Sleep Stage Detection Using Machine Learning Algorithm

Project Report

Course Name : Project & Thesis
Course Code : CSE 800

Submitted by

Roll || Reg
638 || 1797
656 || 1780
646 || 1793

Department of Computer Science and Engineering
FARIDPUR ENGINEERING COLLEGE, FARIDPUR

July 2023

COMPUTER SCIENCE & ENGINEERING DEPARTMENT
FARIDPUR ENGINEERING COLLEGE, FARIDPUR

Declaration

Report Title: **Sleep Stage Detection Using Machine Learning Algorithm**

We, the undersigned, hereby declare that the project report titled "Sleep Stage Detection Using Machine Learning Algorithm" is the original work carried out by us. We confirm that this project report represents our own work and that we have acknowledged all sources used in this report in accordance with the ethical guidelines and academic standards. It is solely prepared for the purpose of fulfilling the academic requirements of the course: “Project & Thesis, CSE 800”. We take full responsibility for the accuracy, authenticity, and originality of the content presented in this report. Any errors or omissions are unintentional and we are willing to rectify them if brought to our attention. We acknowledge that the ownership of the intellectual property and copyright of this project report remains with Faridpur Engineering College, Faridpur. However, we grant the institution the right to use the project report for academic and non-commercial purposes.

Place: Faridpur Engineering College, Faridpur
Date: 16/07/2023

…………………..
(638 || 1797)

…………………..
(656 || 1780)

…………………..
(646 || 1793)

Abstract

Classifying sleep states is one of the most important ways to figure out what's wrong with people who have problems sleeping. Since electroencephalography (EEG) can reliably detect neurological changes that take place during sleep, It is an effective method for studying the correlation between different stages of sleep and brain activity. Multiple brain and body systems are altered and play crucial roles throughout the NREM and REM phases of sleep, respectively. This study's overarching goal is to design a dependable system for automatically recognizing and classifying different stages of sleep, such as NREM and REM, without the need for human intervention. That will classify NREM and REM sleep phases that may be used to explain the model's findings and provide a description of REM and NREM sleep phases based on EEG sleep data. Classifying different types of sleep with the use of ensemble classification models like Random Forest, the XGBoost and The gradient boost was the focus of this study. Overall, we got accuracy for Random Forest **92.15%**, accuracy for Gradient Boosting **87.71%**, and accuracy for XGBoost **94.14%**.

Keywords: *Physiological Signals, Random Forests, Sleep Stage Detection, Electroencephalography (EEG) Healthcare, Electrooculography (EOG), Sleep Disorders, Electromyography (EMG), Support Vector Machines (SVM), Gradient Boosting, Machine Learning, Extreme Gradient Boosting(XGBoost).*

Table of Contents

Table of Figures

Table of Tables

CHAPTER 1

Introduction

Sleep stage detection is the process of analyzing and classifying the different stages of sleep based on bodily signs or data received during sleep. A typical night of sleep consists of alternating periods of REM sleep and non-REM sleep, among other phases.

One of the most basic biological activities, sleep is essential for relieving stress. The brain's basic operations are crucial to the capacities of learning, performance, and physical activity [1]. Research in the field of neuroscience has recently shifted its focus to the study of sleep quality and the diagnosis of sleep disorders.

Sleep stage scoring has become the primary method for analyzing sleep. [2]. In order to diagnose and treat sleep disorders, it is helpful to know which phases of sleep are the most significant.

Researchers use polysomnographic (PSG) data, the continuous monitoring of multiple electrophysiological signals, to assign ratings to different phases of sleep. The most generally used standard for sleep stage categorization is provided by the American Academy of Sleep Medicine (AASM). Non-Rapid Eye Movement (NREM) sleep, Rapid Eye Movement (REM) sleep, and Waking (W) are the three categories into which PSG recordings fall under this standard. The American Academy of Sleep Medicine (AASM) has issued updated guidelines in this area, which may be found at [3]. Each of the five stages of sleep has its own unique wave pattern, and the AASM guidelines outline those waves in detail.

The nervous system and the rest of the organism undergo a number of functional changes during NREM and REM sleep [4]. Alterations in hormone levels also occur throughout these two phases [4]. During deep non-rapid eye movement (NREM) sleep, the activity of the sympathetic nervous system drops. In rapid eye movement (REM) sleep, the breathing rate increases and becomes more erratic. Compared to while you're awake, your metabolic rate and blood flow are about the same during REM sleep, but they drop significantly during NREM sleep [5].Theta and delta waves predominated during awake, alpha and beta waves were reduced during NREM sleep, and REM sleep was characterized by a predominance of alpha and beta waves. There are a number of abnormalities of the neurological system and other bodily functions that may be detected via the careful monitoring of NREM and REM sleep.

1.1 **Motivation**

The many phases of sleep provide important information about the quality of sleep, which is crucial to one's health and well-being. Manual sleep stage assessment by human experts is not advised due to its drawbacks as a time-consuming, subjective, and variable process. Using machine learning methods to automate sleep stage identification makes the procedure more objective, consistent, and rapid. Continuous and non-intrusive monitoring of sleep patterns is made possible by automated sleep stage detection, which aids in the detection of anomalies, sleep disorders, and changes in sleep architecture over time. Individuals and medical professionals alike may benefit from a deeper understanding of sleep and the ability to better regulate their sleep as a result. By classifying sleep phases automatically, people may learn more about their own sleep habits and pinpoint the causes of poor sleep. Individuals may enhance their health and well-being by using this data to make educated decisions about their lifestyles, their sleeping conditions, and the quality of their sleep. Using machine learning methods to identify sleep phases contributes to expanding our understanding of how we sleep. Researchers may better understand sleep disorders, sleep physiology, and the influence of sleep on many aspects of health by automating the analytic process and handling enormous datasets and doing longitudinal studies.In conclusion, the need for objective, efficient, and individual sleep analysis is met by sleep stage identification utilizing machine learning algorithms. It may completely change the way we keep track of and learn about sleep, leading to more effective sleep management and higher standards of health for everyone.

1.2 **Objectives**

The primary objective is to develop machine learning algorithms that can automatically categorize sleep data into awake, REM sleep, and NREM sleep stages. The purpose is to eliminate subjectivity and save time by switching to an automated system for scoring.The algorithms' primary goal should be to reliably and accurately identify the sleep stages of its users. This precision is required for effective sleep analysis and the diagnosis of sleep problems. Because everyone sleeps differently and has their own unique traits, it's important to create models that can appropriately categorize sleep phases for a broad variety of people. The algorithms should enable individual sleep monitoring by detecting the various stages of sleep during which each person typically finds themselves. The goal is to help people become more self-aware of their sleep habits and routines, as well as their specific sleep requirements, so that they may make more educated choices about how to best meet those needs. Connectivity to Personal Wearables: The goal is to create algorithms that can be included into wearable devices like smartwatches or sleep trackers to provide easy and unobtrusive stage identification in one's slumber. The user experience may be improved and the scope of sleep stage identification can be broadened thanks to this connection, which enables smooth data gathering and real-time monitoring. The goal is to improve confidence and facilitate decision-making by allowing users, such as researchers, physicians, and people, to comprehend the aspects and attributes contributing to sleep stage categorization. The ultimate objective is to create accurate and reliable methods for classifying sleep phases in order to further sleep research and enhance clinical practice. Researchers, physicians, and patients might all benefit from these algorithms' potential to enhance sleep disorder diagnosis and treatment monitoring.

1.3 **Problem Definition**

Understanding sleep patterns, identifying sleep disorders, and developing effective therapies to enhance sleep quality all benefit greatly from the identification and study of distinct sleep phases. Manual grading of polysomnography (PSG) recordings by specialist sleep technologists is a time-consuming and subjective procedure that has been used for sleep stage identification in the past.

The goal of this study is to examine whether or not machine learning algorithms can accurately categorize sleep phases. In this article, we focus on three physiological signals—the electroencephalogram (EEG), the electrooculogram (EOG), and the electromyogram (EMG)—that might be utilized to construct a model that accurately classifies sleep stages. By employing machine learning techniques to automate the sleep stage classification process, we want to increase both the quality and efficiency of sleep research.

In this study, we go into the problem of identifying different stages of sleep, and we detail the challenges involved in doing so using just physiological data. We will talk about the disadvantages of using conventional approaches and the advantages of using machine learning algorithms for this job. We will also examine the current methods and research into sleep stage identification to see where the field stands and where it may be improved.

CHAPTER 2

Project Review

Sleep phases are typically categorized using EEG data collected continuously throughout the night. In order to zero in on the most important traits, these research suggest using a number of feature reduction methodologies. The stages of sleep have been categorized in a number of different ways. However, none of these methods limits sleep to NREM or REM states.

Citation	Title	Year Published	Stage Measure	Algorithm Used	Accuracy
[6]	Automatic sleep stage classification using decision tree multi class support vector machines	2015	Awake, Sleep.	SVM	88.01%
[7]	An Automatic Sleep Stage Classification Algorithm Using Improved Model Based Essence Feature	2020	Awa, REM, S1, S2, S3, S4	Random Forest	90.38%
[8]	Machine learning with ensemble stacking model for automated sleep staging using dual-channel EEG signal	2021	REM, N1, N2, N3, N4, Wake	CT-6, CT-5, CT-4, CT-3, CT-2.	85.85%, 84.98%, 85.51%, 85.37%, 87.40%

[9]	A Study of Human Sleep Stage Classification Based on Dual Channels of EEG Signal Using Machine Learning Techniques	2021	Wake, Sleep	SVM, DT, KNN, RF	90.83%, 87.35%, 86.53%, 91.67%
[10]	SleepExplain: Explainable Non-Rapid Eye Movement and Rapid Eye Movement Sleep Stage Classification from EEG Signal	2022	REM, NREM	RF, Gradient Boost.	92.54%, 94,25%

Table 2.1: Project Review

CHAPTER 3

Proposed Scheme

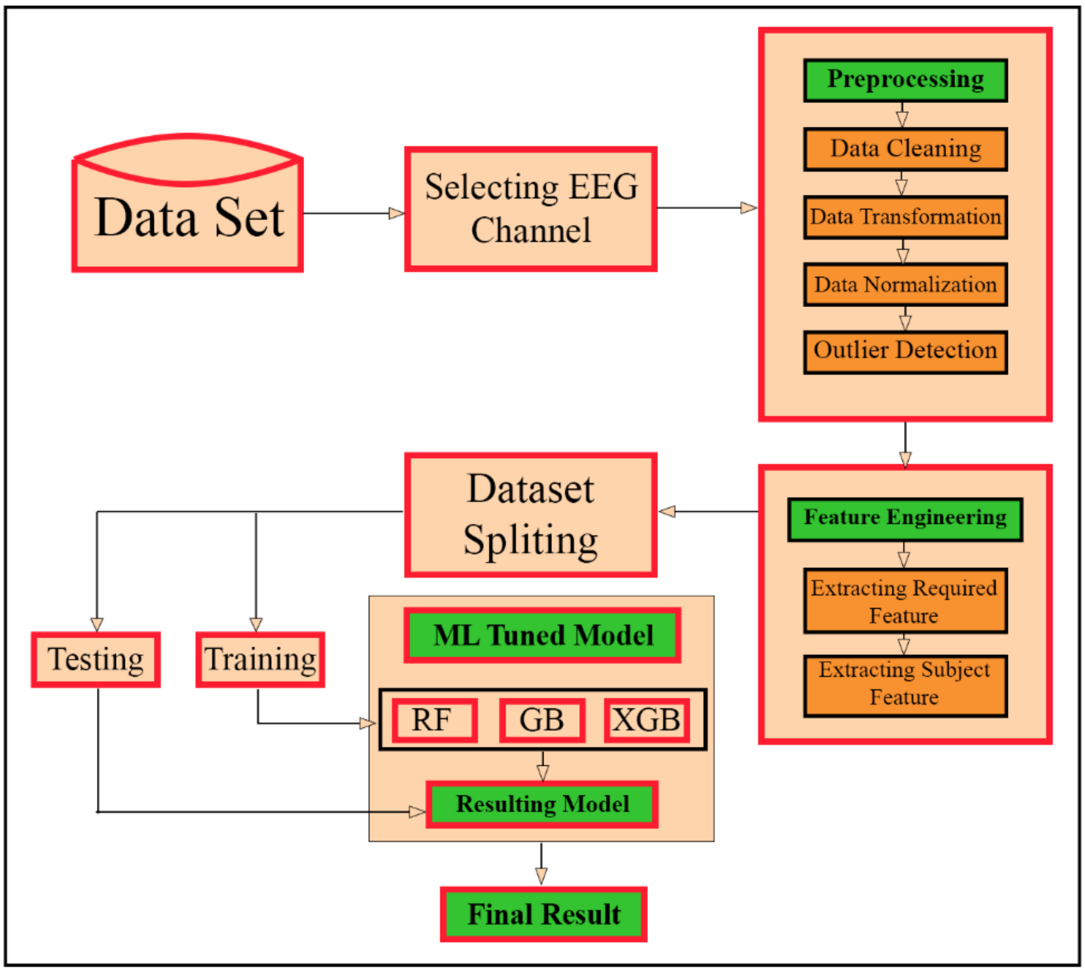


Fig 3.1: Workflow

Dataset: We used sleep recordings from the Haaglanden Medisch Centrum (HMC, The Hague, The Netherlands), which are available in PhysioNet [11] as an open-access public dataset. It was released in 2018 and just came out on March 18, 2022. The sample includes PSG recordings from 154 individuals, with a mean age of 53.8 15.4 years, who slept for a whole night. The group consists of 88 males and 66 females. Patient files were selected at random from a diverse pool of persons referred for PSG studies for a variety of sleep disorders. Brain electrical activity was sampled at 256 Hz and recorded using AgAgCl electrodes on SOMNO-screen PSG, PSG+, and EEG 10-20 recorders (SOMNOmedics, Randersacker, Germany). Two electrooculogram (EOG) channels (E1/M2 and E2/M2), one bipolar chin EMG channel, and one electrocardiogram (ECG) channel are included in each recording. W, N1, N2, N3, and R are recorded over the course of 30 seconds to form the sleep score. Sleep phases were manually assessed using the AASM criteria by well-trained sleep technologists. In this study, we use the usage of three EEG bands (F4, C4, and O2) based on the global 10-20 EEG technique.

3.1 **Project Challenges**

Data Preprocessing: To guarantee that the data used for training models is of good quality and faithfully represents the actual world, machine learning (ML) relies heavily on data cleaning and preprocessing. Some typical data pretreatment and cleaning methods in ML are as follows:

- Data cleaning
- Feature scaling
- Feature encoding
- Dimensionality reduction
- Handling missing data
- Outlier detection and removal
- Data normalization
- Handling class imbalance

We only used Feature Encoding Data Preprocessing Technique.

Missing Data: There are several potential causes of missing data, including mistakes in data collecting or incomplete information. Since many machine learning algorithms fail when presented with missing values, dealing with missing data is an essential step. In ML, missing data may be dealt with in a number of ways.

1. Deletion: To delete is to remove all rows or columns that have null values. However, this approach sometimes leads to data loss.
2. Imputation: Imputation is a method for making a statistical guess to fill in gaps in data. Mean imputation, median imputation, and regression imputation are three common types of imputation.
3. Prediction: The missing values are predicted using machine learning methods and other aspects of the dataset.

Outliers: Extreme numbers that stand out from the rest of the data in a collection are called outliers. Data collecting mistakes may cause outliers, but sometimes unusual data points simply indicate exceptional circumstances. Since outliers may distort findings, they can significantly affect how well machine learning systems function. Several methods exist in ML for dealing with outliers, such as:

1. Removal: The outlier must be eliminated from the data set. However, this approach sometimes leads to data loss.
2. Transformation: Logarithmic transformation and Box-Cox transformation are two examples of transformation methods that may be used to make data more normal and mitigate the effect of outliers.
3. Winsorization: Winsorization is a method wherein outlying numbers are swapped for more moderate ones. For instance, the 95th percentile may be used to replace all numbers above it.

Other Issues: Problems such as multicollinearity and overfitting are also possible in machine learning. When there is an unequal number of samples between categories, we say that there is an imbalance in the data. When three or more variables in a dataset are strongly linked, this is known as multicollinearity. When a model is very complex and closely fits the training data, it is said to be overfit. This leads to poor outcomes when the model is applied to new data. Several methods exist for dealing with such problems, such as:

1. Data augmentation: To achieve statistical parity, data augmentation may be used to artificially inflate the proportion of minority group samples.
2. Feature selection: Features are picked: Using this method, only those independent variables that have a low correlation with the result variable are retained.
3. Regularization: Overfitting may be avoided by regularization, which entails including a penalty term in the objective function of the model. L1 regularization (Lasso) and L2 regularization (Ridge) are two well-known techniques for achieving this goal.

We had no missing data, outliers and other issues in our dataset.

Exploratory Data Analysis: To better comprehend the data, spot anomalies and trends, and choose the best model for the issue at hand, exploratory data analysis (EDA) is a crucial stage in the Machine Learning (ML) process. The features of the data are analyzed and summarized with the help of EDA methods. Some frequent EDA approaches in ML are as follows:

- Statistical descriptions
- Data Visualization
- Correlation analysis
- Missing value imputation
- Outlier detection
- Dimensionality Reduction

When it comes to laying the groundwork for a machine learning model, EDA methods are crucial. These methods aid in the discovery of meaningful connections within the data, which in turn facilitates the development of a more trustworthy model. Describe any associations or trends that were spotted in the data.

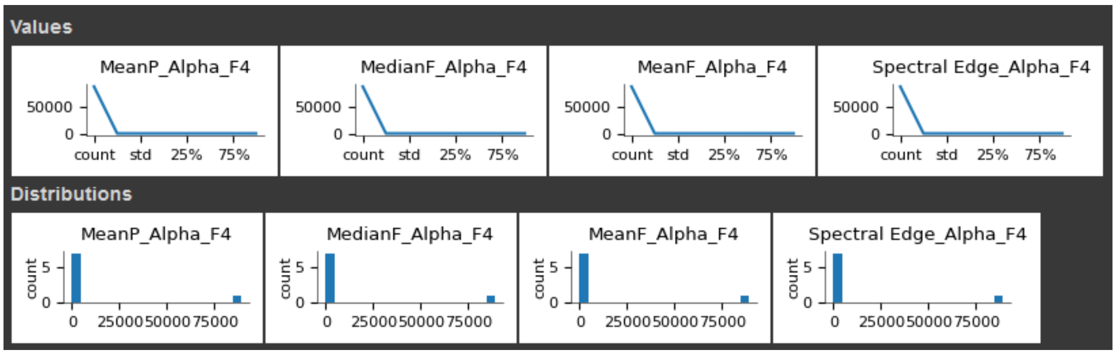


Fig 3.2: Statistical Description

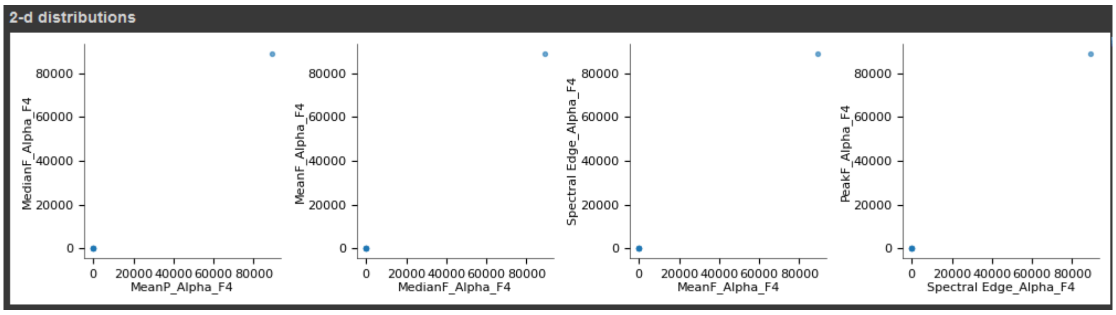


Fig 3.3: 2D Distribution

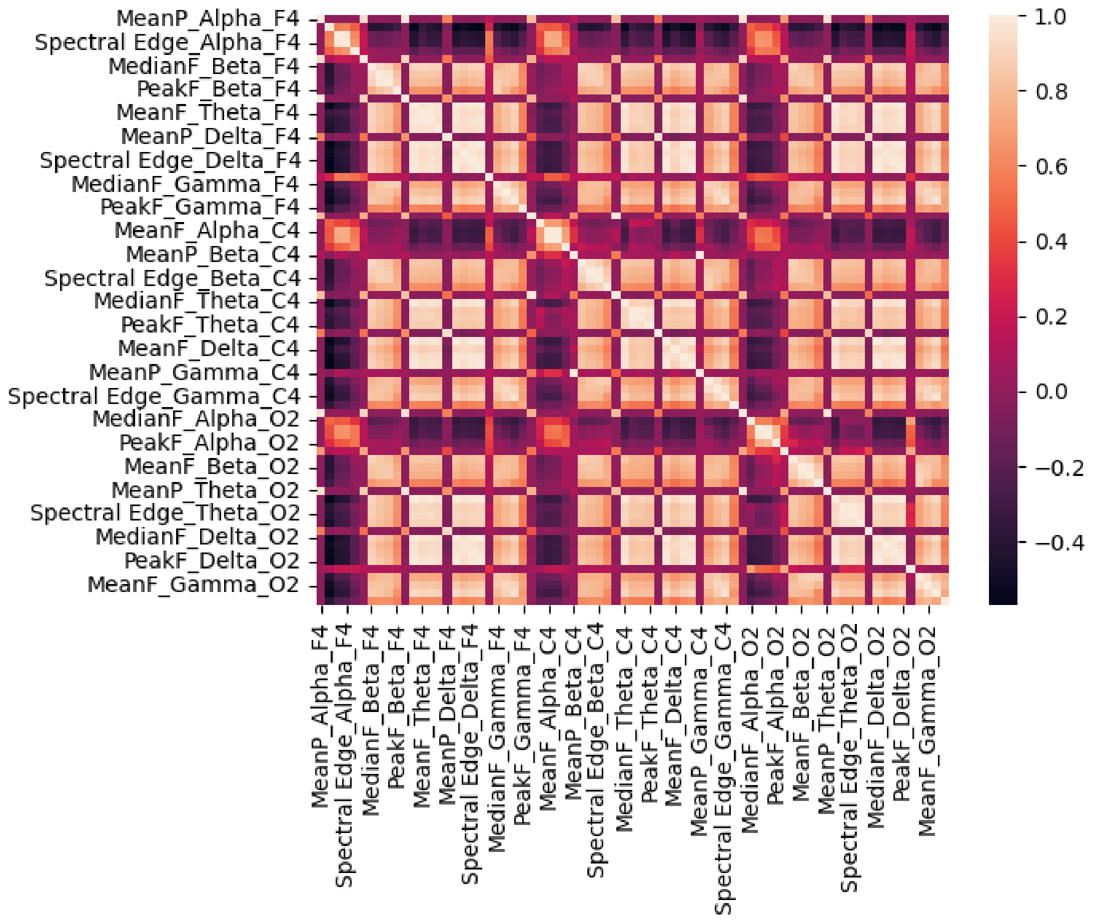


Fig 3.4: Correlation Plot (Heatmap)

Feature Extraction: The electroencephalogram (EEG) may be classified according to the frequencies and strengths of its signals at various frequencies. The frequency range of the different types of brain waves is as follows: 0.5 Hz to 4 Hz for delta waves, 4 Hz to 8 Hz for theta waves, 13 Hz to 30 Hz for alpha waves, 13 Hz to 30 Hz for beta waves, and 30 Hz to 44 Hz for gamma waves. Fast Fourier transforms (FFT) and other approaches were used to extract characteristics from EEG signals so that their potential could be investigated. Power spectral density (PSD) was determined for each time interval using Welch periodograms. Average power, median frequency, mean frequency, spectral edge frequency, and peak frequency at any given time period were all calculated using this PSD. The epoch width was set at 30 seconds. The collected EEG features for this study are summarized in Table 1. The collection includes 75 EEG feature sets that together capture different types of brain activity.

EEG Channel	EEG Spectral Waves	EEG Feature	Number of Feature
F4, C4 and O2	δ,θ,α,β and γ	Median Frequency	15
F4, C4 and O2	δ,θ,α,β and γ	Mean Power	15
F4, C4 and O2	δ,θ,α,β and γ	Spectral Edge	15
F4, C4 and O2	δ,θ,α,β and γ	Mean Frequency	15
F4, C4 and O2	δ,θ,α,β and γ	Peak Frequency	15

Table 3.1. Feature detection based on EEG data.

Feature engineering involves extracting, transforming, and synthesizing novel features from raw data. Feature engineering may improve a model's performance in a few key ways:

- Better Feature Representation.
- Improved Model Interpretability.
- Reduced Overfitting.
- Improved Robustness.

Model Selection: By pooling the results of several individual models, ensemble approaches in machine learning increase prediction accuracy and resilience. The premise behind ensemble approaches is that by combining the results of numerous models, the shortcomings and biases of each model may be reduced, leading to better overall performance.

Due to their potential to increase prediction accuracy, generalization, and resilience, ensemble approaches have gained popularity and achieved exceptional success in a wide range of machine learning tasks and competitions. They have become an integral part of the machine learning process and see extensive practical use.

The two most common forms of ensemble methods are the average approach and the boosting method.

Averaging Methods:

- Bagging: Bagging, an abbreviation for "bootstrap aggregating," is a method in which several models are trained separately on various random subsets of the training data. The predictions of these models are then aggregated by averaging (for regression tasks) or voting (for classification tasks).
- Random Forest: One common ensemble strategy that expands on bagging is called Random Forest. Multiple decision trees are combined, with each tree trained on a different collection of features. All of the trees' forecasts are averaged or voted on to arrive at a single conclusion.

Boosting Methods:

- Gradient Boosting: Gradient boosting is an effective boosting approach that incrementally constructs a set of low-quality learners. Each level involves training a new model to fix the flaws in the earlier stages' work. The models are trained by gradient descent with the objective of minimizing a loss function.
- XGBoost: The gradient boosting algorithm XGBoost (Extreme Gradient Boosting) is widely used. They enhance the performance of the gradient boosting algorithm by including new regularization methods, parallel processing, and tree-building approaches.

The prediction accuracy, overfitting, and generalization may all be enhanced by using an ensemble technique. There are, however, costs associated with them, such as higher computational complexity, larger memory needs, and less interpretability as a result of combining numerous models.

As a strong tool in machine learning, ensemble methods pool the knowledge of many models to outperform those models working alone. They have been shown to be useful in many settings, and more exploration into their potential is ongoing in the field of machine learning.

Model Training: In machine learning, "training" a model is running it against a dataset to identify patterns and correlations that will help it correctly predict or classify incoming data.

Several hyperparameters allow for fine-tuning of the machine learning method Random Forest, Gradient Boost & XGBoost. Important hyperparameters are the maximum depth of each tree (max_depth) and the number of trees (n_estimators).

The forest's tree count is determined by the n_estimators hyperparameter. There is a tradeoff between improved performance and longer training times and computational complexity when increasing the number of trees used in an algorithm. How many estimators (n_estimators) you should use to get the best results is data- and resource-specific. In practice, n_estimators are often varied within a range, and the best performing value is chosen through validation or cross-validation.

In a decision forest, the maximum depth of each tree is set by the max_depth hyperparameter. In general, the performance of the individual trees improves with increasing the maximum depth, but this comes with a higher danger of overfitting. If you want to strike a good balance between model complexity and generalization performance, the best value for max_depth will change depending on the size and complexity of the dataset. In practice, it is customary to experiment with several max_depth settings before settling on the one that yields the best results in validation or cross-validation.

Training machine learning (ML) models is not without its difficulties, some of which include:

1. Quantity and quality of the data: Obtaining high-quality and adequate data for training a model is a significant difficulty in ML. The information must be fair and accurate in its portrayal of the demographic of interest.
2. Engineering of features: Selecting and modifying the most useful characteristics from the given data is known as feature engineering. This is an essential part of the ML process since it determines how well the model will function in practice.
3. Excessive or insufficient snugness: When a model becomes too complicated and learns to fit the training data too well, it is said to have overfit, and it will perform poorly when applied to fresh data.
4. Tuning hyperparameters is the process of adjusting a model's initial settings and parameters before it is trained.
5. Lack of computing resources, such as a strong central processing unit or graphics processing unit, is a typical issue. This may slow down the training process and restrict the models' potential complexity and scale.

6. To get around this problem, you may leverage cloud-based services like Google Collab that provide you access to powerful computers. Even if these tools are accessible, there may not be enough memory or computing power to fully complete the training.

Model Evaluation: Metrics are used to evaluate a machine learning model's accuracy and how effectively it can learn from data it has never seen before. Commonly used assessment measures for sleep stage identification include:

- Accuracy: The proportion of samples in the dataset whose labels are accurate.
- Precision: The percentage of correct model predictions relative to all correct forecasts.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall: The percentage of positive samples that are really representative of the dataset.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- F1 score: Harmonic mean of accuracy and recall with weights.

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

In addition to these measures, the confusion matrix is often used to see how well a model does on various phases of sleep. For each sleep stage, the confusion matrix details how many samples were properly identified and how many were misclassified.

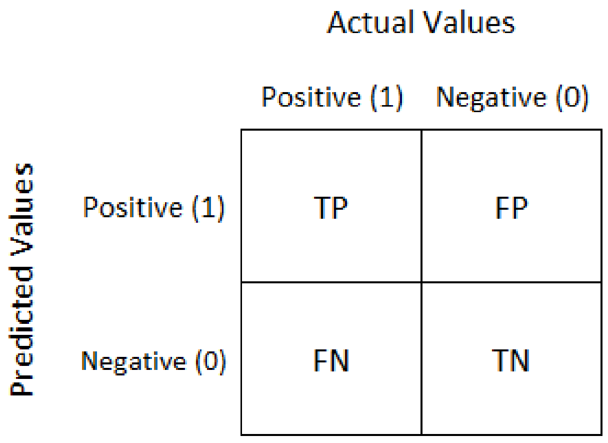


Fig 3.5: Confusion Matrix

CHAPTER 4

Experimental Result

4.1 **Predicted Result**

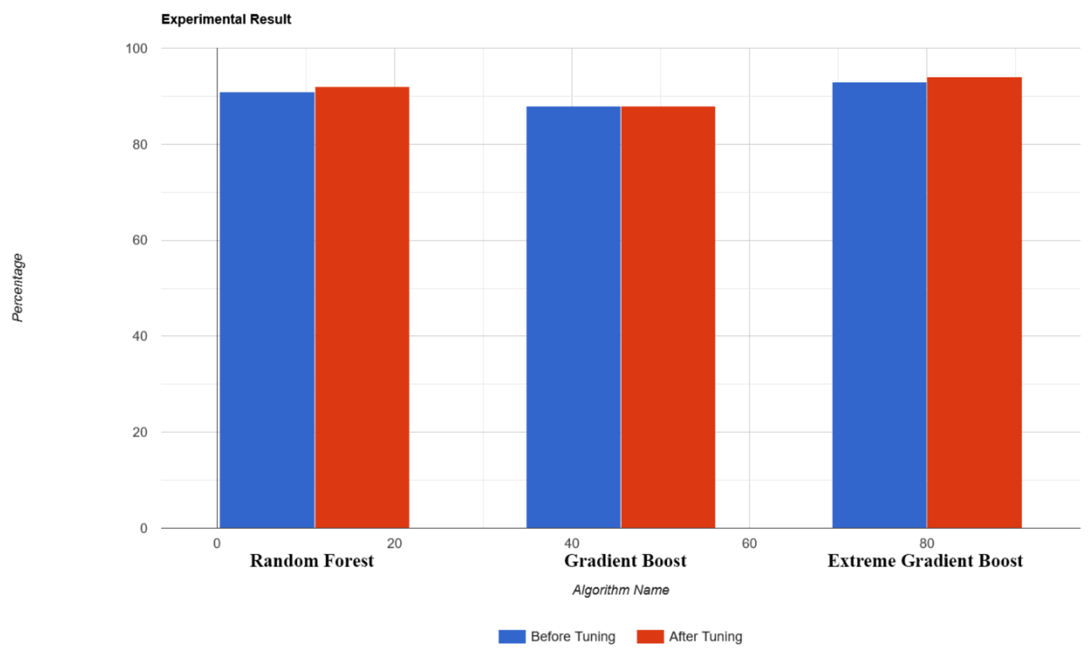


Fig 4.1: Experimental Result

Hyperparameters	RF	GB	XGB
n_estimator	220	2500	2400
max_depth	100	100	100
learning_rate	0.1	0.1	0.1

Table 4.1: Hyperparameters for tuning ML models.

Evaluation Metric	**Random Forest**	
	NREM	REM
Precision	96%	76%
Recall	94%	83%
f1-Score	95%	79%

Table 4.2: Performance Matrix(RF)

Evaluation Metric	**Random Forest**
Accuracy	91%

Table 4.3: Final Performance Matrix(RF)

Evaluation Metric	**Gradient Boosting**	
	NREM	REM
Precision	94%	65%
Recall	91%	73%
f1-Score	92%	69%

Table 4.4 : Performance Matrix(GB)

Evaluation Metric	**Gradient Boosting**
Accuracy	88%

Table 4.5: Final Performance Matrix(GB)

Evaluation Metric	**Extreme Gradient Boosting**	
	NREM	REM
Precision	95%	80%
Recall	96%	79%
f1-Score	96%	80%

Table 4.6: Performance Matrix(XGB)

Evaluation Metric	**Extreme Gradient Boosting**
Accuracy	93%

Table 4.7: Final Performance Matrix(XGB)

4.2 **Result After Tuning**

Evaluation Metric	**Random Forest**	
	NREM	REM
Precision	96%	76%
Recall	94%	84%
f1-Score	95%	80%

Table 4.8: Performance Matrix(RF)

Evaluation Metric	**Random Forest**
Accuracy	92.15%

Table 4.9: Final Performance Matrix(RF)

Evaluation Metric	**Gradient Boosting**	
	NREM	REM
Precision	93%	64%
Recall	91%	71%
f1-Score	92%	68%

Table 4.10: Performance Matrix(GB)

Evaluation Metric	**Gradient Boosting**
Accuracy	87.71%

Table 4.11: Final Performance Matrix(GB)

Evaluation Metric	**Extreme Gradient Boosting**	
	NREM	REM
Precision	96%	84%
Recall	97%	83%
f1-Score	96%	84%

Table 4.12: Performance Matrix(XGB)

Evaluation Metric	**Extreme Gradient Boosting**
Accuracy	94.14%

Table 4.13: Final Performance Matrix(XGB)

Model	Precision	Recall	F1-Score	Accuracy
Random Forest	86%	89%	87.5%	92.15%
Gradient Boosting	78.5%	81%	80%	87.71%
Extreme Gradient Boosting	90%	90%	90%	94.14%

Table 4.14 : Various ML models Precision Recall F1-Score & Accuracy

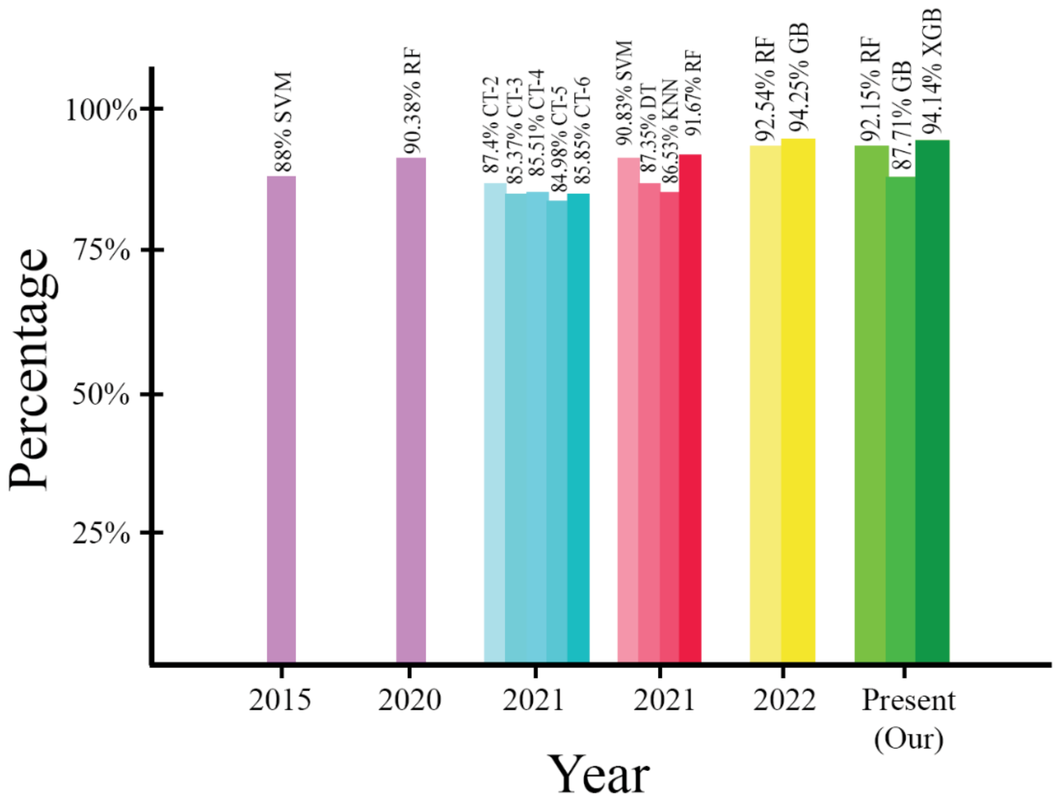
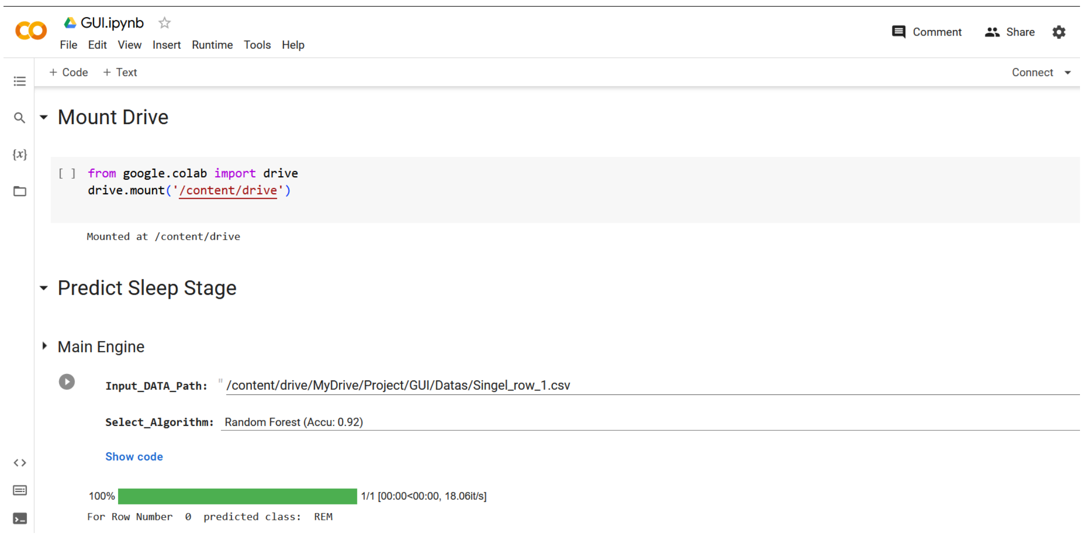


Fig 4.2: Comparison of Accuracy over years.

4.3 Graphical User Interface



Mount Drive

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Predict Sleep Stage

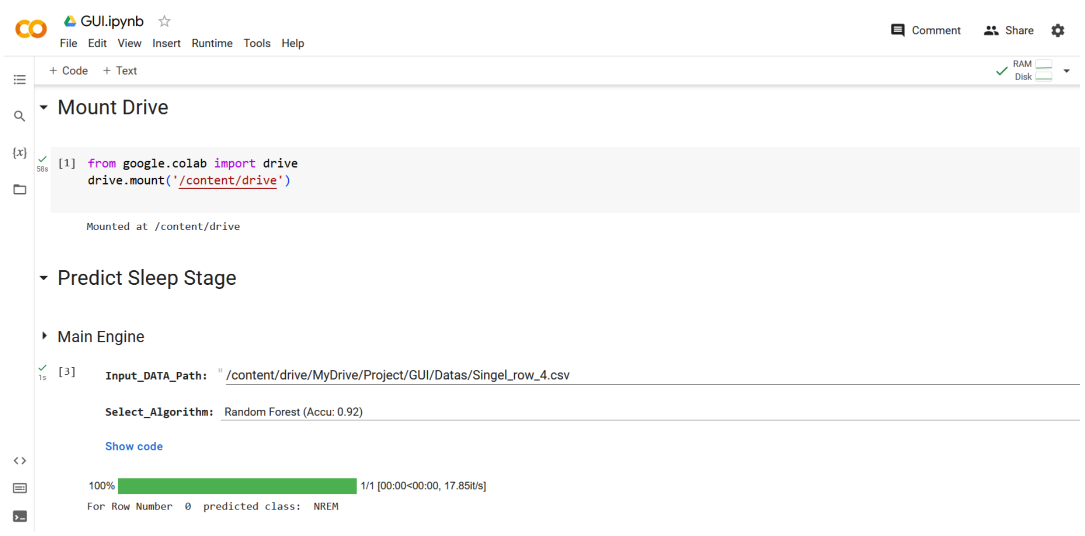
Main Engine

Input_DATA_Path: /content/drive/MyDrive/Project/GUI/Datas/Singel_row_1.csv

Select_Algorithm: Random Forest (Accu: 0.92)

Show code

```
100% 1/1 [00:00<00:00, 18.06it/s]
For Row Number  0  predicted class:  REM
```



Mount Drive

```
[1] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Predict Sleep Stage

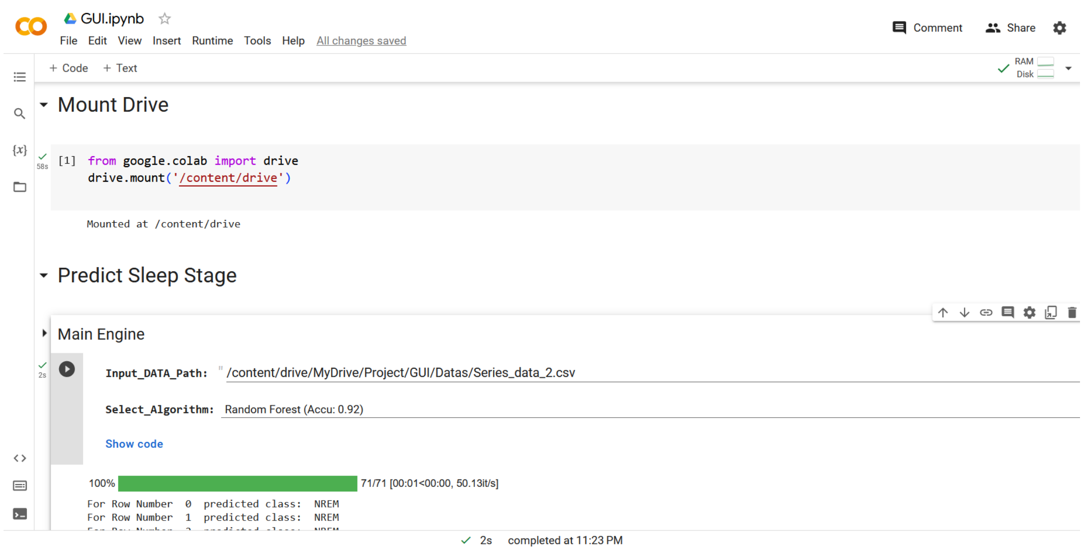
Main Engine

[3] Input_DATA_Path: /content/drive/MyDrive/Project/GUI/Datas/Singel_row_4.csv

Select_Algorithm: Random Forest (Accu: 0.92)

Show code

```
100% 1/1 [00:00<00:00, 17.85it/s]
For Row Number  0  predicted class:  NREM
```



Mount Drive

```
[1] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

Predict Sleep Stage

Main Engine

Input_DATA_Path: " /content/drive/MyDrive/Project/GUI/Datas/Series_data_2.csv

Select_Algorithm: Random Forest (Accu: 0.92)

Show code

```
100% 71/71 [00:01<00:00, 50.13it/s]
For Row Number  0  predicted class:  NREM
For Row Number  1  predicted class:  NREM
```

✓ 2s completed at 11:23 PM

CHAPTER 5

Conclusion & Future Work

5.1 Conclusion

In this study, we created a machine learning model to analyze EEG data and identify different stages of sleep. To train a random forest classifier, we first retrieved numerous characteristics from the epochs.

The performance of the model was evaluated using a variety of metrics, including accuracy, precision, recall, and the F1 score. The model accurately recognizes and classifies distinct stages of sleep using just EEG data, as shown by its high accuracy and F1 score on the test data.

The following goals were met by the project:

1. A machine learning model for identifying sleep stages from EEG data has been built and validated.
2. Used a number of criteria to assess the model's efficacy; test results showed good accuracy and an F1 score.
3. Created a Graphical User Interface that easily can detect the sleep stage perfectly with the trained model.

5.2 **Future Work**

1. Gathering additional data from a wider range of people will improve the model's ability to generalize.
2. To enhance the model's precision and performance, we will investigate alternative feature extraction methods and machine learning algorithms.
3. Creating a tool that can be used by both medical professionals and patients to monitor and analyze sleep patterns in real time.

Reference:

[1] Itakura Distance: A Useful Similarity Measure between EEG and EOG Signals in Computer-aided Classification of Sleep Stages (2005). Conference. 2. 1189-92. 10.1109/IEMBS.2005.1616636. DOI:10.1109/IEMBS.2005. 1616636

[2] Assessment of Itakura Distance as a Valuable Feature for Computer-aided Classification of Sleep Stages. IEEE Engineering in Medicine and Biology Society. Conference. 2007. 3300-3. 10.1109/IEMBS.2007.4353035. DOI:10.1109/ IEMBS. 2007. 4353035

[3] Notice of Removal: An improved K-means clustering algorithm for sleep stages classification. 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Hangzhou, China, 2015, pp. 1222-1227, doi: 10.1109/SICE.2015.7285326.

[4] Sleep Disorders and Sleep Deprivation: An Unmet Public Health Problem. Colten HR, Altevogt BM, editors. Washington (DC): National Academies Press (US); 2006. PMID: 20669438. doi.org/10.17226/11617

[5] Cerebral O2 metabolism and cerebral blood flow in humans during deep and rapid-eye-movement sleep. J Appl Physiol (1985). 1991 Jun;70(6):2597-601. doi: 10.1152/jappl.1991.70.6.2597. PMID: 1885454.

[6] Learning machines and sleeping brains: Automatic sleep stage classification using decision-tree multi-class support vector machines. J Neurosci Methods. 2015 Jul 30;250:94-105. doi: 10.1016/j.jneumeth.2015.01.022. Epub 2015 Jan 25. PMID: 25629798.

[7] An Automatic Sleep Stage Classification Algorithm Using Improved Model Based Essence Features. Sensors 2020, 20, 4677. https://doi.org/10.3390/ s20174677

[8] Machine learning with ensemble stacking model for automated sleep staging using dual-channel EEG signal. Volume 69, 2021, 102898, ISSN 1746-8094, https://doi.org/10.1016/j.bspc.2021.102898.

[9] A Study of Human Sleep Stage Classification Based on Dual Channels of EEG Signal Using Machine Learning Techniques. SN Computer Science. 2. 10.1007/s42979-021-00528-5.

[10] SleepExplain: Explainable Non-Rapid Eye Movement and Rapid Eye Movement Sleep Stage Classification from EEG Signal, 2022 25th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, 2022, pp. 248-253, doi: 10.1109/ICCIT57492 .2022.10055956.

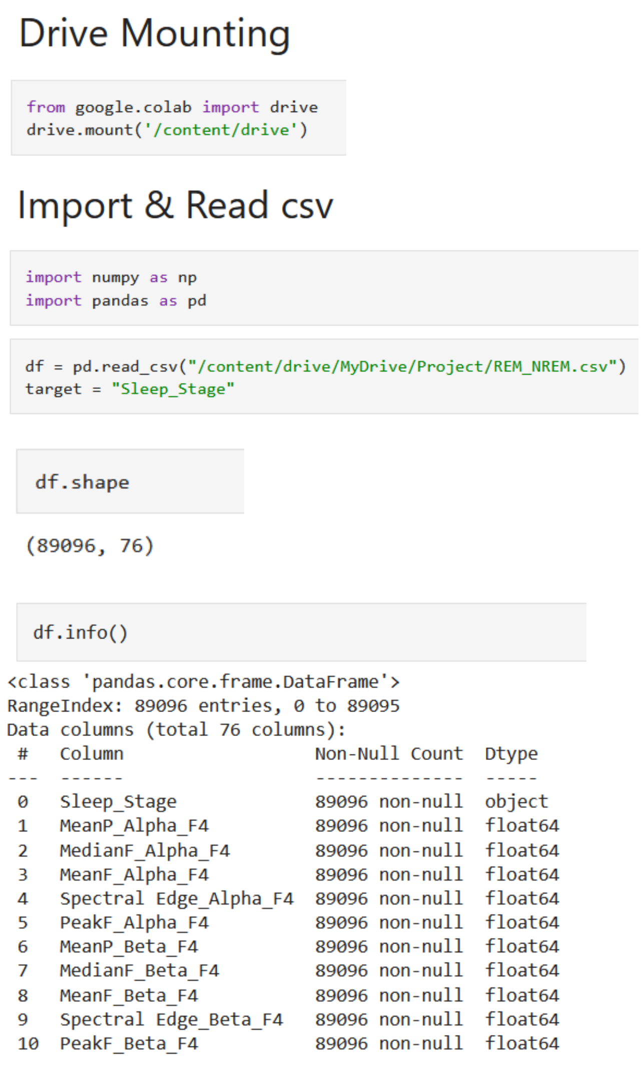
[11] Haaglanden Medisch Centrum sleep staging database (version 1.1). Alvarez-Estevez, D., & Rijsman, R. (2022). PhysioNet. https://doi.org/ 10.13026/t79q-fr32.

Appendix

https://github.com/RKNahid/Automated-Sleep-Stage-Detection-using-Machine-Learning-Algorithm/tree/main

Files:

- ProjectMain.ipynb
- RFTuning.ipynb
- GBTuning.ipynb
- XGBTuning.ipynb
- GUI.ipynb



Drive Mounting

```
from google.colab import drive
drive.mount('/content/drive')
```

Import & Read csv

```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv("/content/drive/MyDrive/Project/REM_NREM.csv")
target = "Sleep_Stage"
```

```
df.shape
```

```
(89096, 76)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89096 entries, 0 to 89095
Data columns (total 76 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Sleep_Stage            89096 non-null  object
 1   MeanP_Alpha_F4         89096 non-null  float64
 2   MedianF_Alpha_F4       89096 non-null  float64
 3   MeanF_Alpha_F4         89096 non-null  float64
 4   Spectral Edge_Alpha_F4 89096 non-null  float64
 5   PeakF_Alpha_F4         89096 non-null  float64
 6   MeanP_Beta_F4          89096 non-null  float64
 7   MedianF_Beta_F4        89096 non-null  float64
 8   MeanF_Beta_F4          89096 non-null  float64
 9   Spectral Edge_Beta_F4  89096 non-null  float64
 10  PeakF_Beta_F4          89096 non-null  float64
```

Encoding (Label)

```
sorted(list(set(df[target])))
```

```
['NREM', 'REM']
```

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

df[target] = encoder.fit_transform(df[target])
```

```
sorted(list(set(df[target])))
```

```
[0, 1]
```

Dataset Spliting

```
x = df.loc[:,df.columns != target]

y = df[target]
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2)
```

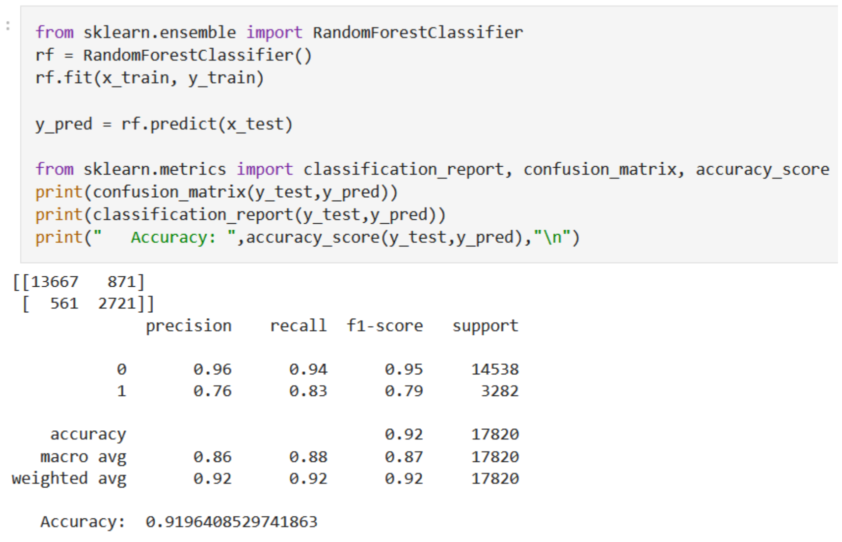
Balancing Dataset (Oversampling)

```
!pip install imbalanced-learn

from imblearn.over_sampling import SMOTE

def balance(x_temp,y_temp):
  smote = SMOTE()

  x_temp, y_temp = smote.fit_resample(x_temp, y_temp)

  return x_temp, y_temp
```

```
x_train, y_train = balance(x_train, y_train)
```

Training Phase

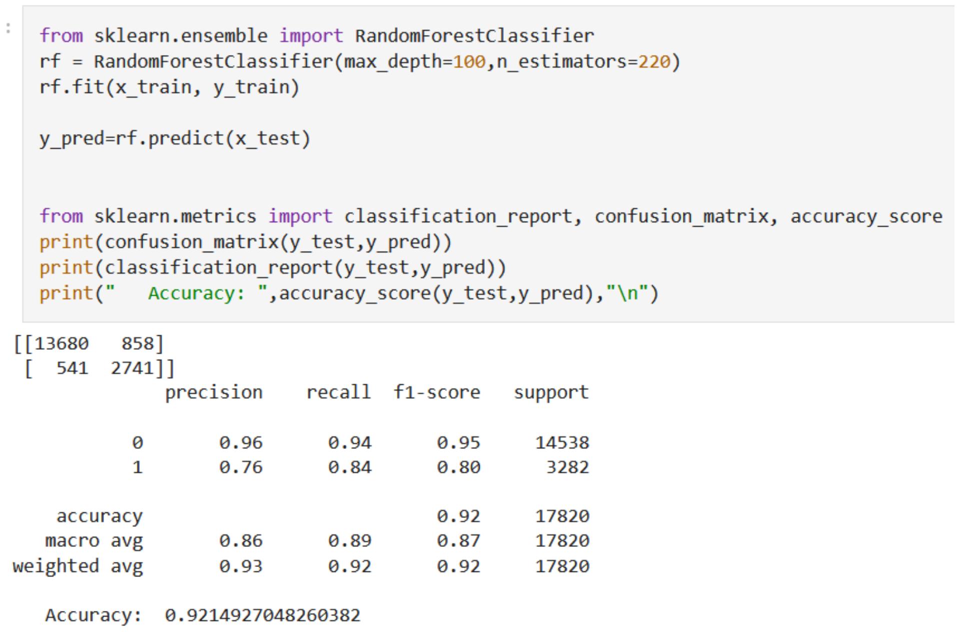
Random Forest



```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train, y_train)

y_pred = rf.predict(x_test)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print("   Accuracy: ",accuracy_score(y_test,y_pred),"\n")
```

```
[[13667   871]
 [  561  2721]]
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     14538
           1       0.76      0.83      0.79      3282

    accuracy                           0.92     17820
   macro avg       0.86      0.88      0.87     17820
weighted avg       0.92      0.92      0.92     17820

   Accuracy:  0.9196408529741863
```

After Tunning



```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=100,n_estimators=220)
rf.fit(x_train, y_train)

y_pred=rf.predict(x_test)


from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print("   Accuracy: ",accuracy_score(y_test,y_pred),"\n")
```

```
[[13680   858]
 [  541  2741]]
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     14538
           1       0.76      0.84      0.80      3282

    accuracy                           0.92     17820
   macro avg       0.86      0.89      0.87     17820
weighted avg       0.93      0.92      0.92     17820

   Accuracy:  0.9214927048260382
```