased physical, psychological, and physiological stimulation is considered to be a key underlying in most reports of chronic insomnia along with perpetuating behavioural issues like prolonged time in bed. It is also reported in a research work that insomnia can also combine with cardiovascular diseases. It has a very adverse effect on the job, mental, social, and overall quality of life of an individual. Insomnia is one of the serious sleep disorders. In this disease, a patient feels trouble falling asleep and stay awake for the whole night. Insomnia also co-morbid with other diseases like cancer, cardiovascular disease, asthma, Alzheimer's, Parkinson's disease. This research aims at a method for identifying insomnia using both electrocardiogram (ECG) and electromyogram (EMG) signals. Features with high discriminative ability are extracted from signals to classify via logistic regression (LR), support vector machines (SVM), K-nearest neighbour (KNN), decision tree (DT), ensemble classifier (EC), and Naive Bayes (NB) classification methods. we have achieved the highest accuracy of 100% on both ECG and EMG signals from our proposed methodology

## **2.2 TITLE: AUTOMATIC IDENTIFICATION OF INSOMNIA BASED ON SINGLE-CHANNEL EEG LABELLED WITH SLEEP STAGE ANNOTATIONS**

**AUTHOR: BUFANG YANG**

**YEAR: 2020**

**DESCRIPTION:** In this paper, we proposed a 1D-CNN model for automatic insomnia identification based on single-channel EEG labelled with sleep stage annotations, and further investigated the identification performance based on different sleep stages. Our experiments demonstrated that our 1D-CNN leveraging the 3 sub datasets composed of REM, LSS and SWS epochs, respectively, achieved higher average accuracy in comparison with baseline methods under both intra-patient and inter-patient paradigms. The experimental results also indicated that amongst all the sleep stages, 1D-CNN leveraging REM and SWS epochs exhibited the best insomnia identification performance in intra-patient paradigm, whereas no statistically significant difference was found in inter-patient paradigm. Overall, for automatic insomnia identification based on single-channel EEG labelled with sleep stages, 1D-CNN model introduced in this paper could achieve superior performance than traditional methods. Further experiment based on larger sleep databases under inter-patient paradigm is still required in future work. Moreover, the comparison of the two experiments demonstrated that the average identification accuracy of our 1D-CNN could achieve 99.16% in intra-patient experiment, whereas it could only reach 87.49% in inter-patient experiment with larger standard deviation. We consider the high accuracy in intra-patient experiment is caused by the similarity of epochs, i.e., epochs from the same patient are utilized both for training and testing. However, inter-patient experiment is more realistic evaluation paradigm which could guarantee the generalizability of the method. Therefore, we suggest future research on automatic insomnia identification based on deep learning should focus on the inter-patient experiment performance.



## **2.3 TITLE: A SHORT-TIME INSOMNIA DETECTION SYSTEM BASED ON SLEEP EOG WITH RCMSE ANALYSIS**

**AUTHOR: CHIH-EN KUO**

**YEAR: 2020**

**DESCRIPTION:** Sleep takes approximately one-third of human live. A good sleep can help us getting the body to work right again, improved learning ability, physical development, emotional regulation, and good quality of life in human physiology. However, the prevalence of insomnia symptoms without restrictive criteria is approximately 33% in the general population. In the United States of America, 50-70 million people suffer from sleep disorders: among them, 30% of patients suffer from insomnia and 10% from chronic insomnia. Insomnia is defined as chronic when it has persisted for at least three months at a frequency of at least three times per week. When the disorder meets the symptom criteria but has persisted for less than three months, it is considered short-term insomnia. To diagnose insomnia, the physician may first execute a physical exam to look for signs of medical problems that may be related to insomnia and ask some sleep-related questions, such as sleep-wake pattern and daytime sleepiness. In addition, the physician may ask a subject to keep a sleep diary with the actigraphy for a couple of weeks. If the cause of insomnia is not clear, the subject should spend one or two nights at a sleep centre diagnosing another sleep disorder using polysomnography (PSG), such as sleep apnoea. PSG recordings, which including electroencephalogram (EEG), electrooculogram (EOG), electromyogram (EMG), and other physiological signals are usually obtained from patients and scored by a well-trained clinical staff. Moreover, manual sleep scoring and diagnosis is a time consuming and subjective process, and the conventional polysomnography which uses many wires to connect instrument to patient is often a problem that leads to sleep disturbance. In general, people will experience the first night effect, which often interferes with sleep quality, if they sleep in hospitals or sleep centres.

## **2.4 TITLE: INSOMNIA: TOWARDS CONCEPT-DRIFT ROBUSTNESS IN NETWORK INTRUSION DETECTION**

**AUTHOR: GIUSEPPINA ANDRESINI**

**YEAR: 2021**

**DESCRIPTION:** This paper outlines a set of open challenges facing modern ML-based intrusion detectors relating to a lack of uniformity in the distribution of traffic data over time. To tackle them, we propose INSOMNIA, a semi-supervised approach that uses active learning to reduce latency in the model updates, label estimation to reduce labelling overhead, and applies explainable AI to describe how the model evolves to fit the shifting distribution. We extend the TESSERACT framework to perform a time-aware evaluation of INSOMNIA on a recently published, revised version of CICIDS2017 and demonstrate that modern intrusion detection systems must address concept drift in order to be effective. They rely on a subset of ground truth labels to retrain at each time step. In contrast to these previously mentioned approaches, INSOMNIA only requires labels at the initial training time, and then sustains itself without requiring manual labelling; instead, it generates pseudo labels based on the nearest centroid neighbour. Updating with the model's predicted labels has been proposed as a solution in the malware domain, but has been shown to lead to self-poisoning; this risk is mitigated in INSOMNIA's use of curtaining by using two distinct algorithms for label estimation and prediction. The results of our evaluation highlight yet another important open problem for NIDS detection of stealthy, low-prevalence attacks. INSOMNIA struggles to detect the few instances of the Infiltration attack at windows and we observe that maintaining sensitivity to attacks with a very low base rate in the presence of high-volume attacks such as Dos is very challenging as is generalizing to attack categories of greatly different character. While generalizing across attack types remains a holy grail, future work may consider ensembles that tackle different attack types with separate models.

## **2.5 TITLE: AUTOMATIC IDENTIFICATION OF INSOMNIA USING OPTIMAL ANTISYMMETRIC BIORTHOGONAL WAVELET FILTER BANK WITH ECG SIGNALS**

**AUTHOR: MANISH SHARMA**

**YEAR: 2021**

**DESCRIPTION:** Sleep is a fundamental human physiological activity required for adequate working of the human body. Sleep disorders such as sleep movement disorders, nocturnal front lobe epilepsy, insomnia, and narcolepsy are caused due to low sleep quality. Insomnia is one such sleep disorder where a person has difficulty in getting quality sleep. There is no definitive test to identify insomnia; hence it is essential to develop an automated system to identify it accurately. A few automated methods have been proposed to identify insomnia using either polysomnogram (PSG) or electroencephalogram (EEG) signals. To the best of our knowledge, we are the first to automatically detect insomnia using only electrocardiogram (ECG) signals without combining them with any other physiological signals. In the proposed study, an optimal antisymmetric biorthogonal wavelet filter bank (ABWFB) has been used, which is designed to minimize the joint duration-bandwidth localization (JDBL) of the underlying filters. We created ten different subsets of ECG signals based on annotations of sleep stages, namely wake (W), S1, S2, S3, S4, rapid eye movement (REM), light sleep stage (LSS), slow-wave sleep (SWS), non-rapid eye movement (NREM) and W+S1+S2+S3+S4+REM for the automated identification of insomnia. Our proposed ECG-based system obtained the highest classification accuracy of 97.87%, F1-score of 97.39%, and Cohen's kappa value of 0.9559 for K-nearest neighbour (KNN) with the ten-fold cross-validation strategy using ECG signals corresponding to the REM sleep stage. The support vector machine (SVM) yielded the highest value of 0.99 for area under the curve with the tenfold cross-validation corresponding to REM sleep stage

## **2.6 TITLE: MULTITASK FMRI AND MACHINE LEARNING APPROACH IMPROVE PREDICTION OF DIFFERENTIAL BRAIN ACTIVITY PATTERN IN PATIENTS WITH INSOMNIA DISORDER**

**AUTHOR: MI HYUN LEE**

**YEAR: 2021**

**DESCRIPTION:** Insomnia is a common, distressing, and clinically important symptom, which causes difficulty initiating sleep; frequent awakening; or early-morning awakening with daytime dysfunction. Approximately one-third of the general population experiences insomnia symptoms throughout their lifetime; 1-year follow-up of patients with insomnia revealed that 69% exhibited chronic symptoms. Because of its high prevalence and frequent associations with medical and psychiatric disorders, there is a need to study the neurobiological mechanisms of insomnia to improve quality of life and reduce socioeconomic burden. According to the International Classification of Sleep Disorders Second Edition (ICSD-2), psychophysiological insomnia (PI) is a form of insomnia that comprises heightened arousal and learned sleep-preventing associations, which cause insomnia and poor functioning during wakefulness. Patients with PI frequently complain of sleep-related worries and ruminations that may reduce cognitive performance. Hyperarousal is an important model to understand the pathophysiology of insomnia. While psychophysiological experiments related to hyperarousal insomnia have been actively investigated, there have been few task-related neuroimaging studies involving the provision of sleep-related stimuli for evaluation of hyperarousal reactions in patients with PI. Baglioni et al. reported that, compared with good sleepers, patients with insomnia disorder presented heightened activity in the amygdala in response to insomnia-related pictures. In a previous study of 14 patients with PI, the regional brain activity in response to sleep-related sound was reportedly reduced in the left middle temporal and left middle occipital gyri after cognitive behavioural therapy for insomnia.

**2.7 TITLE: DISCRIMINATING PARADOXICAL AND PSYCHOPHYSIOLOGICAL INSOMNIA BASED ON STRUCTURAL AND FUNCTIONAL BRAIN IMAGES: A PRELIMINARY MACHINE LEARNING STUDY**

**AUTHOR: MORTAZA AFSHANI**

**YEAR: 2022**

**DESCRIPTION:** Recent studies have suggested that there are various ID subtypes with different ethology and distinct pathophysiology. Subtyping ID is an ongoing debate in sleep medicine and has changed over different versions of the International Classification of Sleep Disorders (ICSD). In ICSD-2, several ID subtypes were introduced including paradoxical insomnia (PDI) and psychophysiological insomnia (PPI). ICSD-2 defines PDI as subjective complaints of insomnia, while polysomnography (PSG) shows near-normal sleep patterns, causing a discrepancy between subjective and objective sleep measures. PPI, instead, is characterized by increased arousal before or during sleep and learned-sleep preventing associations in a routine bedroom setting. Put differently, the more a patient tries to sleep, the more he/she gets anxious and more difficult to initiate sleep, which causes low functional performance during wakefulness. Thereby, PPI is described by fear of sleep and the bedroom environment. However, although ICSD-3 has emphasized the existence of the ID subtypes, it proposed to consider it as a single category, entitled chronic ID, as there are no customized therapeutic approaches for each subtype. This preliminary study provides evidence that structural and functional brain measures can help to distinguish two common subtypes of ID from each other and from healthy subjects. Moreover, we observed that the multimodal brain measure is a bit better than the unimodal brain measure to separate ID subtypes. Future studies with larger sample sizes should further investigate neurobiological mechanisms underlining ID subtypes to optimize and develop new personalized therapeutic approaches in the future.

## **2.8 TITLE: PREDICTION OF CHRONIC INSOMNIA USING MACHINE LEARNING TECHNIQUES**

**AUTHOR: MD. MUHAJMINUL ISLAM**

**YEAR: 2020**

**DESCRIPTION:** The analysis shows that our Logistic regression model performed best compared to other models. It will be very much handy in real-life prediction for its outstanding cross-validation score. But still, there have been some limitations in our study. For the lack of time, we were unable to gather massive data. Though the accuracy of our model is great but if we could gather more data, it could be greater. Again, this study is done basically done for predicting chronic insomnia in humans. So, feature was selected according to external symptoms only. If we had collected data on regular habits or lifestyle (smoking, drinking, level of using radio-wave devices) then it could be determined also that which factors cause insomnia and the work would be much beneficial. The thoughts upon this work are not limited just in here. If we have enough scope in the future, we could complete this works also. We would like to deploy the model and want to build a web-based automated system where patients could check whether their sleep status conflicts with insomnia or not. We would also like to add more data in our current dataset. There is a possibility that this model will perform much better on a large dataset. This is a resampling procedure that is used to evaluate the models on a limited dataset. In simple, when the whole dataset is divided into training and testing set there can be such a situation that the testing set contains random data that the training doesn't talk about at all. In that case, the accuracy of the testing data will be less than what it should have. Cross-validation solves this issue. It has a parameter  $k$  which indicates the number of groups that a dataset will be divide into. This method takes near every random data in the training set and carries on.

## **2.9 TITLE: FUNCTIONAL CONNECTOME FINGERPRINT OF SLEEP QUALITY IN INSOMNIA PATIENTS: INDIVIDUALIZED OUT-OF-SAMPLE PREDICTION USING MACHINE LEARNING**

**AUTHOR: XIAOFEN MA**

**YEAR: 2020**

**DESCRIPTION:** In conclusion, the present study demonstrated the nodal functional connectivity strength predicted unseen individuals' sleep quality in both short-term/acute and chronic insomnia. We further revealed changes in the functional connectivity pattern during the transition from the short-term/acute insomnia to chronic insomnia. The study may have clinical value by informing the diagnosis of sleep quality of insomniac patients, and may provide novel insights into the neural basis underlying the heterogeneity of insomnia. Finally, the present work showed that it is important to differentiate the stages of sleep quality in future studies. Insomnia disorder has been reclassified into short-term/acute and chronic subtypes based on recent etiological advances. However, understanding the similarities and differences in the neural mechanisms underlying the two subtypes and accurately predicting the sleep quality remain challenging. Using resting state functional magnetic resonance imaging (rsfMRI), prior studies have consistently demonstrated abnormal spontaneous regional brain activity in patients with insomnia disorder. For example, these patients showed lower spontaneous activity in regions of higher-order cognitive networks. Typically, a specific behaviour score is initially estimated in a connectivity-based predictive model using training samples; it is then validated using independent testing samples. This approach generally employs machine learning, which is a multivariate pattern analysis approach that can capture the relationship between the complex pattern of whole-brain features and behaviours, and can therefore provide more information beyond the traditional mass-univariate analysis. Once the model can generalize well within the testing samples, it captures the brain representation of the behaviour.

## **2.10 TITLE: IDENTIFICATION OF DISCRIMINATIVE NEUROIMAGING MARKERS FOR PATIENTS ON HAEMODIALYSIS WITH INSOMNIA: A FRACTIONAL AMPLITUDE OF LOW FREQUENCY FLUCTUATION-BASED MACHINE LEARNING ANALYSIS**

**AUTHOR: ZE-YING WEN**

**YEAR: 2022**

**DESCRIPTION:** Patients receiving maintenance haemodialysis (HD) frequently report insomnia complaints, with a high prevalence ranging from 40 to 85% worldwide. It has been shown that HD patients with insomnia (HDWI) present a variety of comorbidities such as irritability, immune suppression, anxiety, depression, cognitive impairment, etc., which may have a potentially great impact on their quality of life and even survival. Despite the fact that the current management including medication, behavioural cognitive therapy (CBT-i) and acupuncture has developed for HDWI, it is far from satisfactory and standardized clinical procedures regardless of individual differences may increase the subjects' risk factors amongst HD patients, highlighting the urgent need to fully understand the pathophysiology of the disorder and help achieve advances in the prevention and treatment of the condition. This is the first we constructed a SVM classifier to find a promising model for HDWI classification. Although previous studies have offered insights into brain functional and structural abnormalities of neurological diseases including primary insomnia using traditional group-level fMRI analysis, they could not translate into diagnostic or predictive neural markers for such neurological diseases, especially in HD cohort. The study just felled the gap in such cohort by constructing a highly discriminative SVM classifier, which was efficient to provide preliminary support to develop the individualized therapeutic aid for HDWI. In fact, the ability to advise clinicians and patients accurately regarding the chances of proper therapy is of great importance, particularly as improper therapy is an occupation of medical resource waste and may has some side reactions.



## **CHAPTER 3**

### **3. EXISTING SYSTEM**

#### **3.1 EXISTING WORK**

Insomnia is a common sleep disorder that can lead to difficulty falling asleep, staying asleep, or waking up too early. There are various existing systems for insomnia sleep stage prediction that can provide valuable information about sleep patterns. Polysomnography (PSG) is considered the gold standard for sleep stage classification and involves monitoring various physiological signals to identify different stages of sleep. There are also various wearable devices available in the market that claim to track sleep stages, and machine learning models can be used to predict sleep stages based on various physiological signals. Support Vector Machine (SVM) is another machine learning algorithm that can be used for sleep stage classification. SVM is a supervised learning algorithm that can classify data into different categories by finding the best separating hyperplane. The extracted features are then used to train the SVM, which can then classify new epochs of PSG recordings into different sleep stages. Some common features used for sleep stage classification using SVMs include spectral features, such as power in different frequency bands, and statistical features, such as the mean and variance of the EEG signal. SVMs can also be combined with other machine learning algorithms, such as k-nearest neighbours (k-NN) or decision trees, to improve classification performance.

#### **3.2 DISADVANTAGES**

- Computational Requirements: Deep learning algorithms require significant computational resources and time to train, which can be a challenge for healthcare systems with limited resources.
- Time complexity is high
- Does not support large datasets
- Accuracy is less

## **CHAPTER 4**

### **4. PROPOSED SYSTEM**

#### **4.1 PROPOSED WORK**

Sleep is the brain's primary function and plays a fundamental role in individual performance, learning ability, and physical movement. One of the essential physiological processes of humans is sleep vital for physical and cognitive well-being and resurgence. Sleep is a reversible state in which the eyes are closed, and several nerve centres are disabled. Sleep creates partial or unique or full anaesthesia for the individual, in which case the brain becomes a less complicated network. The conventional solution is to detect the artefacts and denoise the signal by removing corresponding epochs from the sleep signal. However, this way, the EEG signal will be manipulated and may lose important information. One of the motivations of this thesis is to develop and improve noise cancelation method that does not manipulate the signal and protect its originality. The proposed system for sleep stage classification using CNNs from an EDF and CSV file involves several steps. The first step is data pre-processing, where the raw data is filtered, resampled, and features are extracted. This step helps to remove noise and ensure that the data is consistent across all recordings. The next step is data splitting, where the data is split into training, validation, and testing sets. The training set is used to train the CNN model, the validation set is used to tune the hyperparameters and prevent overfitting, and the testing set is used to evaluate the performance of the model. Once the data is pre-processed and split, the next step is to design the CNN architecture. The architecture should be designed to learn relevant features from the pre-processed data and classify the sleep stages accurately. A common approach is to use 1D or 2D CNNs with multiple convolutional layers and pooling layers. The output of the final layer is fed to a fully connected layer and softmax activation to predict the sleep stage. And also predict the sleeping disorder with new datasets, then provide the precaution details.

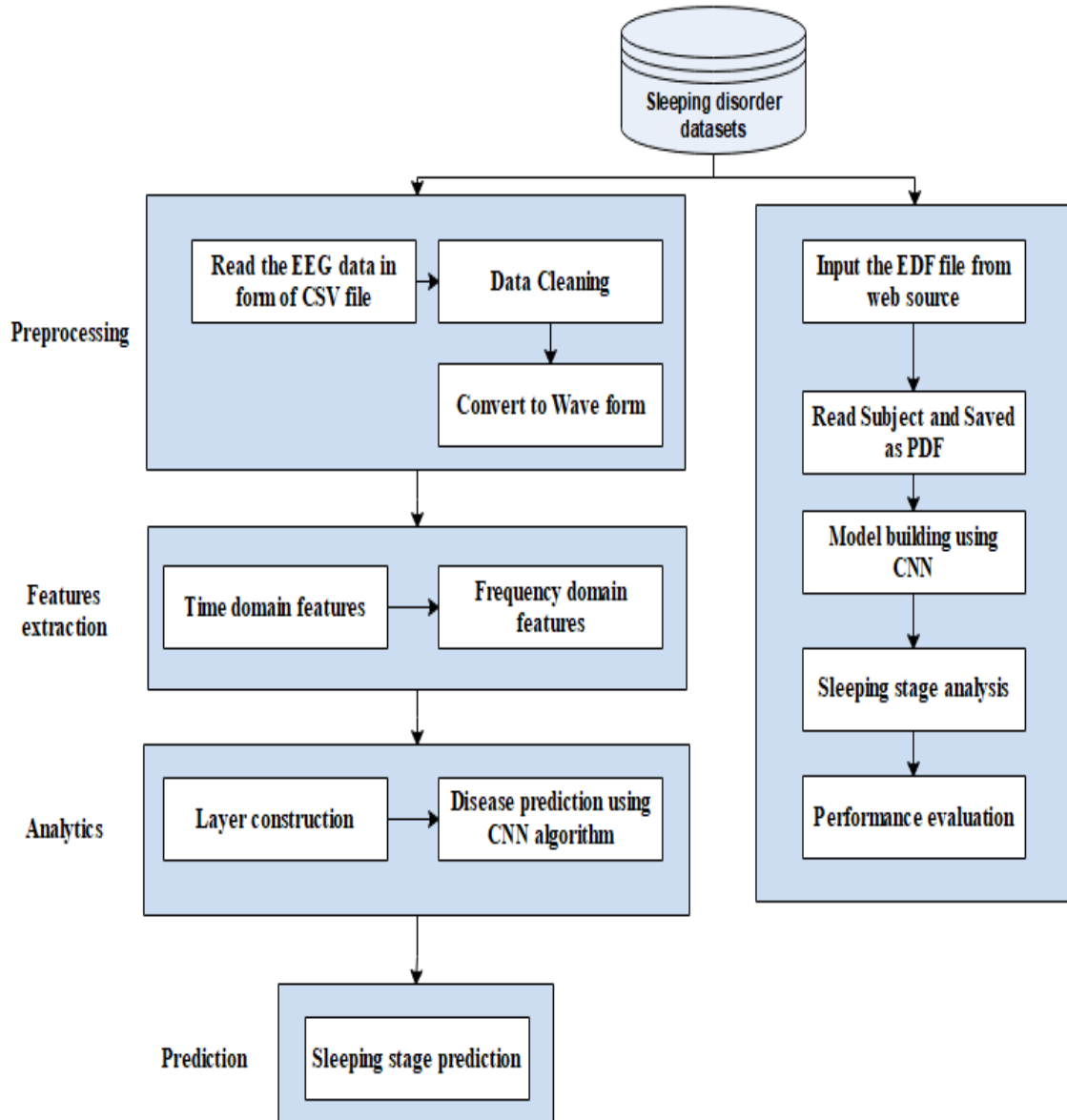


Figure 4.1 Architecture Diagram

## 4.2 MODULES

### 4.2.1 Datasets Acquisition

In this module we can input the CSV file about EEG data and also EDF file. The Physio net dataset is a dataset of EEG recordings for multiple persons Physio Net

is a public repository of biomedical signals and time series data, which can be used for research and development in the field of sleep stage prediction. It contains large datasets of sleep and physiological data, including EEG, ECG, and respiration signals, which can be used to train and evaluate deep learning algorithms for sleep stage prediction. In this module, we can input the CSV file and EDF file. CSV file contains the frequency values. EDF (European Data Format) is a common file format for storing physiological signals, including those recorded during sleep. EDF files typically contain several signals, including EEG, EOG, EMG, and ECG. These signals are recorded using electrodes placed on the scalp, face, and body, and provide information about the electrical activity of the brain and muscles during sleep. To use an EDF file for sleeping stage prediction, the first step is to load the file into a software program or Python library that can read EDF files. Examples of such libraries include MNE-Python and pyEDFlib. Once the EDF file is loaded, the next step is to extract features from the signals that are relevant to sleep stage prediction. Common features include the power spectrum, the amplitude of specific frequency bands, and the correlation between different signals.

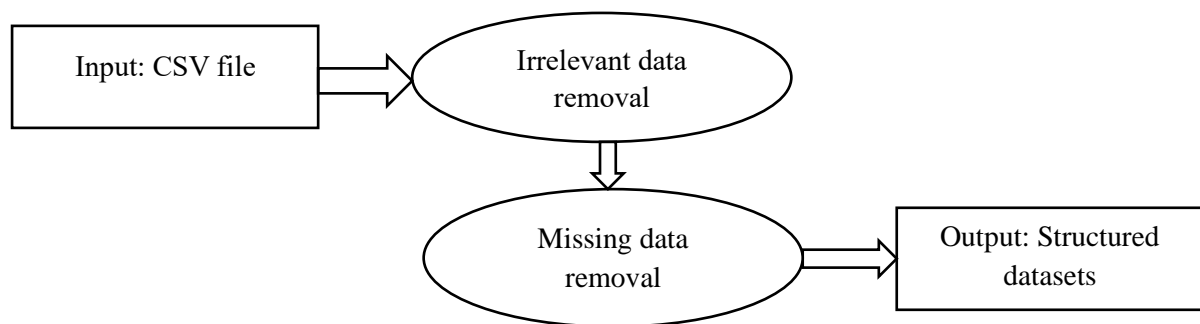


Figure 4.2 Datasets Acquisition

### 4.2.2 Preprocessing

Preprocessing is an important step in the analysis of EEG data, as it helps to remove artifacts and improve the quality of the data. In the pre-processing stage, continuous EEG recordings are firstly segmented without overlapping by a sliding time

window. In this module, perform preprocessing steps to eliminate the irrelevant and missing datasets from uploaded CSV file. The first step is to load the EDF file into memory using a library such as MNE-Python or pyEDFlib. The CSV file containing the sleep stage annotations can also be loaded into a Pandas DataFrame. Sleep signals are often contaminated by noise and artifacts. Signal filtering can be used to remove these unwanted components and improve the quality of the signal. Common filtering techniques include bandpass filtering, notch filtering, and high-pass filtering.

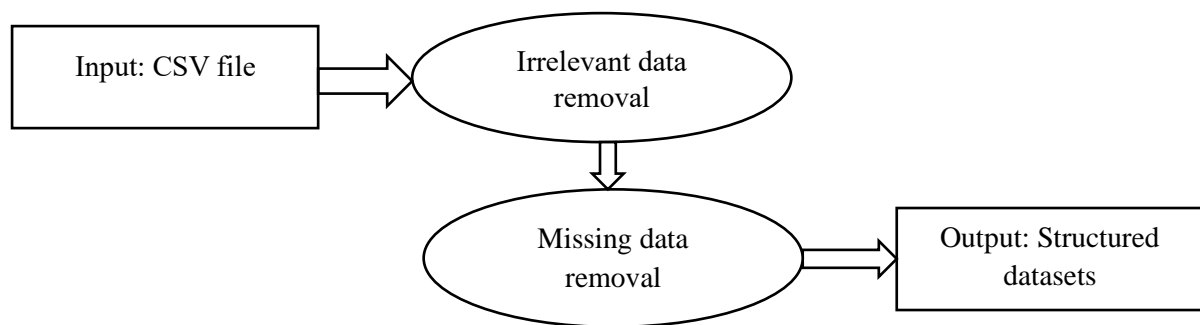


Figure 4.3 Preprocessing

### 4.2.3 Features Extraction

Feature extraction is the process of transforming raw data into a set of features that can be used to represent the data in a more meaningful and useful way. Relevant features need to be extracted from the signal to represent the underlying physiological processes. Common features for sleep stage classification include power spectral density, amplitude of specific frequency bands, and correlation between different signals. In the context of a Convolutional Neural Network (CNN) for EEG data, feature extraction involves transforming the raw EEG signals into a set of features that can be used as input to the CNN. In this module, we can calculate the mean and standard deviation for uploaded CSV file. Normalizing the data is important to ensure that all features have the same scale and range. This helps to improve the performance of the machine learning algorithm. In CSV file, train the Y-values with stage and X-train values contain the frequency values. Based on features model file can be created using CNN algorithm for future verification.

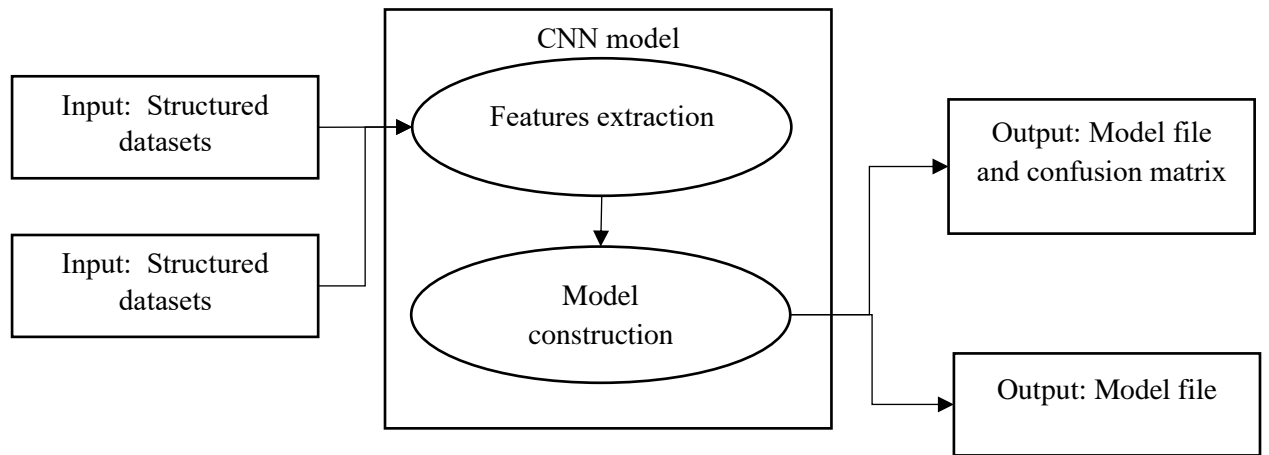


Figure 4.4 Features Extraction

#### 4.2.4 Classification

The features are divided into training, validation, and test sets, and are organized into a format that can be used as input to the CNN. A CNN is designed and configured, including the choice of architecture, number of layers, and activation functions. The CNN is trained on the training set, using a loss function and an optimization algorithm. The model is validated on the validation set to prevent overfitting and monitor the performance of the model. The trained CNN is evaluated on the test set to assess its performance in classifying the sleep stages. A CSV (Comma-Separated Values) file can be used for sleep stage classification in EEG datasets by providing annotations for each epoch of data. An epoch is a fixed-length segment of EEG data, typically 30 seconds in length. The annotations in the CSV file indicate the sleep stage that each epoch belongs to, as determined by a sleep expert. To classify EEG datasets using a CSV file, the first step is to load the EEG data and the corresponding CSV file into memory. The EEG data can be loaded using a library such as MNE-Python or pyEDFlib, while the CSV file can be loaded into a Pandas Data Frame. Once the data is loaded, the next step is to preprocess the EEG data to extract features that are relevant for sleep stage classification. This can involve techniques such as filtering the data, resampling the data, and extracting features such as power spectral density, amplitude of specific frequency bands, and correlation between different EEG channels. A common

approach for EEG classification is to use a convolutional neural network (CNN). The CNN is trained on the labeled epochs to learn the relationship between the EEG features and the sleep stages. The performance of the CNN is evaluated using metrics such as accuracy, precision, recall, and F1 score.

#### 4.2.5 Prediction

Finally, the trained CNN can be used to classify the sleep stages in new EEG datasets. The EEG data is divided into epochs, and the CNN predicts the sleep stage for each epoch. This information can be used to analyze the sleep patterns of individuals and can provide valuable insights for clinical and research purposes. In this module, input the EEG readings in terms of numeric format. And match the reading with CNN model file. Then classify the stages of sleeping disorders and provide the precautions to users

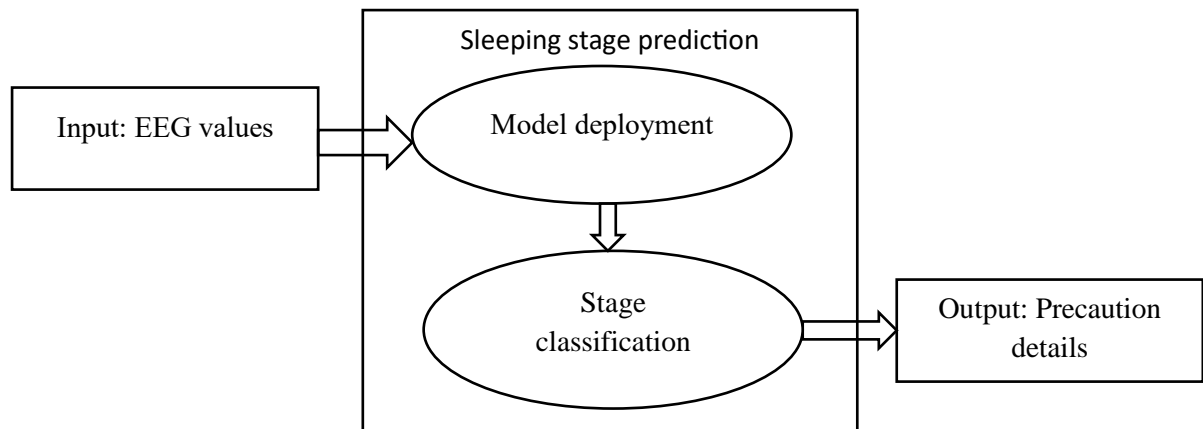


Figure 4.5 Prediction

## 4.3 ALGORITHM

### 4.3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) A CNN is a multilayer perceptron designed specifically to recognize two dimensional shapes with a high degree of invariance to translation, scaling, skewing, and other forms of distortion. Learning section of this classifier is done in supervised method which includes the following structure:

- Feature extraction
- Feature mapping
- Subsampling

The weights in all layers of a CNN are learned through training. Also, the network learns to extract its own features automatically

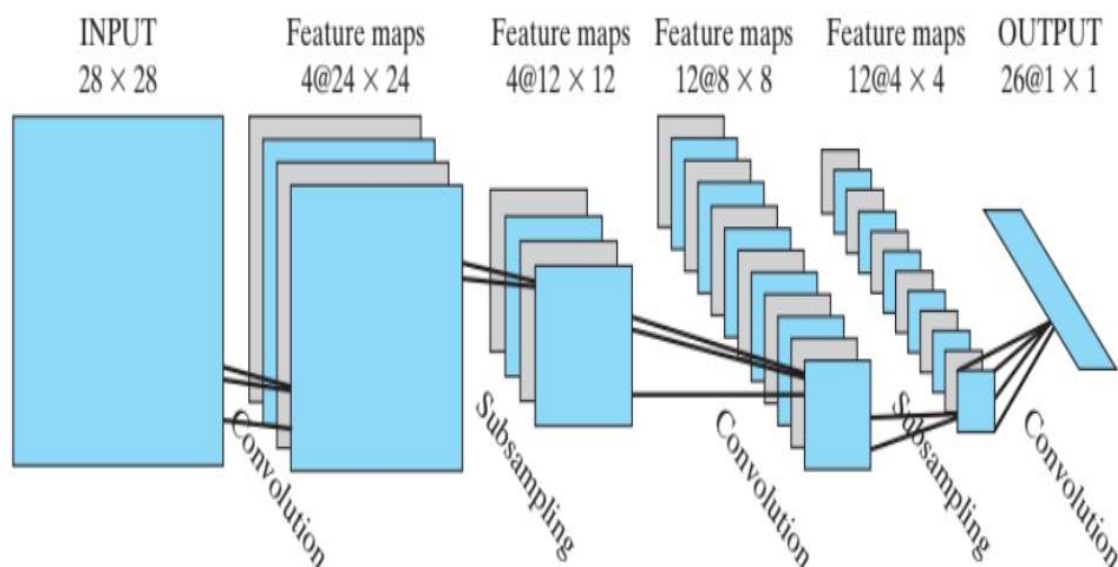


Figure 4.6 Convolutional neural network framework

This diagram shows the multiple layers such as input layer, four hidden layers, and an output layer. This network is designed to perform image processing. The input layer consists of  $28 \times 28$  sensor neurones, receives the images of different characters



that have been approximately centred and normalized in size. We can use the CNN algorithm framework to classify multiple sleeping stages.

### 4.3.2 Applications

Sleep stage classification has several applications in the field of sleep medicine and research, including:

- **Sleep disorder diagnosis:** Accurate sleep stage classification can help diagnose sleep disorders, such as sleep apnea, insomnia, and narcolepsy, which can have a significant impact on a person's health and quality of life.
- **Sleep quality assessment:** Sleep stage classification can be used to assess the quality of a person's sleep, which can be useful in evaluating the effectiveness of treatments for sleep disorders or other conditions that can affect sleep.
- **Sleep tracking and monitoring:** Sleep stage classification can be used to track and monitor a person's sleep patterns over time, which can provide insights into sleep hygiene, lifestyle factors, and other factors that can affect sleep.
- **Athletic performance optimization:** Sleep stage classification can be used to optimize athletic performance by identifying the best times for training and recovery based on an individual's sleep patterns.
- **Personalized medicine:** Sleep stage classification can be used to develop personalized treatment plans for sleep disorders, based on an individual's unique sleep patterns and physiology.

### 4.3.3 Advantages

- Reduce the time and computational steps
- False positive rate is low
- Handle the large number datasets
- Layer based classification

## CHAPTER 5

### 5. IMPLEMENTATION

#### 5.1 DATASETS

Sleep is the brain's primary function and plays a fundamental role in individual performance, learning ability, and physical movement. One of the essential physiological processes of humans is sleep vital for physical and cognitive well-being and resurgence. Sleep is a reversible state in which the eyes are closed, and several nerve centres are disabled. Sleep creates partial or unique or full anaesthesia for the individual, in which case the brain becomes a less complicated network. The gold standard of sleep analysis, Polysomnography (PSG), includes measurements of several body functions, including brain activity and heart rhythm. The sleep stages are then manually classified. These factors cause the method to have high accuracy, but also makes it costly. The fact that the procedure is usually conducted in a sleep laboratory or hospital can have a negative impact on the sleep quality of the subject, in addition to the discomfort of wearing the equipment. This is a weakness because the goal is usually to analyse the normal sleep patterns of the subject. Another way to analyse sleep is actigraphy. An actigraphy is a body-worn sensor, consisting of a three-dimensional accelerometer and possibly other sensors. The body movements data can be analysed in a variety of ways. Because of the inexpensive equipment and low intrusiveness of the method, it is preferable in some situations, for instance when the subjects are children, when data collection for several days is necessary, or when a large group of people is participating in a study. In this research work, an effective and robust method is applied to classify the sleep stages automatically based on the selected optimal set of features with an ensemble learning stacking model. The main purpose of this study is to analyse the effectiveness of selected features with a combination of ensemble learning for multi-class sleep stages classification problems. The proposed approach considers two different categories of sleep recordings which include subjects' effects with different types of sleep related disorders and the other category is subject

having complete healthy control. The proposed work is divided into two phases, in the first phase identifying the suitable features from the extracted feature vector through obtaining different selection algorithms. In the second phase, an ensemble learning stacking model is considered for the classification of the sleep stages

Today, PSG is done to identify various disorders based on the analysis of sleep stages, the main component of which is the measurement of brain activity with EEG signal. Sleep disorders include several disorders associated with various symptoms such as insomnia, respiratory disorders, behavioural and motor-related sleep disorders that significantly affect EEG signals. By classifying the different stages of sleep, the correct diagnosis of sleep-related disorders can be achieved. After recording the EEG signal, feature extraction, and analysis of the signal recorded in a specified range, a classification algorithm to identify the sleep stage is used. However, improving the classification accuracy and reducing complexity are two main challenges in the classification of sleep stages. Fig 1 shows the different stages of sleeping cycles.

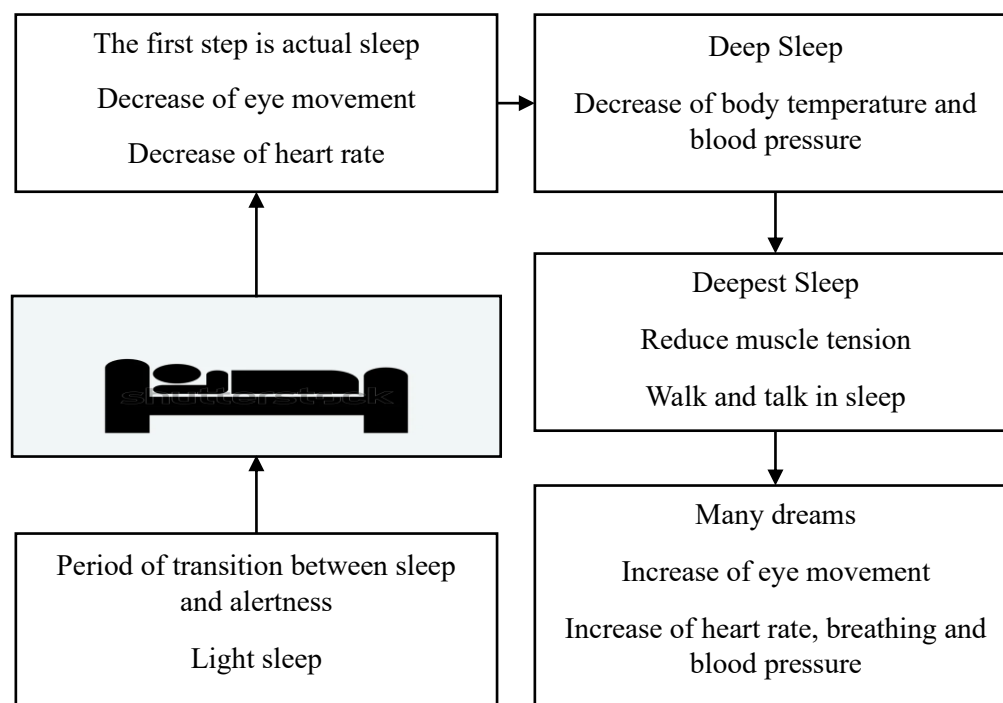


Figure 5.1 Different stages of Sleep

The EEG datasets created in the Python framework and connected to different stages of sleep are plotted in this investigation. The various classification methods are evaluated and contrasted using accuracy as the primary criterion. Repeated random sub-sampling validation is used to assess the effectiveness of the sleep stage classification.

## **5.2 HARDWARE REQUIREMENTS**

- Processor : Intel core processor 2.6.0 GHZ
- RAM : 4 GB
- Hard disk : 160 GB
- Compact Disk : 650 Mb
- Keyboard : Standard keyboard
- Monitor : 15 inch color monitor

## **5.3 SOFTWARE REQUIREMENTS**

- Operating System : Windows OS
- Front-End : python
- IDE : py charm
- Libraries : Tensorflow, keras

## **5.4 FEATURES OF PYTHON**

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages due to its simplicity, readability, and versatility. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. Python also has a large and active community of developers who contribute to open source

libraries and packages that extend its capabilities. Python is an interpreted language, which means that it is executed line-by-line by an interpreter rather than compiled into machine code like C or C++. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch. It is also commonly used for scripting and automation tasks due to its ease of use and readability. Overall, Python is a powerful and versatile programming language that is widely used in a variety of domains due to its simplicity, ease of use, and active community.



Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard

library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture. "Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one and preferably only one obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity.[ When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar. A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is *pythonic* is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms

with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

Python also has a large and active community of developers who contribute to a wide range of open-source libraries and tools, making it easy to find and use pre-built code to solve complex problems.

Python has a wide range of applications, including:

**Data Science:** Python is one of the most popular languages for data science, thanks to libraries like NumPy, Pandas, and Matplotlib that make it easy to manipulate and visualize data.

**Machine Learning:** Python is also widely used in machine learning and artificial intelligence, with libraries like TensorFlow, Keras, and Scikit-learn that provide powerful tools for building and training machine learning models.

**Web Development:** Python is commonly used in web development, with frameworks like Django and Flask that make it easy to build web applications and APIs.

**Scientific Computing:** Python is used extensively in scientific computing, with libraries like SciPy and SymPy that provide powerful tools for numerical analysis and symbolic mathematics.

In addition to its versatility and ease of use, Python is also known for its portability and compatibility. Python code can be run on a wide range of platforms, including Windows, macOS, and Linux, and it can be integrated with other languages like C and Java.

Overall, Python is a powerful and versatile programming language that is well-suited for a wide range of applications, from data science and machine learning to web development and scientific computing. Its simplicity, readability, and large community of developers make it an ideal choice for beginners and experts alike.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.

2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.



One of the strengths of Python is its rich ecosystem of third-party libraries and tools. These libraries provide a wide range of functionality, from scientific computing and data analysis to web development and machine learning. Some popular Python libraries and frameworks include:

**NumPy:** A library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

**Pandas:** A library for data manipulation and analysis in Python, providing support for reading and writing data in a variety of formats, as well as powerful tools for manipulating and analyzing data.

**Matplotlib:** A plotting library for Python that provides a variety of visualization tools, including line plots, scatter plots, bar plots, and more.

**TensorFlow:** An open-source machine learning library for Python that provides a variety of tools and algorithms for building and training machine learning models.

**Django:** A popular web framework for Python that provides a full-stack framework for building web applications, with support for everything from url routing to user authentication and database integration.

Python's popularity has also led to a large and active community of developers who contribute to open-source projects and share code and resources online. This community provides a wealth of resources for learning Python, including tutorials, online courses, and forums for asking and answering questions.

## 5.5 TENSORFLOW LIBRARIES IN PYTHON

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is one of the most popular libraries for building and training machine learning models, especially deep neural networks. TensorFlow allows developers to build complex models with ease, including image and speech

recognition, natural language processing, and more. One of the key features of TensorFlow is its ability to handle large-scale datasets and complex computations, making it suitable for training deep neural networks. It allows for parallelization of computations across multiple CPUs or GPUs, allowing for faster training times. TensorFlow also provides a high-level API called Keras that simplifies the process of building and training models.

TensorFlow offers a wide range of tools and libraries that make it easy to integrate with other Python libraries and frameworks. It has built-in support for data preprocessing and visualization, making it easy to prepare data for training and analyze model performance. One of the major advantages of TensorFlow is its ability to deploy models to a variety of platforms, including mobile devices and the web. TensorFlow Lite is a mobile-optimized version of TensorFlow that is designed for deploying models on Android, iOS, and other mobile platforms. TensorFlow.js is a JavaScript library that allows for training and deployment of models directly in the browser. TensorFlow provides a wide range of features and tools for building and training machine learning models. Some of the key features of TensorFlow include

**Graph-based computation:** TensorFlow uses a graph-based computation model, which allows for efficient execution of computations across multiple devices and CPUs/GPUs.

**Automatic differentiation:** TensorFlow provides automatic differentiation, which allows for efficient computation of gradients for use in backpropagation algorithms. **High-level APIs:** TensorFlow provides high-level APIs, such as Keras, that allow developers to quickly build and train complex models with minimal code.

**Pre-processing and data augmentation:** TensorFlow provides a range of tools for preprocessing and data augmentation, including image and text preprocessing, data normalization, and more.

**Distributed training:** Distributed training in TensorFlow is a technique that involves training a machine learning model using multiple devices, such as CPUs or GPUs, working together in a coordinated manner. By leveraging the capabilities of distributed systems, TensorFlow enables faster training times and more efficient utilization of resources. There are two main approaches to distributed training: data parallelism and model parallelism. In data parallelism, the training data is divided among the devices, which process subsets of the data simultaneously and exchange gradients to collectively update the model parameters

**Model deployment:** Model deployment in TensorFlow refers to the process of making trained models available for use in real-world applications on various platforms. TensorFlow provides features and tools that simplify the deployment process, enabling models to be deployed on a range of platforms, such as mobile devices and the web. When it comes to deploying TensorFlow models, several options are available. One common approach is to convert the trained model into a format that is optimized for inference, such as TensorFlow Saved Model or TensorFlow Lite. Saved Model is a universal format that allows models to be deployed across different TensorFlow platforms, while TensorFlow Lite is specifically designed for efficient deployment on resource-constrained devices like mobile phone.

**Visualization tools:** Visualization tools are essential for analyzing and understanding the performance and behaviour of machine learning models. TensorFlow offers a variety of visualization tools that facilitate model analysis and debugging. One of the most popular tools is Tensor Board, which provides real-time visualizations of training metrics, model architectures, and other important information. Tensor Board allows users to visualize various aspects of their TensorFlow models. It provides interactive dashboards that display metrics such as loss, accuracy, and learning curves, helping users track the progress of their models during training

## 5.6 UML DIAGRAMS

### ACTIVITY DIAGRAM

An activity diagram is a type of UML (Unified Modelling Language) diagram used to describe the flow of activities and actions in a system or process. It depicts the steps and decisions involved in a particular activity or process and shows the interactions between different actors, objects, or system components. Activity diagrams are often used to model the behaviour of software systems, business processes, or workflows. They are a visual representation of the steps required to complete a task or achieve a specific goal, and can be used to communicate complex processes to stakeholders.

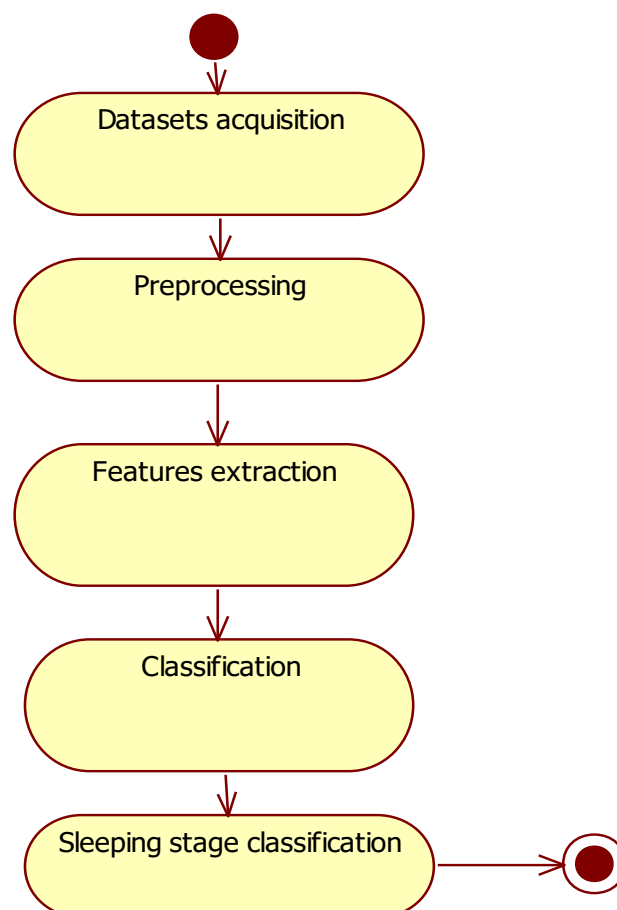


Figure 5.2 ACTIVITY DIAGRAM

## USECASE DIAGRAM

A use case diagram is a type of UML (Unified Modelling Language) diagram that is used to represent the functional requirements of a system or software application. It provides a high-level view of the system's functionality and the actors or users who interact with it. In a use case diagram, use cases are represented by ovals, and actors are represented by stick figures or icons. The use cases describe the various interactions that the actors can have with the system, and the relationships between the actors and the use cases.

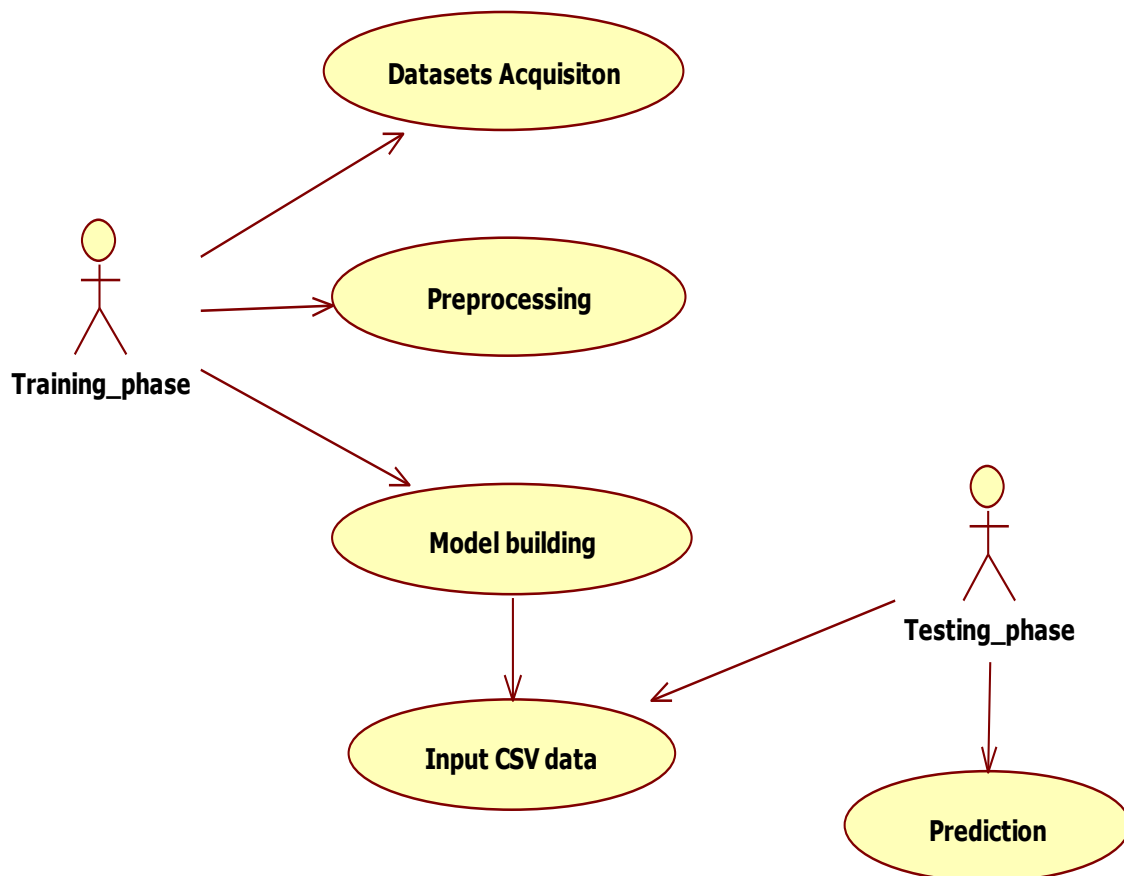


Figure 5.3 USECASE DIAGRAM

## SEQUENCE DIAGRAM

A sequence diagram is a type of UML (Unified Modelling Language) diagram that is used to visualize the interactions between objects or components in a system or software application. It depicts the sequence of messages exchanged between objects and the order in which they occur. In a sequence diagram, objects are represented by rectangles, and the messages exchanged between objects are represented by arrows. The objects are listed vertically on the left-hand side of the diagram, and the messages are listed horizontally across the top of the diagram.

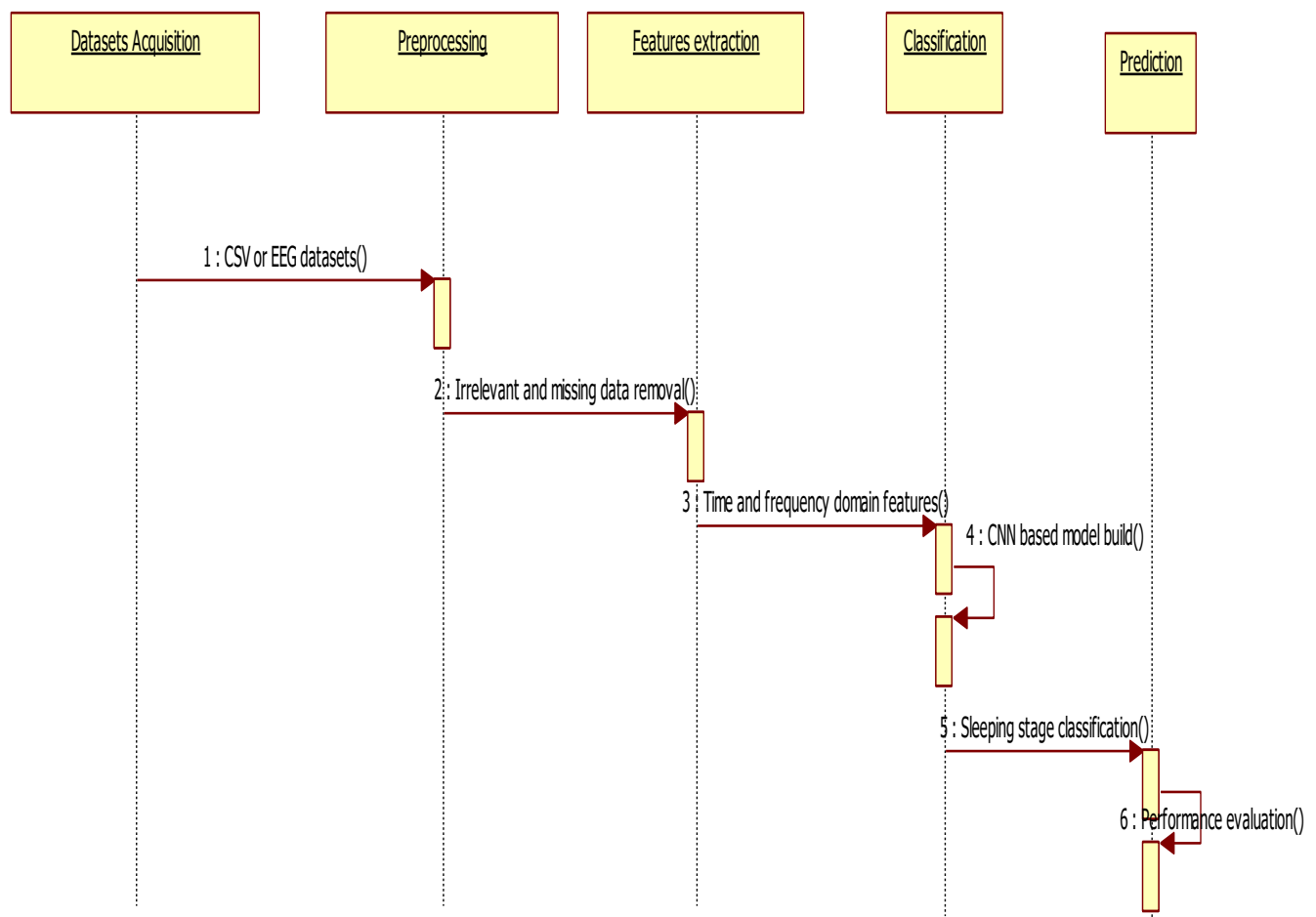


Figure 5.4 SEQUENCE DIAGRAM

## CLASS DIAGRAM

A class diagram is a type of UML (Unified Modelling Language) diagram used to represent the static structure of a system or software application. It shows the classes and interfaces that make up the system, as well as the relationships and dependencies between them.

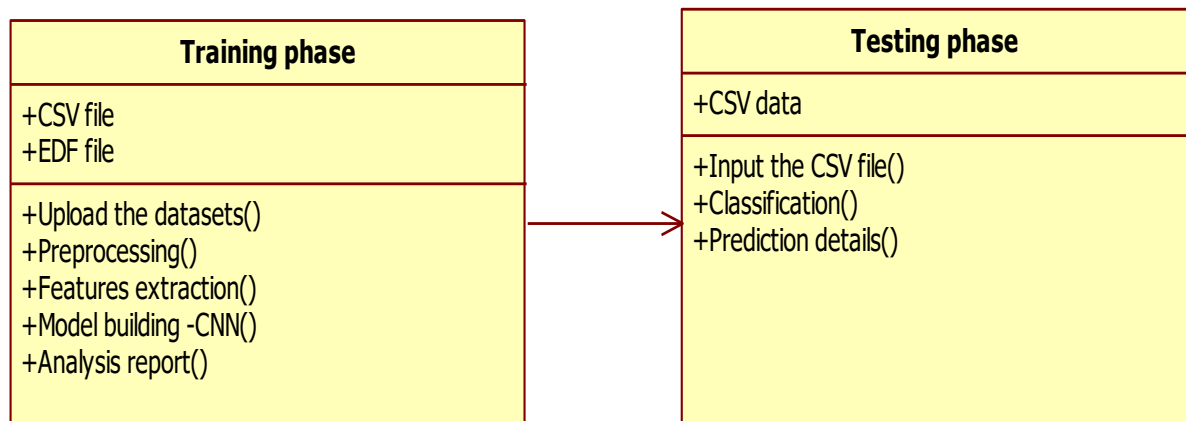


Figure5.5 CLASS DIAGRAM

## COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). Developers can use these diagrams to portray the dynamic behaviour of a particular use case and define the role of each object.

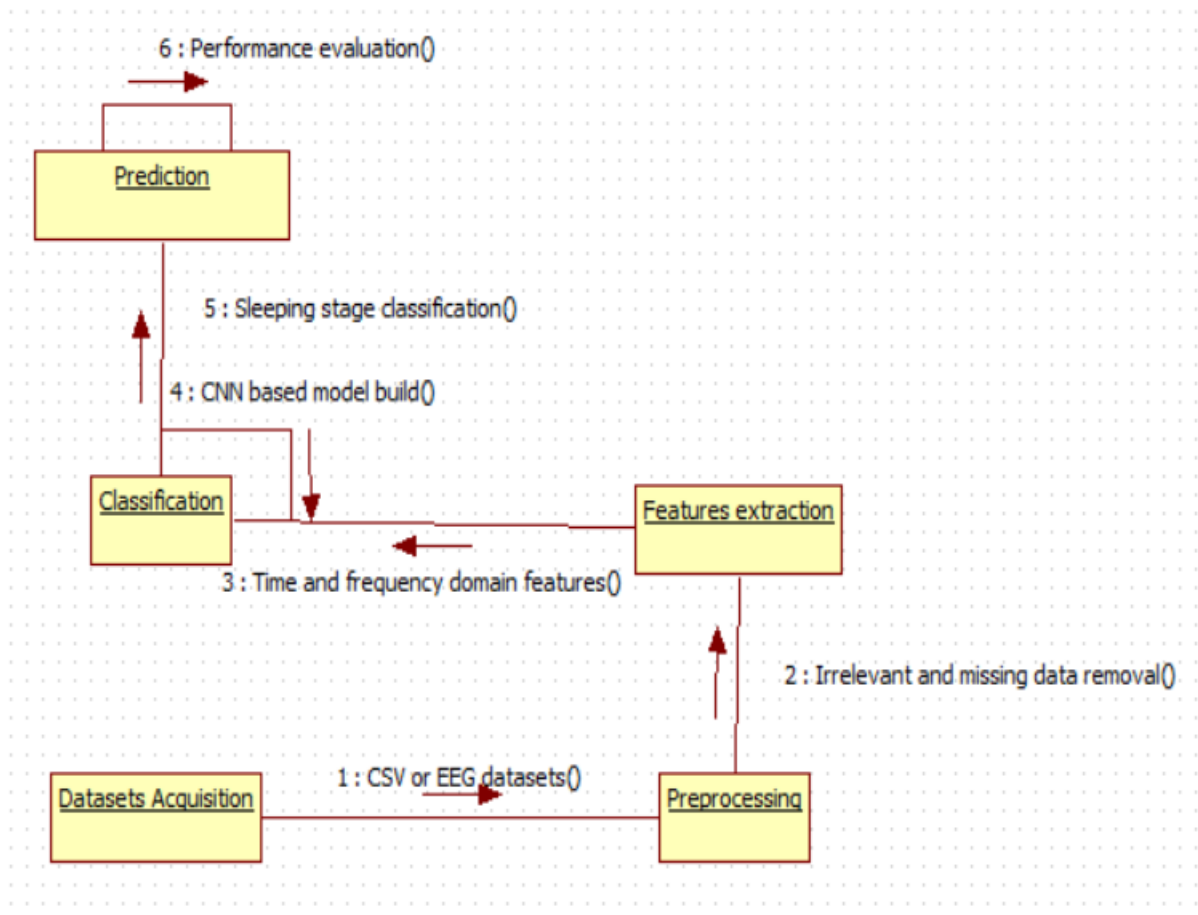


Figure5.6 COLLABORATION DIAGRAM



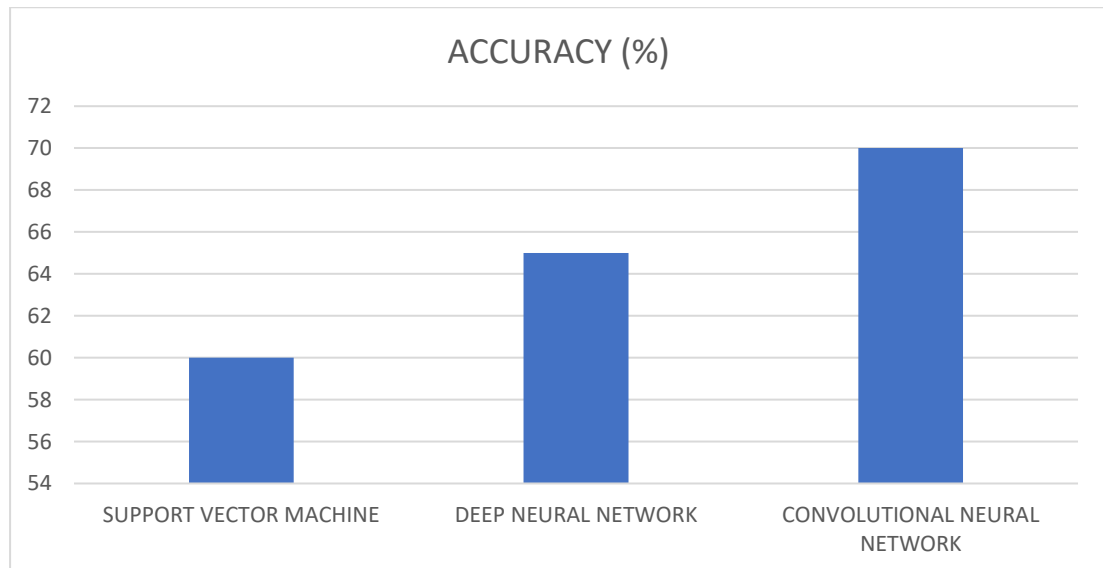
## CHAPTER 6

### 6. RESULTS AND ANALYSIS

In this study we can input the EEG datasets related to sleeping stages and developed in Python framework. Accuracy as main criterion is considered for evaluating and comparing the different classification methods. The performance of the sleep stage classification is evaluated using repeated random sub-sampling validation. To measure the classification accuracy, the overall accuracy value is calculated as follows

$$\text{ACCURACY} = \frac{\text{Number of true detections}}{\text{Total number of epochs}}$$

ALGORITHMS	ACCURACY (%)
SUPPORT VECTOR MACHINE	60
DEEP NEURAL NETWORK	65
CONVOLUTIONAL NEURAL NETWORK	70



**Fig 6.1 Accuracy**

From the diagram, convolutional neural network provides the improved accuracy than the existing frameworks

## **CHAPTER 7**

### **7. CONCLUSION AND FUTURE WORK**

#### **7.1 CONCLUSION**

In conclusion, using a CSV file for sleep stage classification in EEG datasets can provide valuable information about an individual's sleep patterns. By loading the EEG data and CSV file into memory and pre-processing the data to extract relevant features, a labelled dataset of epochs can be created that can be used to train and evaluate machine learning algorithms. A CNN is a common machine learning algorithm used for EEG classification, which can learn the relationship between the EEG features and the sleep stages. The performance of the CNN can be evaluated using metrics such as accuracy, precision, recall, and F1 score. The trained CNN can then be used to predict the sleep stage in new EEG datasets by dividing the data into epochs and classifying each epoch based on its EEG features. Overall, CSV file classification for EEG datasets provides a powerful tool for sleep researchers and clinicians to analyse large amounts of sleep data quickly and accurately. It can help to identify sleep disorders, monitor treatment progress, and provide insights into the mechanisms of sleep.

#### **7.2 FUTURE ENHANCEMENT**

While sleeping stage classification using a CSV file and EEG datasets has shown promising results, there is still room for future work to improve the accuracy and efficiency of the classification. One area for future work is to explore more advanced machine learning techniques beyond CNNs. For example, recurrent neural networks (RNNs) could be used to incorporate temporal information in the classification process, as sleep stages are known to have temporal dependencies. Furthermore, hybrid models that combine CNNs and RNNs could also be explored to take advantage of the strengths of both approaches.

## SCREENSHOTS

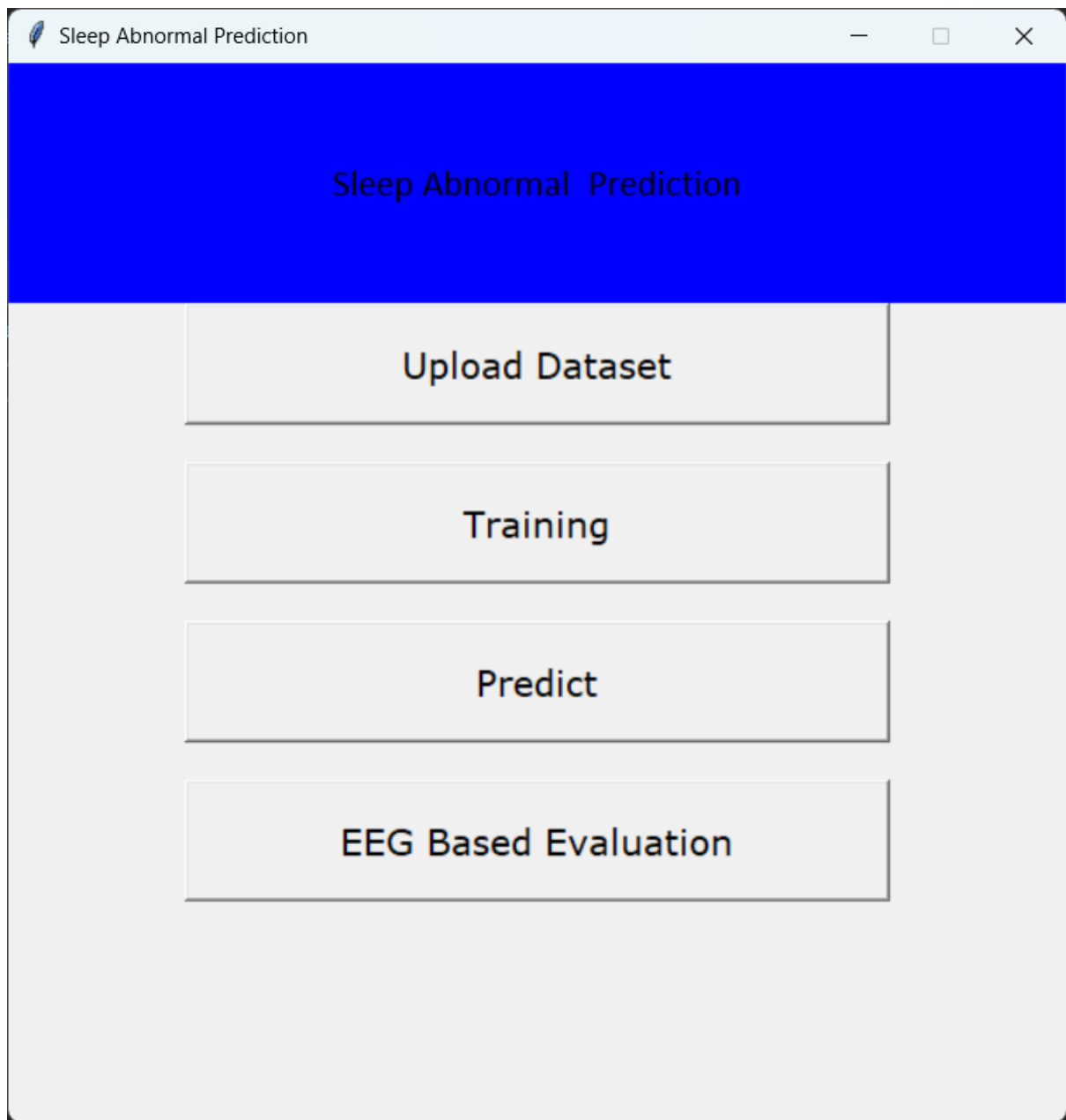


FIGURE 7.1 HOME PAGE

## DATASETS

Number of records of Non sleep 9200 VS sleep 2300

	count	mean	std	min	25%	50%	75%	max
X1	9200.0	-8.992609	70.455286	-566.0	-44.0	-7.0	26.0	1726.0
X2	9200.0	-8.877174	70.560110	-609.0	-44.0	-7.0	27.0	1713.0
X3	9200.0	-8.910435	70.372582	-594.0	-45.0	-7.0	28.0	1697.0
X4	9200.0	-8.969783	70.030409	-549.0	-45.0	-8.0	27.0	1612.0

```

X5  9200.0 -9.085326 69.377958 -603.0 -45.0 -8.0 27.0 1437.0
...  ...      ...      ...      ...      ...      ...      ...
X175 9200.0 -9.848587 69.550894 -570.0 -45.0 -9.0 27.0 1958.0
X176 9200.0 -9.620435 70.353607 -594.0 -46.0 -8.0 27.0 2047.0
X177 9200.0 -9.395435 70.934300 -563.0 -45.0 -9.0 27.0 2047.0
X178 9200.0 -9.240435 71.185850 -559.0 -45.0 -8.0 27.0 1915.0
y    9200.0 0.000000  0.000000  0.0  0.0 0.0  0.0  0.0

```

[179 rows x 8 columns]

```

      count    mean      std   min   25%   50%   75%   max
X1  2300.0 -21.936522 342.361939 -1839.0 -193.25 -16.0 159.00 1314.0
X2  2300.0 -19.049130 343.398782 -1838.0 -191.25 -18.0 168.25 1356.0
X3  2300.0 -15.293913 337.489643 -1835.0 -187.00 -12.5 169.25 1274.0
X4  2300.0 -9.836087 332.354833 -1845.0 -184.00 -6.0 166.25 1226.0
X5  2300.0 -3.707391 332.211163 -1791.0 -174.25 -12.0 170.00 1518.0
...  ...      ...      ...      ...      ...      ...      ...
X175 2300.0 -25.830870 339.650467 -1863.0 -195.00 -14.5 153.25 1205.0
X176 2300.0 -25.043913 335.747017 -1781.0 -192.00 -18.0 150.00 1371.0
X177 2300.0 -24.548261 335.244512 -1727.0 -190.25 -21.5 151.25 1445.0
X178 2300.0 -24.016522 339.819309 -1829.0 -189.00 -23.0 157.25 1380.0
y    2300.0  1.000000  0.000000   1.0   1.00  1.0  1.00   1.0

```

[179 rows x 8 columns]

Total Mean VALUE for sleep: 290.129360958884

Total Std VALUE for sleep: 53.56315864740058

Total Mean VALUE for NON sleep: 1260.098927262616

Total Std VALUE for NON sleep: 15.561044289100993

Number of records of non-Sleep 18400 VS Sleep 18400

Normalized Total Mean VALUE for Sleep: 1150.0132102785494

Normalized Total Std VALUE for Sleep: 0.02256476904844563

Normalized Total Mean VALUE for NOT Sleep: 1150.0065271687797

Normalized Total Std VALUE for NOT Sleep: 0.002007429551034863

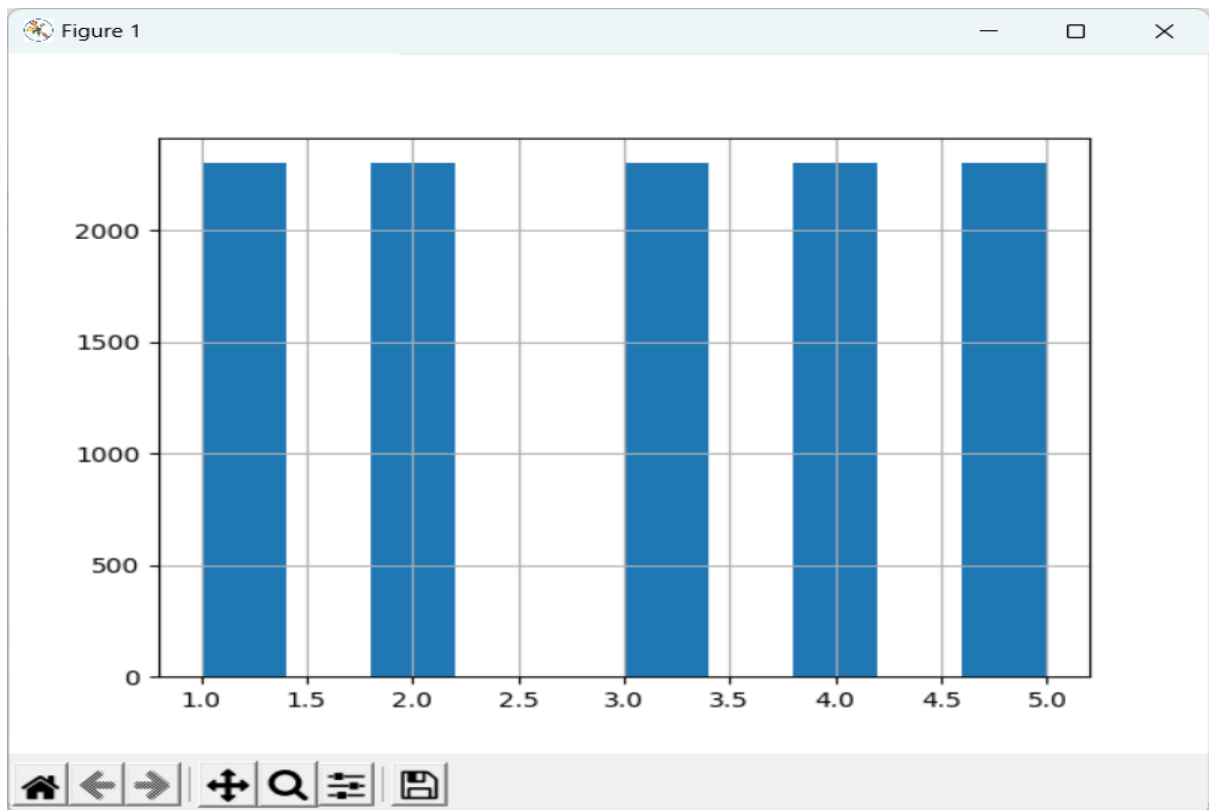


FIGURE 7.2 COUNTER PLOT

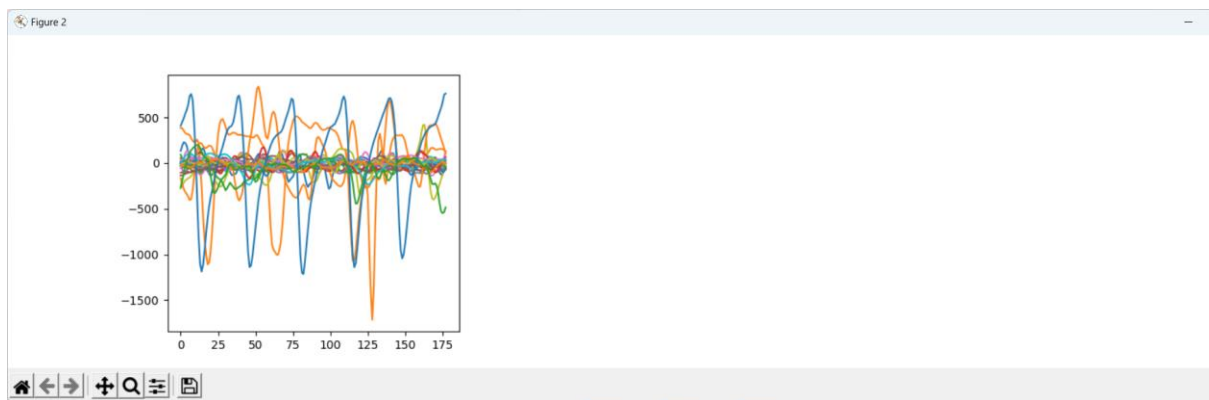


FIGURE 7.3 OVERALL CLASS DATA

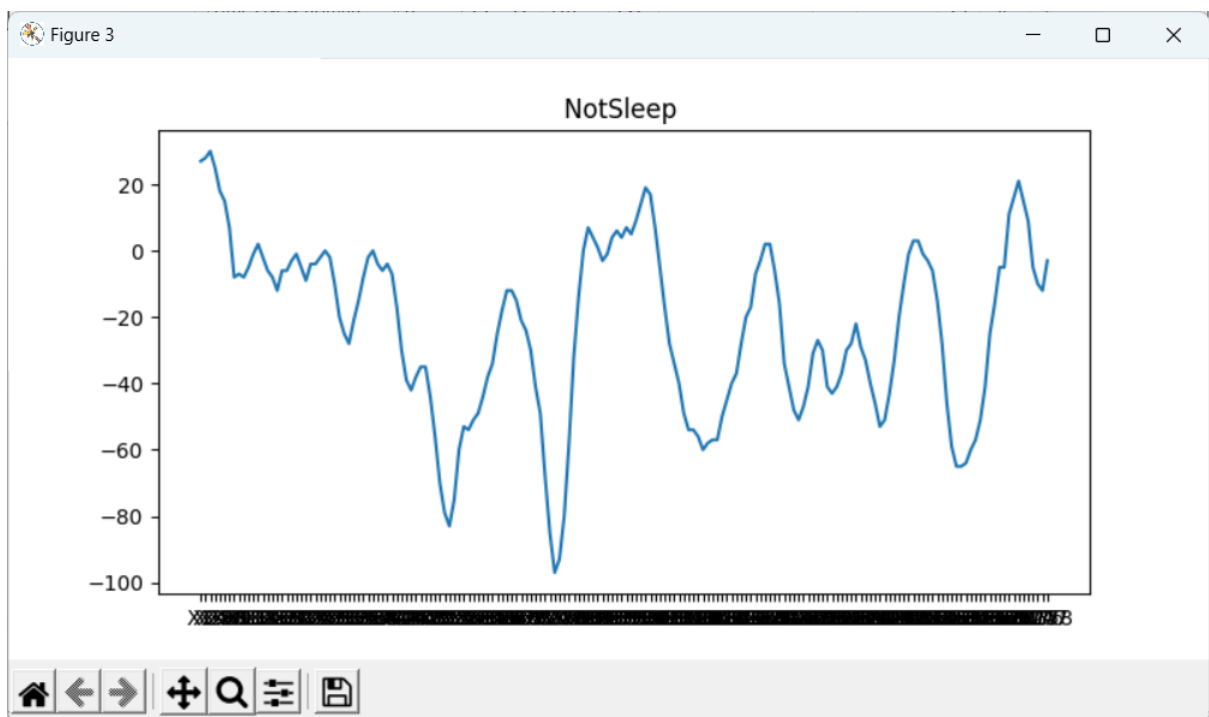


FIGURE 7.4 STAGE 1 NOT SLEEP DATA

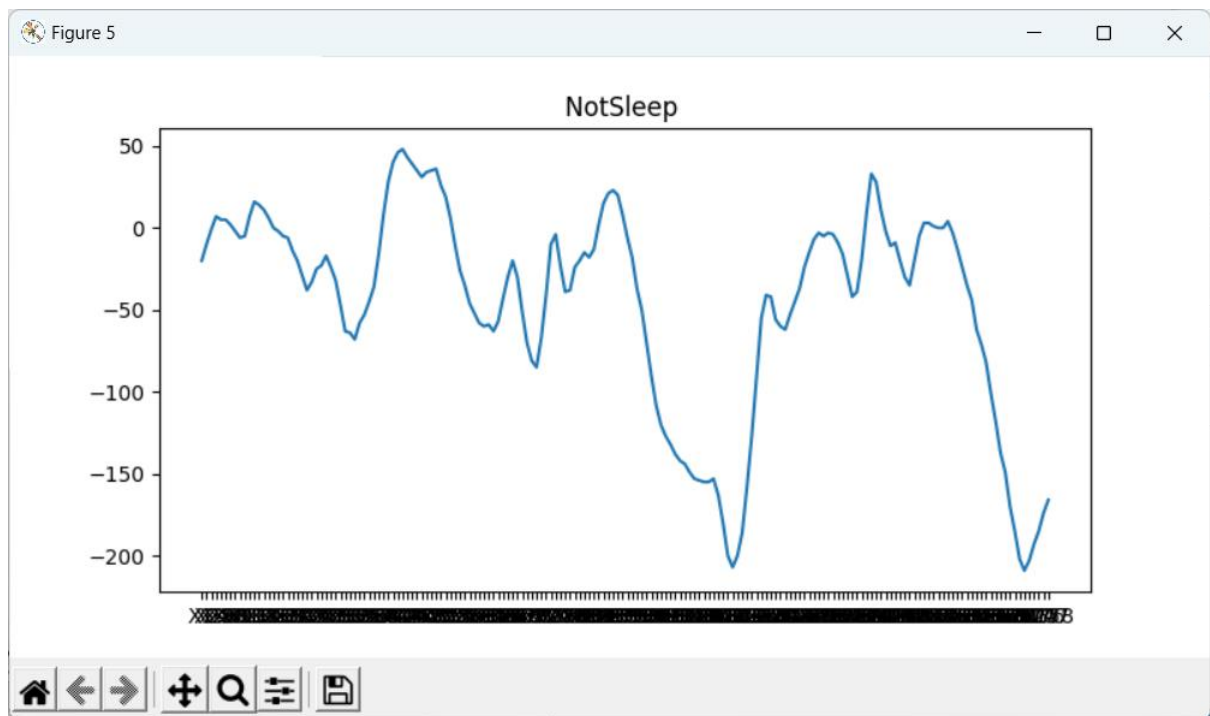


FIGURE 7.5 STAGE 2 NOT SLEEP DATA

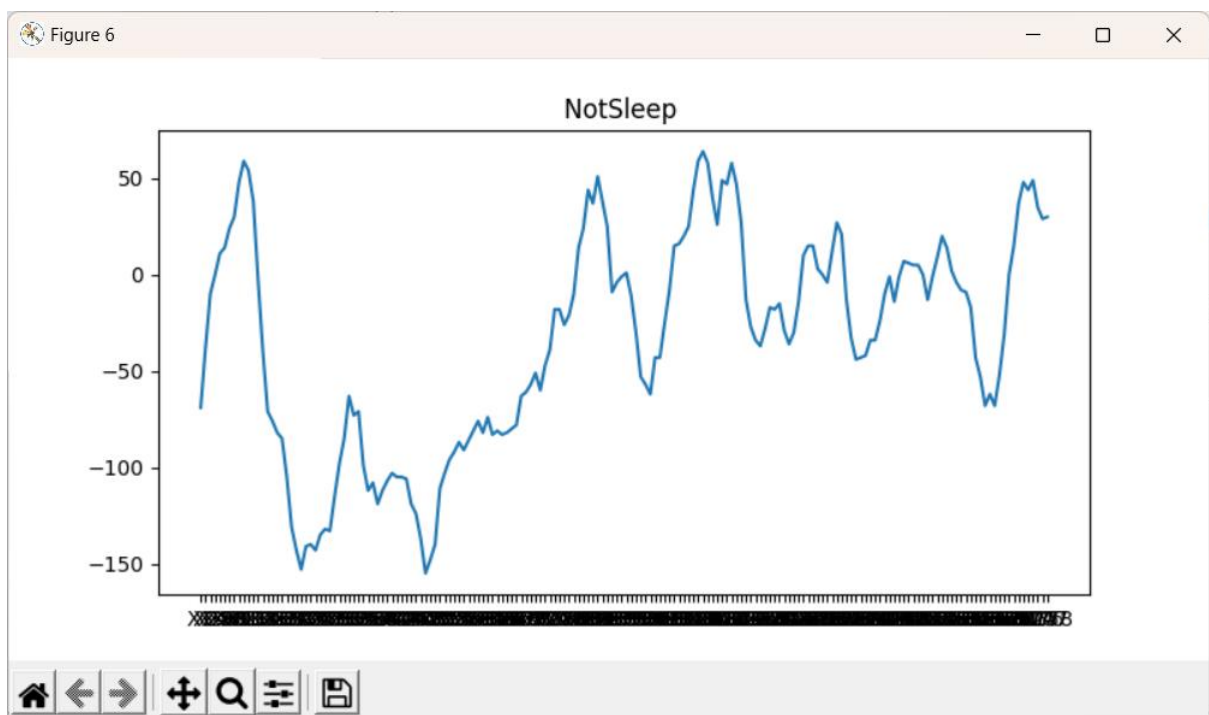


FIGURE 7.6 STAGE 3 NOT SLEEP DATA

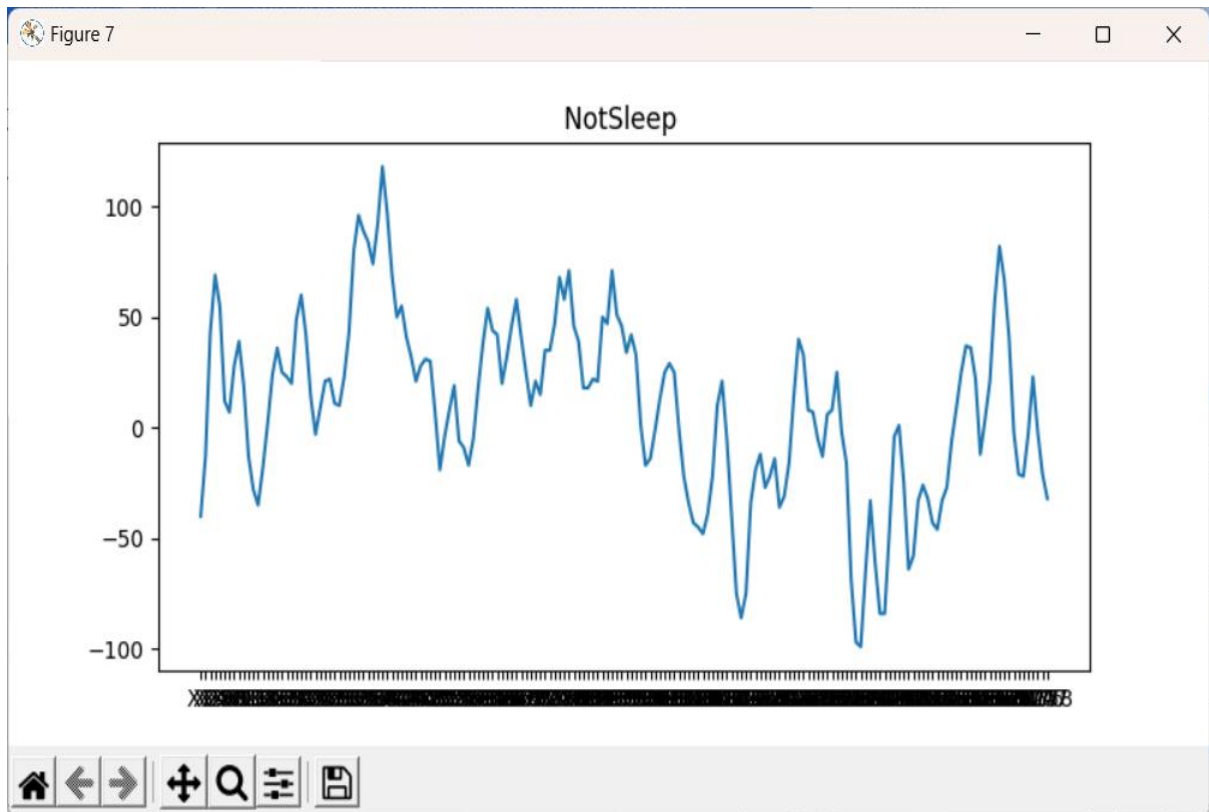


FIGURE 7.7 STAGE 4 NOT SLEEP DATA

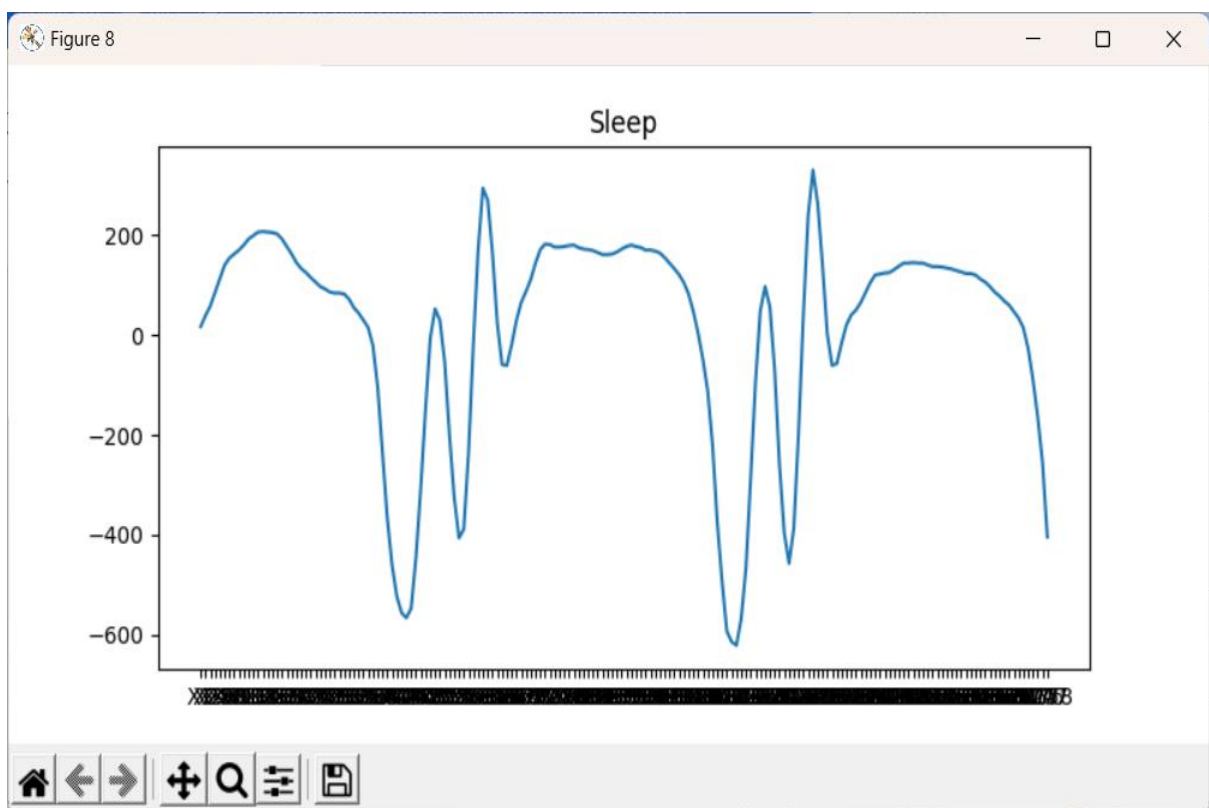


FIGURE 7.8 STAGE 1 SLEEP DATA



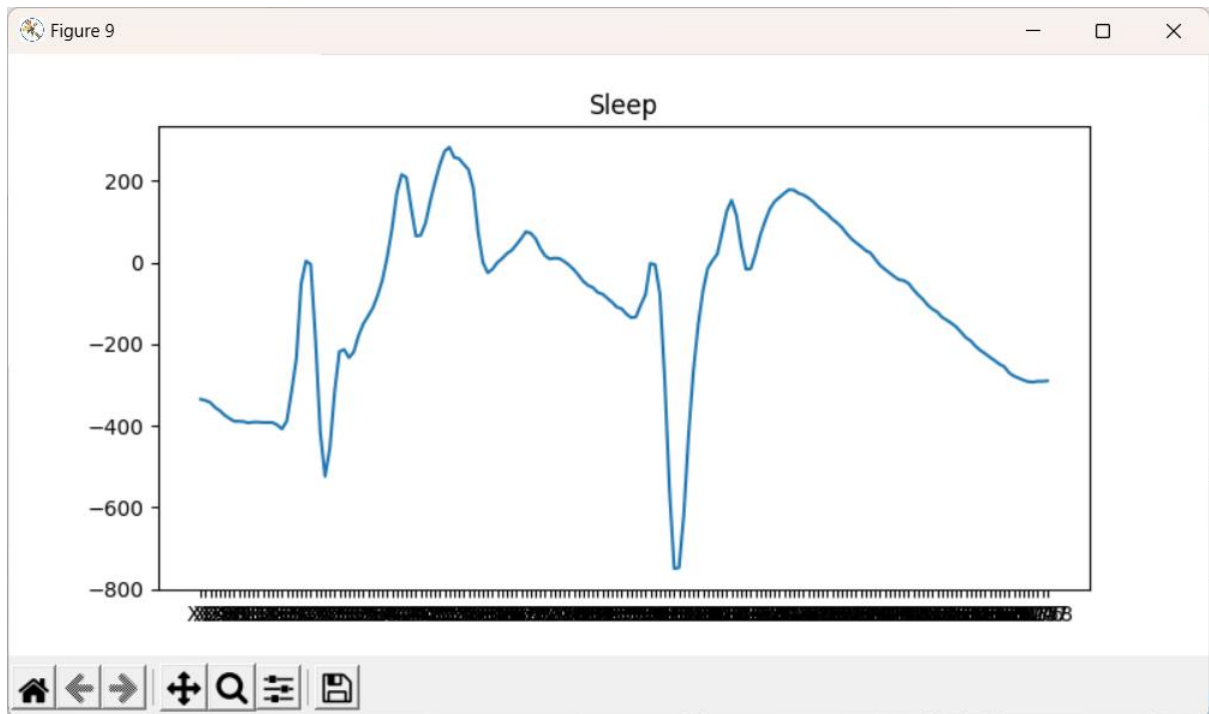


FIGURE 7.9 STAGE 2 SLEEP DATA

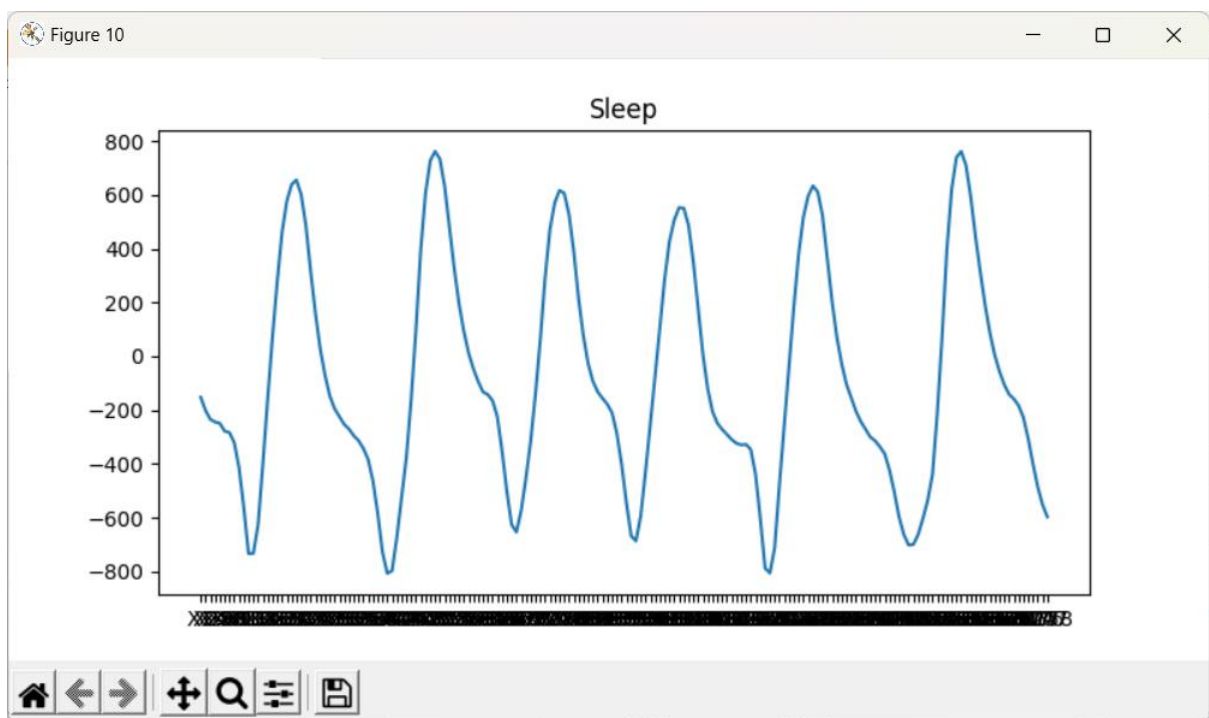


FIGURE 7.10 STAGE 3 SLEEP DATA

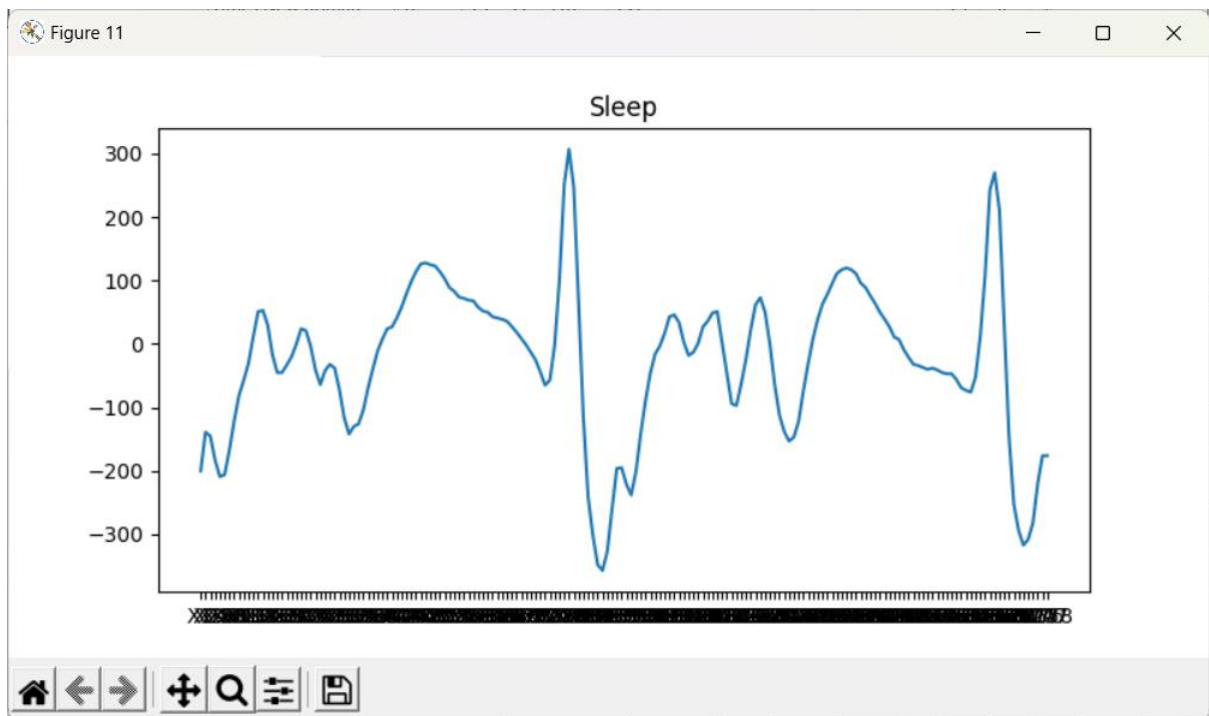


FIGURE 7.11 STAGE 4 SLEEP DATA



FIGURE 7.12 STAGE 5 SLEEP DATA

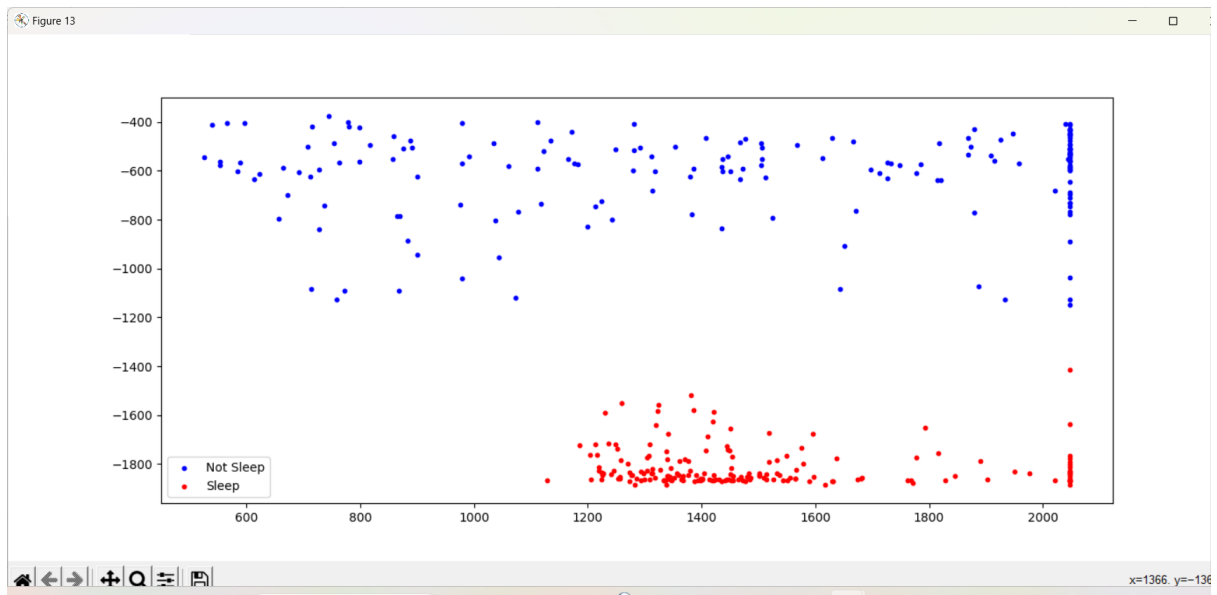


FIGURE 7.13 SCATTER MATRIX FOR SLEEP AND NON-SLEEP

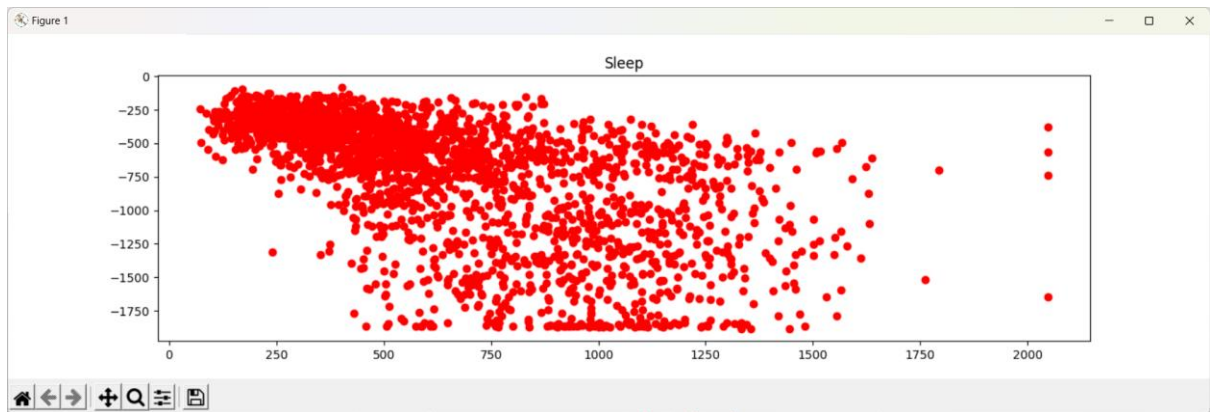


FIGURE 7.14 SCATTER MATRIX FOR SLEEP

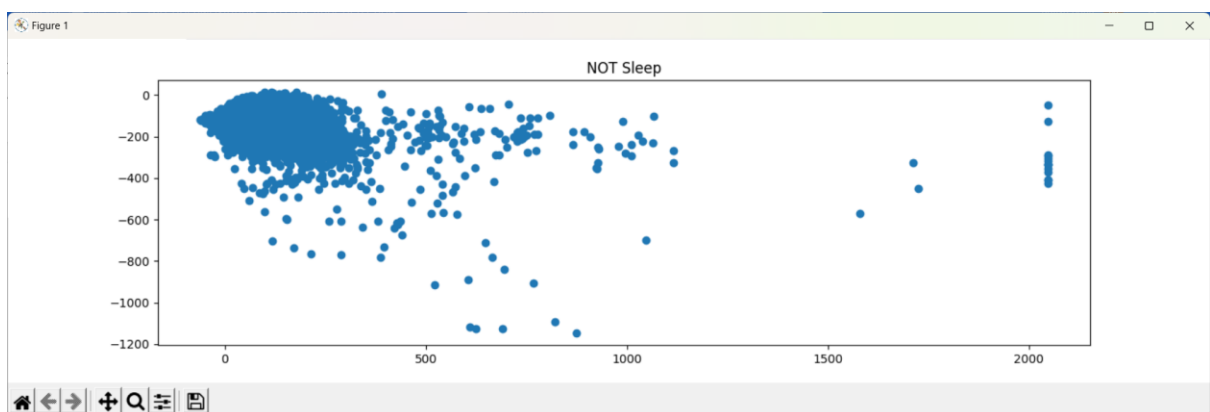


FIGURE 7.15 SCATTER MATRIX FOR NON-SLEEP

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 176, 256)	1024
dropout (Dropout)	(None, 176, 256)	0
conv1d_1 (Conv1D)	(None, 174, 128)	98432
dropout_1 (Dropout)	(None, 174, 128)	0
conv1d_2 (Conv1D)	(None, 172, 128)	49280
dropout_2 (Dropout)	(None, 172, 128)	0
conv1d_3 (Conv1D)	(None, 170, 64)	24640
dropout_3 (Dropout)	(None, 170, 64)	0
conv1d_4 (Conv1D)	(None, 168, 32)	6176
dropout_4 (Dropout)	(None, 168, 32)	0
max_pooling1d (MaxPooling1D )	(None, 84, 32)	0
flatten (Flatten)	(None, 2688)	0
dense (Dense)	(None, 5)	13445

---

---

Total params: 192,997

Trainable params: 192,997

Non-trainable params: 0

---

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.99	0.94	0.96	465
---	------	------	------	-----

1	0.63	0.69	0.66	459
---	------	------	------	-----

2	0.74	0.40	0.52	450
---	------	------	------	-----

3	0.89	0.79	0.84	457
---	------	------	------	-----

4	0.71	0.88	0.79	469
---	------	------	------	-----

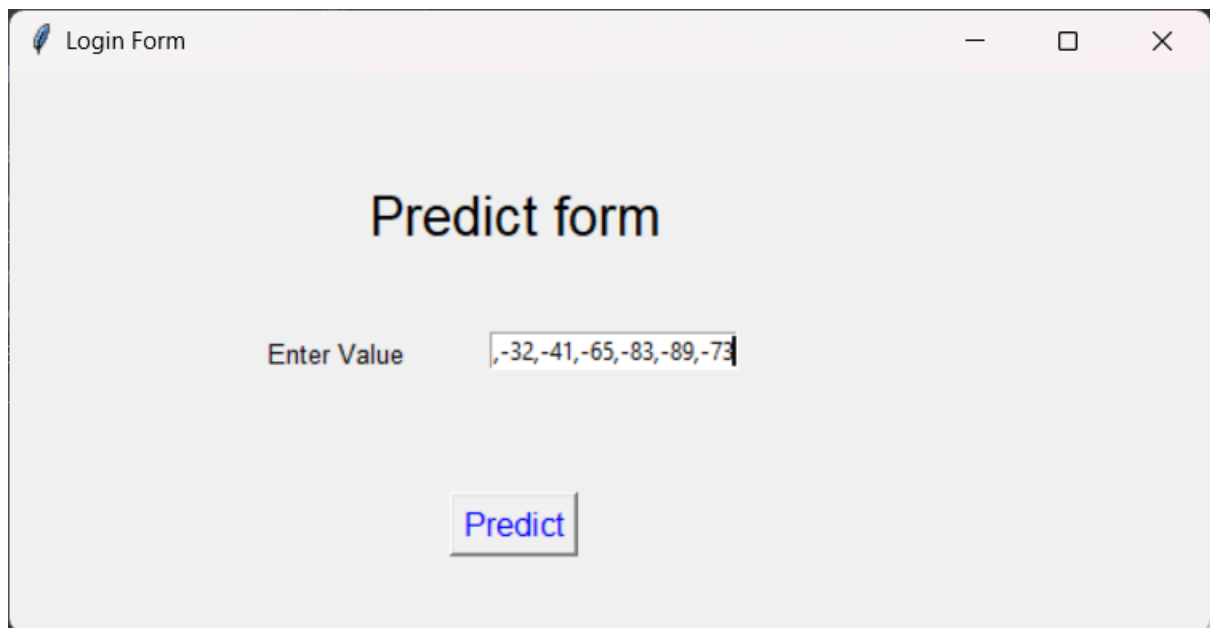
micro avg	0.78	0.74	0.76	2300
-----------	------	------	------	------

macro avg	0.79	0.74	0.75	2300
-----------	------	------	------	------

weighted avg	0.79	0.74	0.75	2300
--------------	------	------	------	------

samples avg	0.74	0.74	0.74	2300
-------------	------	------	------	------

accuracy: 76.43%



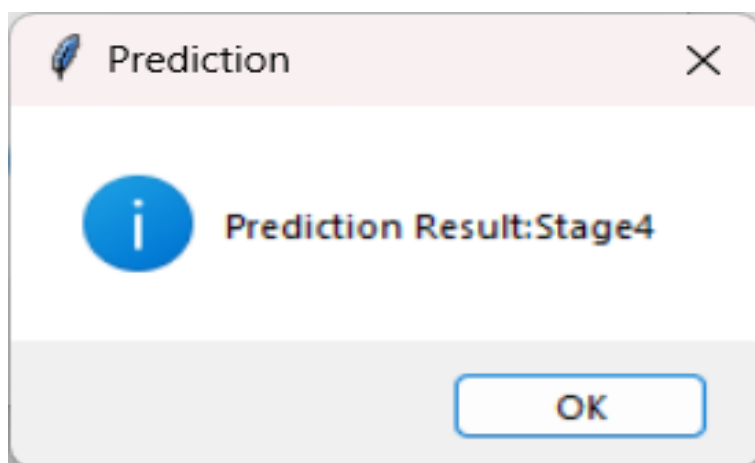
Login Form

## Predict form

Enter Value

Predict

FIGURE 7.16 PREDICTION

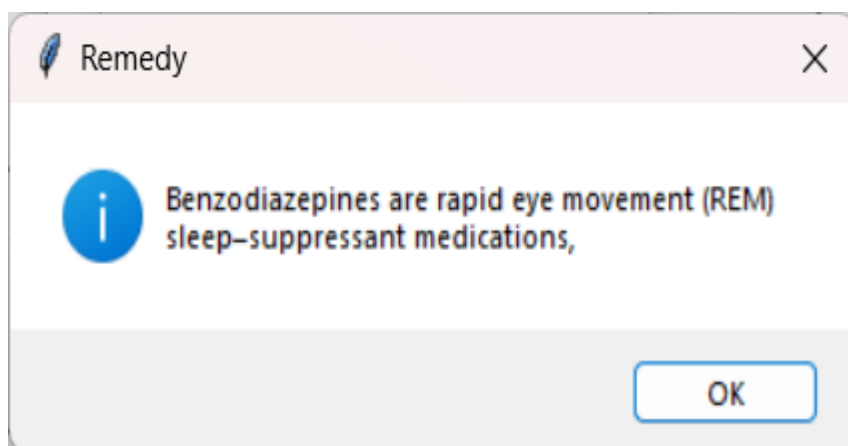


Prediction

*i* Prediction Result: Stage4

OK

FIGURE 7.17 STAGE CLASSIFICATION



Remedy

*i* Benzodiazepines are rapid eye movement (REM) sleep-suppressant medications,

OK

FIGURE 7.18 PRECAUTION DETAILS

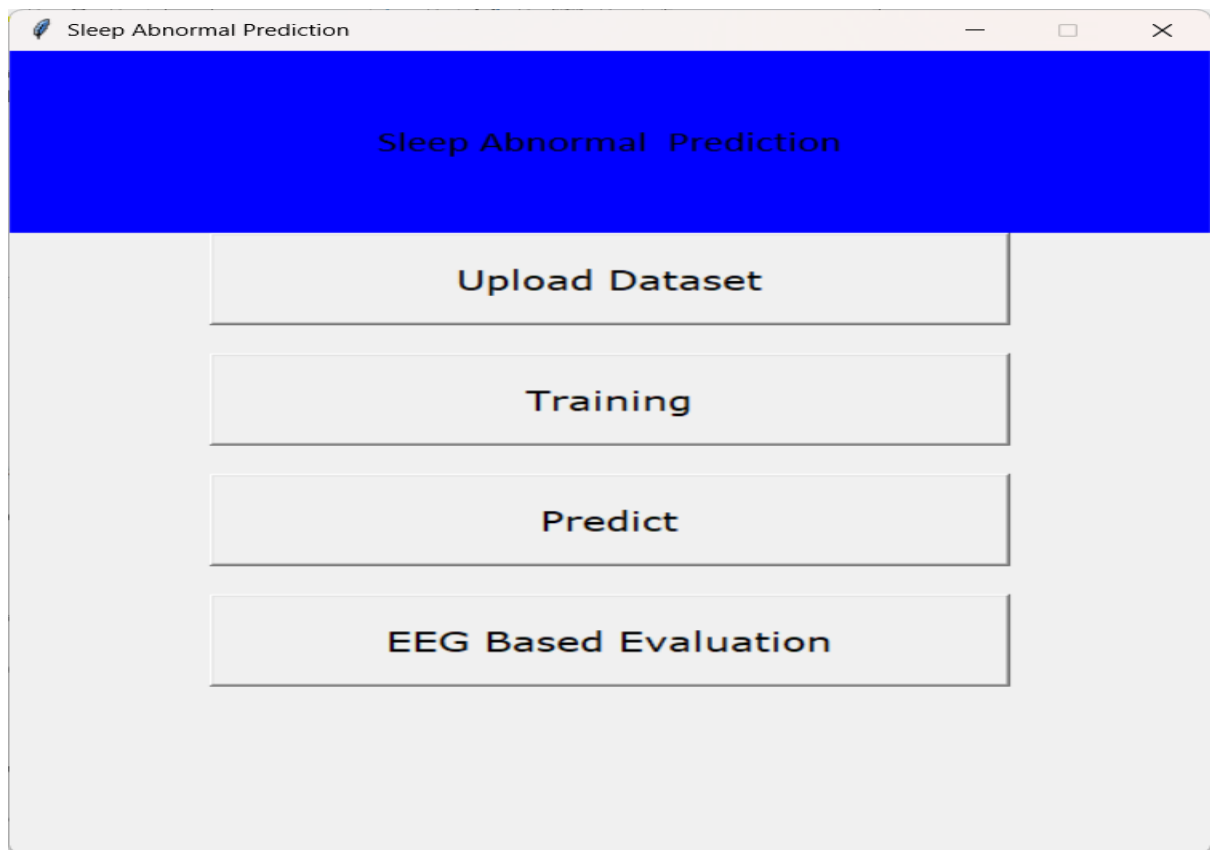


FIGURE 7.19 EEG BASED EVALUATION

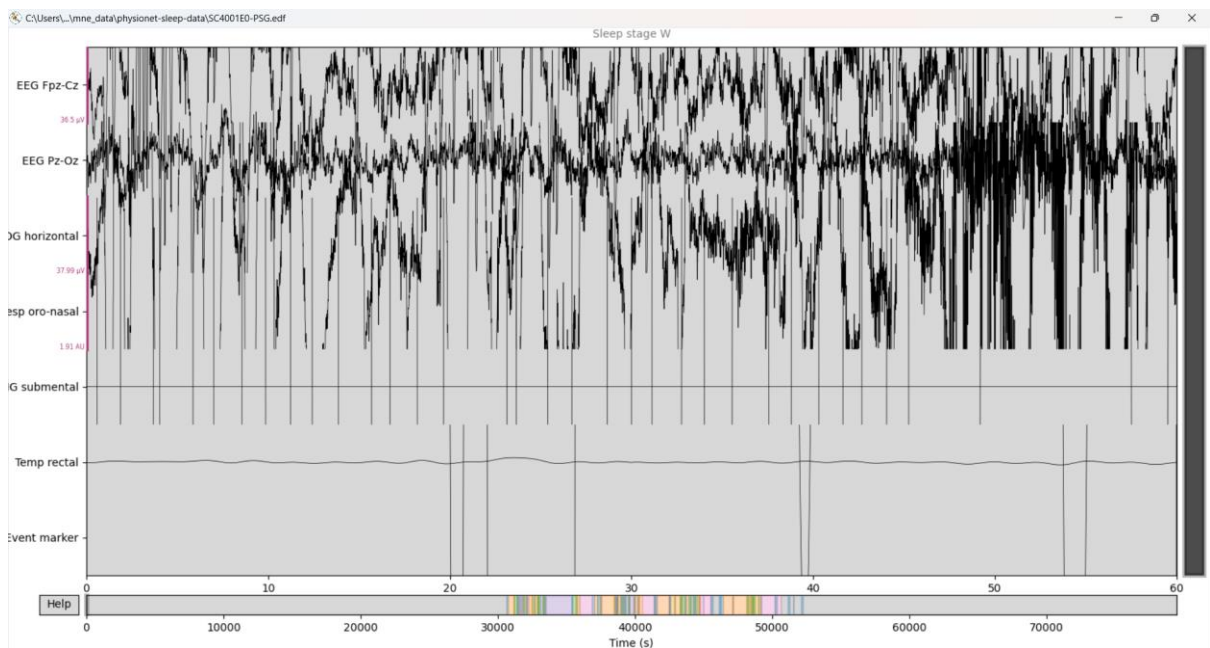


FIGURE 7.20 OVERALL EEG PLOT

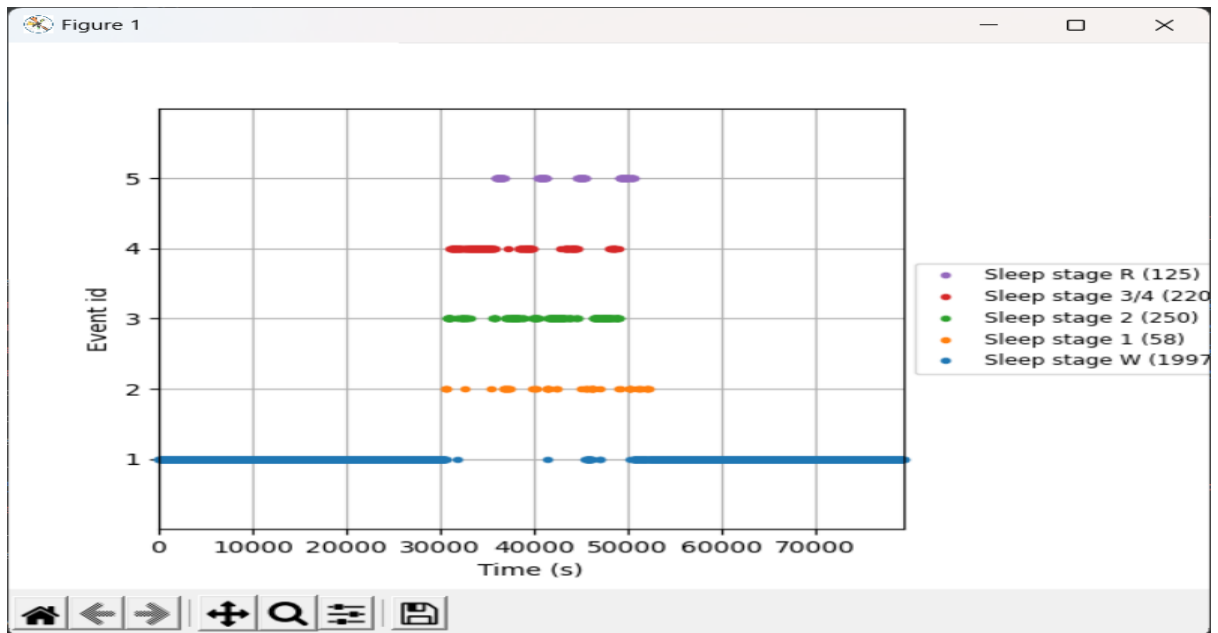


FIGURE 7.21 ANNOTATIONS OF MULTIPLE SLEEPING STAGE

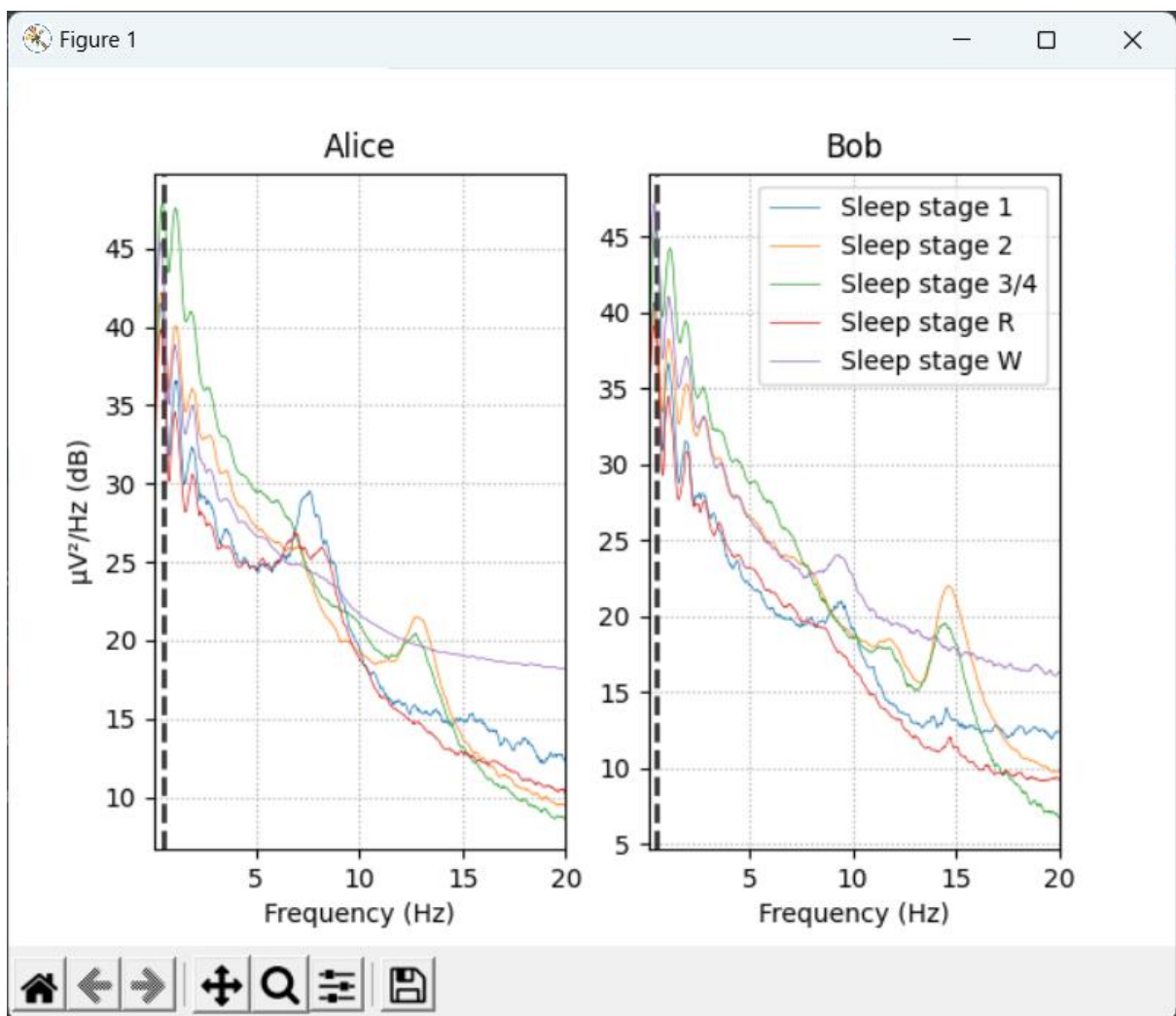


FIGURE 7.22 ALICE AND BOB SLEEPING STAGE



Accuracy score: 0.6699909338168631

[[155 0 0 2 0]

[ 96 5 4 1 3]

[ 78 12 443 21 8]

[ 0 0 5 100 0]

[ 84 27 23 0 36]]

	precision	recall	f1-score	support
Sleep stage W	0.38	0.99	0.54	157
Sleep stage 1	0.11	0.05	0.07	109
Sleep stage 2	0.93	0.79	0.85	562
Sleep stage 3/4	0.81	0.95	0.87	105
Sleep stage R	0.77	0.21	0.33	170

accuracy		0.67		1103
macro avg	0.60	0.60	0.53	1103
weighted avg	0.73	0.67	0.65	1103

## APPENDIX

### SOURCE CODE

```
from tkinter import *
import os
from tkinter import filedialog
from tkinter import messagebox
import pandas as pd
import tensorflow as tf
import numpy as np
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense, Activation, Conv1D, MaxPooling1D, Dropout, Flatten
from sklearn.utils import shuffle
from sklearn.preprocessing import normalize, StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

def endprogram():
    print("\nProgram terminated!")
    sys.exit()

def upload():
    # import cufflinks
    # cufflinks.go_offline()
    # cufflinks.set_config_file(world_readable=True, theme='pearl')
    import plotly
    import plotly.figure_factory as ff
    from plotly import tools
    from plotly.offline import init_notebook_mode, iplot
```

```

import imblearn
init_notebook_mode(connected=True)
pd.set_option('display.max_columns', 100)

data = pd.read_csv('Data/Sleepdataset.csv')

data.head()
data.tail()
data.shape
data['y'].value_counts()
data.y.hist();
plt.figure(figsize=(50, 4))
plt.subplot(131)
[plt.plot(data.values[i][1:-1]) for i in range(23)];
dic = {5: 0, 4: 0, 3: 0, 2: 0, 1: 1}
data['y'] = data['y'].map(dic)
print(data['y'].value_counts())
data.head()
data = data.drop('Unnamed', axis=1)
data = shuffle(data)
data.describe()
data.info()
print('Number of records of Non sleep {0} VS sleep {1}'.format(len(data[data['y']
== 0]),
len(data[data['y'] == 1])))

data[data['y'] == 0].describe().T
print(data[data['y'] == 0].describe().T)
data[data['y'] == 1].describe().T
print(data[data['y'] == 1].describe().T)

```

```

    print('Totall Mean VALUE for sleep : {}'.format((data[data['y'] ==
1].describe().mean()).mean()))

    print('Totall Std VALUE for sleep : {}'.format((data[data['y'] ==
1].describe().std()).std()))

    print('Totall Mean VALUE for NON sleep : {}'.format((data[data['y'] ==
0].describe().mean()).mean()))

    print('Totall Std VALUE for NON sleep : {}'.format((data[data['y'] ==
0].describe().std()).std()))

    [(plt.figure(figsize=(8, 4)), plt.title('NotSleep '), plt.plot(data[data['y'] ==
0].iloc[i][0:-1])) for i in
    range(5)];

```

```

    [(plt.figure(figsize=(8, 4)), plt.title('Sleep'), plt.plot(data[data['y'] == 1].iloc[i][0:-
1])) for i in range(5)];

```

```

# lists of arrays containing all data without y column

Not_Schizophrenia = [data[data['y'] == 0].iloc[:, range(0, len(data.columns) -
1)].values]

Schizophrenia = [data[data['y'] == 1].iloc[:, range(0, len(data.columns) - 1)].values]

```

# We will create and calculate 2d indicators in order plot data in 2 dimensions;

```
def indic(data):
```

```
    """Indicators can be different. In our case we use just min and max values
```

```
    Additionally, it can be mean and std or another combination of indicators"""
```

```
    max = np.max(data, axis=1)
```

```
    min = np.min(data, axis=1)
```

```
    return max, min
```

```
x1, y1 = indic(Not_Schizophrenia)
```

```
x2, y2 = indic(Schizophrenia)
```

```
fig = plt.figure(figsize=(14, 6))
```

```
ax1 = fig.add_subplot(111)
```

```
ax1.scatter(x1, y1, s=10, c='b', label='Not Sleep')
ax1.scatter(x2, y2, s=10, c='r', label='Sleep')
plt.legend(loc='lower left');
plt.show()
```

```
# Just Epileptic
```

```
x, y = indic(data[data['y'] == 1].iloc[:, range(0, len(data.columns) - 1)].values)
plt.figure(figsize=(14, 4))
plt.title('Sleep')
plt.scatter(x, y, c='r');
plt.show()
```

```
x, y = indic(data[data['y'] == 0].iloc[:, range(0, len(data.columns) - 1)].values)
plt.figure(figsize=(14, 4))
plt.title('NOT Sleep')
plt.scatter(x, y);
plt.show()
```

```
# define oversampling strategy
```

```
oversample
imblearn.over_sampling.RandomOverSampler(sampling_strategy='minority')
```

```
# fit and apply the transform
```

```
X, y = oversample.fit_resample(data.drop('y', axis=1), data['y'])
```

```
X.shape, y.shape
```

```
print('Number of records of Non Sleep {0} VS Sleep {1}'.format(len(y == True),
len(y == False)))
```

```
normalized_df = pd.DataFrame(normalize(X))
```

```
normalized_df
```

```
normalized_df['y'] = y
```

```

print('Normalized Totall Mean VALUE for Sleep: {}'.format(
    (normalized_df[normalized_df['y'] == 1].describe().mean()).mean()))
print(
    'Normalized Totall Std VALUE for Sleep: {}'.format(
        (normalized_df[normalized_df['y'] == 1].describe().std()).std()))

print('Normalized Totall Mean VALUE for NOT Sleep: {}'.format(
    (normalized_df[normalized_df['y'] == 0].describe().mean()).mean()))
print('Normalized Totall Std VALUE for NOT Sleep: {}'.format(
    (normalized_df[normalized_df['y'] == 0].describe().std()).std()))

def training():
    data = pd.read_csv('Data/Sleepdataset.csv')
    data.head()
    from sklearn.preprocessing import LabelBinarizer
    X = np.asarray(data.values)
    X = np.asarray(X[:, 1:-1])
    X = X.astype(float)
    minm = X.min()
    maxm = X.max()
    X_norm = (X - minm) / (maxm - minm)
    print(X.shape)
    from sklearn.model_selection import train_test_split
    mlb = LabelBinarizer()
    y = np.asarray(data['y'])
    # y=np.array.astype('float')
    # Y=mlb.fit_transform(y)
    Y = y.astype(float)
    # X=X.reshape(1200,178,1)

```

```

# print(X.shape)
Y = Y - 1
Y
# from tensorflow.python.keras.utils import to_categorical
# from tensorflow.python.keras.utils.np_utils import to_categorical
from tensorflow.python.keras.utils.np_utils import to_categorical
Y = to_categorical(Y)
Y
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20,
random_state=42)
model = Sequential()
# model.add(Conv1D(filters=2048, kernel_size=3, activation='relu', ))
# model.add(Dropout(0.5))
# model.add(MaxPooling1D(pool_size=3))
# model.add(Conv1D(filters=1024, kernel_size=3, activation='relu'))
# model.add(Dropout(0.5))
# model.add(Conv1D(filters=512, kernel_size=3, activation='relu',
input_shape=(178,1)))
# model.add(Dropout(0.5))
model.add(Conv1D(filters=256, kernel_size=3, activation='relu',
input_shape=(178, 1)))
model.add(Dropout(0.3))
model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
model.add(Dropout(0.3))
model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
model.add(Dropout(0.3))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(Dropout(0.3))
model.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
model.add(Dropout(0.3))

```

```

model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
# model.add(Dense(32, activation='relu'))
model.add(Dense(5, activation='softmax'))
# model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.summary()

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
y_ints = [y.argmax() for y in Y_train]
from sklearn.utils import class_weight
class_weights = class_weight.compute_class_weight('balanced',
                                                    np.unique(y_ints),
                                                    y_ints)

class_weight_dict = dict(enumerate(class_weights))
# class_weight_dict

from keras.callbacks import EarlyStopping
# from keras.callbacks import ModelCheckpoint
# earlyStopping = EarlyStopping(monitor='val_loss', patience=50, verbose=0,
mode='min')
# mcp_save = ModelCheckpoint('cnn.hdf5', save_best_only=True, verbose=1,
mode='max')

history2 = model.fit(X_train, Y_train, 100, 150, verbose=1) # ,
class_weight=class_weight_dict

model.save('cnn.hdf5')
acc = history2.history['accuracy']

```



```

loss = history2.history['loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'b', label=' accuracy')
plt.title('accuracy')
plt.legend()
plt.show()

```

```

# Train and validation loss
plt.plot(epochs, loss, 'b', label=' loss')
plt.title(' loss')
plt.legend()
plt.show()

```

```

#from sklearn.metrics import classification_report
#from sklearn.metrics import precision_recall_fscore_support
#scores = model.evaluate(X_test, Y_test, verbose=0, batch_size=200)
#pred = model.predict(X_test, verbose=0, batch_size=200)
#pred = np.round(pred)
#print(classification_report(Y_test, pred))
#f1 = precision_recall_fscore_support(Y_test, pred, average='micro')

#print("%s: %.2f%%" % (model.metrics_names[1], scores[1] * 100))

```

```

def eeeg():
    import numpy as np
    import matplotlib.pyplot as plt

    import mne
    from mne.datasets.sleep_physionet.age import fetch_data

```

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import FunctionTransformer
#import pandas as pd
#sleep_data = pd.read_csv('sleep.csv')
#sleep_data.info()
import numpy as np
import matplotlib.pyplot as plt
import mne

from mne.datasets.sleep_physionet.age import fetch_data
# from mne.time_frequency import
ALICE, BOB = 0, 1
[alice_files, bob_files] = fetch_data(subjects=[ALICE, BOB], recording=[1])
mapping = {'EOG horizontal': 'eog',
           'Resp oro-nasal': 'misc',
           'EMG submental': 'misc',
           'Temp rectal': 'misc',
           'Event marker': 'misc'}

raw_train = mne.io.read_raw_edf(alice_files[0])
annot_train = mne.read_annotations(alice_files[1])

raw_train.set_annotations(annot_train, emit_warning=False)
raw_train.set_channel_types(mapping)

```

```

# plot some data
raw_train.plot(duration=60, scalings='auto')
plt.show()

annotation_desc_2_event_id = {'Sleep stage W': 1,
                               'Sleep stage 1': 2,
                               'Sleep stage 2': 3,
                               'Sleep stage 3': 4,
                               'Sleep stage 4': 4,
                               'Sleep stage R': 5}

events_train, _ = mne.events_from_annotations(
    raw_train, event_id=annotation_desc_2_event_id, chunk_duration=30.)

event_id = {'Sleep stage W': 1,
            'Sleep stage 1': 2,
            'Sleep stage 2': 3,
            'Sleep stage 3/4': 4,
            'Sleep stage R': 5}

# plot events
mne.viz.plot_events(events_train, event_id=event_id,
                    sfreq=raw_train.info['sfreq'])

# keep the color-code for further plotting
stage_colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
plt.show()

tmax = 30. - 1. / raw_train.info['sfreq'] # tmax in included

```

```

epochs_train = mne.Epochs(raw=raw_train, events=events_train,
                           event_id=event_id, tmin=0., tmax=tmax, baseline=None)

print(epochs_train)

raw_test = mne.io.read_raw_edf(bob_files[0], stim_channel='Event marker',
                               infer_types=True)
annot_test = mne.read_annotations(bob_files[1])
annot_test.crop(annot_test[1]['onset'] - 30 * 60,
                annot_test[-2]['onset'] + 30 * 60)
raw_test.set_annotations(annot_test, emit_warning=False)
events_test, _ = mne.events_from_annotations(
    raw_test, event_id=annotation_desc_2_event_id, chunk_duration=30.)
epochs_test = mne.Epochs(raw=raw_test, events=events_test, event_id=event_id,
                          tmin=0., tmax=tmax, baseline=None)

print(epochs_test)

# visualize Alice vs. Bob PSD by sleep stage.
fig, (ax1, ax2) = plt.subplots(ncols=2)

# iterate over the subjects
stages = sorted(event_id.keys())
for ax, title, epochs in zip([ax1, ax2],
                             ['Alice', 'Bob'],
                             [epochs_train, epochs_test]):

    for stage, color in zip(stages, stage_colors):

```

```

spectrum = epochs[stage].compute_psd(fmin=0.1, fmax=20.)
spectrum.plot(ci=None, color=color, axes=ax,
              show=False, average=True, spatial_colors=False)
ax.set(title=title, xlabel='Frequency (Hz)')
ax1.set(ylabel='μV2/Hz (dB)')
ax2.legend(ax2.lines[2::3], stages)

plt.show()

```

```
def eeg_power_band(epochs):
```

```
    """EEG relative power band feature extraction.
```

This function takes an ``mne.Epochs`` object and creates EEG features based on relative power in specific frequency bands that are compatible with scikit-learn.

#### Parameters

-----

epochs : Epochs

The data.

#### Returns

-----

X : numpy array of shape [n\_samples, 5]

Transformed data.

```
    """
```

```
# specific frequency bands
```

```
FREQ_BANDS = {"delta": [0.5, 4.5],
```

```
              "theta": [4.5, 8.5],
```

```
"alpha": [8.5, 11.5],  
"sigma": [11.5, 15.5],  
"beta": [15.5, 30]}
```

```
spectrum = epochs.compute_psd(picks='eeg', fmin=0.5, fmax=30.)  
psds, freqs = spectrum.get_data(return_freqs=True)  
# Normalize the PSDs  
psds /= np.sum(psds, axis=-1, keepdims=True)
```

```
X = []  
for fmin, fmax in FREQ_BANDS.values():  
    psds_band = psds[:, :, (freqs >= fmin) & (freqs < fmax)].mean(axis=-1)  
    X.append(psds_band.reshape(len(psds), -1))
```

```
return np.concatenate(X, axis=1)
```

```
pipe = make_pipeline(FunctionTransformer(eeg_power_band, validate=False),  
                      RandomForestClassifier(n_estimators=100, random_state=42))
```

```
# Train
```

```
y_train = epochs_train.events[:, 2]  
pipe.fit(epochs_train, y_train)
```

```
# Test
```

```
y_pred = pipe.predict(epochs_test)
```

```
# Assess the results
```

```
y_test = epochs_test.events[:, 2]  
acc = accuracy_score(y_test, y_pred)
```

```

print("Accuracy score: {}".format(acc))

print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred, target_names=event_id.keys()))

def predict1():
    import tensorflow as tf
    import numpy as np
    from tkinter import messagebox

    model = tf.keras.models.load_model('Model/cnn.hdf5')

    data = np.array([-9,-65,-98,-102,-78,-48,-16,0,-21,-59,-90,-103,-84,-43,-9,3,-21,-
60,-96,-103,-75,-29,14,55,78,73,28,-13,-43,-68,-78,-75,-55,-41,-19,-20,-29,-36,-
20,1,16,14,-14,-42,-56,-45,-45,-45,-38,-47,-45,-37,-3,23,39,27,0,-28,-44,-37,-
22,5,30,31,6,-32,-27,-27,2,13,-6,-29,-41,-22,-13,-16,-31,-52,-60,-40,-
16,0,14,24,36,39,34,17,-7,-14,-1,16,27,28,18,-2,-
8,9,27,23,21,10,15,22,41,49,55,57,46,37,31,40,38,35,30,3,-34,-51,-42,-23,-
1,23,35,35,17,-1,-17,-8,26,55,54,38,19,4,-1,10,22,26,37,38,26,10,-4,-13,-
8,0,10,19,29,57,63,45,7,-13,-23,-9,9,11,3,-1,-2,4,18,27,27,14,15,11,10,4,2,-12,-32,-
41,-65,-83,-89,-73])

    pred = model.predict(data, verbose=0, batch_size=200)
    ind = np.argmax(pred)
    print(ind)
    out = ""
    med = ""
    if ind == 0:
        out='Normal'

```

```

elif ind == 1:
    out = 'Stage1'
    med ="Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
elif ind == 2:
    out = 'Stage2'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
elif ind == 3:
    out = 'Stage3'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
elif ind == 4:
    out = 'Stage4'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
print(out)

```

```

messagebox.showinfo("Prediction", "Prediction Result:"+ out)
messagebox.showinfo("Remedy", med)

```

```

def Adminlog():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Predict")
    login_screen.geometry("600x280")
    login_screen.title("Login Form")
    global username_verify
    global password_verify

```



```

username_verify = StringVar()
password_verify = StringVar()

global username_login_entry
global password_login_entry
label_0 = Label(login_screen, text="Predict form", width=20, font=("bold", 20))
label_0.place(x=90, y=53)
label_1 = Label(login_screen, text="Enter Value", width=20, font=("bold", 10))
label_1.place(x=80, y=130)
username_login_entry = Entry(login_screen, textvariable=username_verify)
username_login_entry.place(x=240, y=130)

bluebutton = Button(login_screen, text="Predict", fg="blue", font=('helvetica', 12),
command=preed)
bluebutton.place(x=220, y=210)

def preed():
    username1 = username_verify.get()
    print(username1)
    import tensorflow as tf
    import numpy as np
    from tkinter import messagebox

    model = tf.keras.models.load_model('Model/cnn.hdf5')

    float_array = np.fromstring(username1, dtype=float, sep=',')

```

```

reshaped_array = float_array.reshape(1, -1)

pred = model.predict(reshaped_array, verbose=0, batch_size=200)
ind = np.argmax(pred)
print(ind)
out = ""
med = ""
if ind == 0:
    out = 'Normal'
elif ind == 1:
    out = 'Stage1'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
elif ind == 2:
    out = 'Stage2'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
elif ind == 3:
    out = 'Stage3'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
elif ind == 4:
    out = 'Stage4'
    med = "Benzodiazepines are rapid eye movement (REM) sleep-suppressant
medications,"
print(out)

messagebox.showinfo("Prediction", "Prediction Result:" + out)
messagebox.showinfo("Remedy", med)

```

```

def main_account_screen():
    global main_screen
    main_screen = Tk()
    width = 600
    height = 600
    screen_width = main_screen.winfo_screenwidth()
    screen_height = main_screen.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    main_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))
    main_screen.resizable(0, 0)
    # main_screen.geometry("300x250")
    main_screen.title("Sleep Abnormal Prediction")

    Label(text="Sleep Abnormal Prediction", bg="Blue", width="300", height="5",
font=("Calibri", 16)).pack()

    Button(text="Upload Dataset", font=(
        'Verdana', 15), height="2", width="30", command=upload,
highlightcolor="black").pack(side=TOP)
    Label(text="").pack()

    Button(text="Training", font=(
        'Verdana', 15), height="2", width="30", command=training,
highlightcolor="black").pack(side=TOP)
    Label(text="").pack()

    Button(text="Predict", font=(
        'Verdana', 15), height="2", width="30", command=Adminlog,
highlightcolor="black").pack(side=TOP)

```

```
Label(text="").pack()
```

```
Button(text="EEG Based Evaluation", font=('Verdana', 15), height="2",  
width="30",command=eeeg).pack(side=TOP)
```

```
Label(text="").pack()
```

```
main_screen.mainloop()
```

```
main_account_screen()
```

## REFERENCES

- [1] Malik, Asra, et al. "Detection of Insomnia using Electrocardiography and Electromyography." 2021 International Conference on Artificial Intelligence (ICAI). IEEE, 2021.
- [2] Yang, Bufang, and Hongxing Liu. "Automatic identification of insomnia based on single-channel EEG labelled with sleep stage annotations." *IEEE Access* 8 (2020): 104281-104291.
- [3] Kuo, Chih-En, and Guan-Ting Chen. "A short-time insomnia detection system based on sleep EOG with RCMSE analysis." *IEEE Access* 8 (2020): 69763-69773.
- [4] Andresini, Giuseppina, et al. "Insomnia: Towards concept-drift robustness in network intrusion detection." *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*. 2021.
- [5] Sharma, Manish, Harsh S. Dhiman, and U. Rajendra Acharya. "Automatic identification of insomnia using optimal antisymmetric biorthogonal wavelet filter bank with ECG signals." *Computers in Biology and Medicine* 131 (2021): 104246.
- [6] Lee, Mi Hyun, et al. "Multitask fMRI and machine learning approach improve prediction of differential brain activity pattern in patients with insomnia disorder." *Scientific reports* 11.1 (2021): 1-13.
- [7] Afshani, Mortaza, et al. "Discriminating paradoxical and psychophysiological insomnia based on structural and functional brain images: a preliminary machine learning study." (2022).
- [8] Islam, Md Muhaiminul, et al. "Prediction of chronic insomnia using machine learning techniques." 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2020.

- [9] Ma, Xiaofen, et al. "Functional connectome fingerprint of sleep quality in insomnia patients: Individualized out-of-sample prediction using machine learning." *NeuroImage: Clinical* 28 (2020): 102439.
- [10] Wen, Ze-ying, et al. "Identification of discriminative neuroimaging markers for patients on hemodialysis with insomnia: a fractional amplitude of low frequency fluctuation-based machine learning analysis." *BMC Psychiatry* 23.1 (2023): 1-13.
- [11] Rim, Beanbonyka, et al. "Deep learning in physiological signal data: A survey." *Sensors* 20.4 (2020): 969.
- [12] Mostafa, Sheikh Shanawaz, et al. "A systematic review of detecting sleep apnea using deep learning." *Sensors* 19.22 (2019): 4934.
- [13] Yildirim, Ozal, Ulas Baran Baloglu, and U. Rajendra Acharya. "A deep learning model for automated sleep stages classification using PSG signals." *International journal of environmental research and public health* 16.4 (2019): 599.
- [14] Craik, Alexander, Yongtian He, and Jose L. Contreras-Vidal. "Deep learning for electroencephalogram (EEG) classification tasks: a review." *Journal of neural engineering* 16.3 (2019): 031001.
- [15] Hong, Shenda, et al. "Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review." *Computers in Biology and Medicine* 122 (2020): 103801.
- [16] Lashgari, Elnaz, Dehua Liang, and Uri Maoz. "Data augmentation for deep-learning-based electroencephalography." *Journal of Neuroscience Methods* 346 (2020): 108885.
- [17] Zhang, Xiang, et al. "A survey on deep learning-based non-invasive brain signals: recent advances and new frontiers." *Journal of neural engineering* 18.3 (2021): 031002.

- [18] Zhu, Tianqi, Wei Luo, and Feng Yu. "Convolution-and attention-based neural network for automated sleep stage classification." *International Journal of Environmental Research and Public Health* 17.11 (2020): 4152.
- [19] Shoeibi, Afshin, et al. "Epileptic seizures detection using deep learning techniques: a review." *International Journal of Environmental Research and Public Health* 18.11 (2021): 5780.
- [20] Mekruksavanich, Sakorn, and Anuchit Jitpattanakul. "Biometric user identification based on human activity recognition using wearable sensors: An experiment using deep learning models" *Electronics* 10.3 (2021): 308.
- [21] Zhang, Linda, et al. "Automated sleep stage scoring of the Sleep Heart Health Study using deep neural networks." *Sleep* 42.11 (2019): zsz159.
- [22] Hosseini, Mohammad-Parsa, Amin Hosseini, and Kiarash Ahi. "A review on machine learning for EEG signal processing in bioengineering." *IEEE reviews in biomedical engineering* 14 (2020): 204-218.
- [23] Singh, Himali, Rajesh Kumar Tripathy, and Ram Bilas Pachori. "Detection of sleep apnea from heart beat interval and ECG derived respiration signals using sliding mode singular spectrum analysis." *Digital Signal Processing* 104 (2020): 102796.
- [24] Kulkarni, Prathamesh M., et al. "A deep learning approach for real-time detection of sleep spindles." *Journal of neural engineering* 16.3 (2019): 036004.
- [25] Fukazawa, Yusuke, et al. "Smartphone-based mental state estimation: A survey from a machine learning perspective." *Journal of Information Processing* 28 (2020): 16-30.
- [26] da Silveira, T.L.T.; Kozakevicius, A.J.; Rodrigues, C.R. Single-channel EEG sleep stage classification based on a streamlined set of statistical features in wavelet domain. *Med. Biol. Eng. Comput.* 2017.

- [27] Memar, P.; Faradji, F. A Novel Multi-Class EEG-Based Sleep Stage Classification System. *IEEE Trans. Neural Syst. Rehabil. Eng.* 2018.
- [28] Yulita, I.N.; Fanany, M.I.; Arymurthy, A.M. Fast convolutional method for automatic sleep stage classification. *Healthc. Inform. Res.* 2018
- [29] Talo, M.; Baloglu, U.B.; Yildirim, Ö.; Acharya, U.R. Application of deep transfer learning for automated brain abnormality classification using MR images. *Cogn. Syst. Res.* 2019, 54, 176–188.
- [30] Faust, O.; Hagiwara, Y.; Hong, T.J.; Lih, S.; Acharya, R. Deep learning for healthcare applications based on physiological signals: A review *Computing. Methods Programs Biomed.* 2018.