

KONGU ENGINEERING COLLEGE

PERUNDURAI

**HUMAN STRESS DETECTION IN AND THROUGH SLEEP USING MACHINE
LEARNING**

BHUVANESHWARI.C (21ADR009)

HARINI.M (21ADR015)

SNEHAVARSHINI.S (21ADR048)

SOUNDHARYA.M (21ADR050)

harinim.21aid@kongu.edu

bhuvaneshwaric.21aid@kongu.edu

snehavarshinis.21aid@kongu.edu

soundharyam.21aid@kongu.edu

TEAM MENTOR

MS N.ABINAYA

CHAPTER 1

INTRODUCTION

Stress is a common and significant issue in today's fast-paced world, leading to various physical and mental health problems. Detecting and understanding stress patterns in individuals can play a crucial role in promoting well-being and providing timely interventions. Sleep is an essential aspect of human life and is closely linked to stress levels. Analyzing sleep data using machine learning algorithms can offer valuable insights into stress detection.

The use of machine learning algorithms, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RDF), Logistic Regression (LR), and Naive Bayes (NB), to detect and predict stress levels based on sleep data. By leveraging these algorithms, create a predictive models that can identify stress patterns through sleep analysis.

Sleep data can be collected through various means, such as wearable devices, mobile applications, or polysomnography (PSG) tests. Sleep-related features, such as sleep duration, sleep stages, sleep efficiency, and sleep disruptions, can serve as potential inputs for the machine learning algorithms. Additionally, other relevant information, including heart rate, respiratory rate, and body movement, can be incorporated into the analysis.

The algorithms have been widely used in various machine learning tasks and have shown promising results. KNN is a simple yet effective algorithm that classifies data points based on their proximity to other instances. SVM aims to find an optimal hyperplane that separates different classes. RDF constructs an ensemble of decision trees to make predictions. LR models the relationship between the dependent variable and independent variables using a logistic function. NB is a probabilistic classifier based on Bayes' theorem, assuming independence between features.

By comparing the performance of these algorithms on the sleep data, determine which algorithm(s) are most suitable for stress detection. The evaluation metrics, such as accuracy, precision, recall, and F1 score, will be used to assess the effectiveness of the models. The impact of different feature combinations and preprocessing techniques on the performance of the algorithms.

The results can contribute to the development of automated systems that detect stress levels based on sleep data. Such systems can provide personalized recommendations, interventions, or alerts to help individuals manage stress and improve their overall well-being.

1.1 Background

Prior research has established a link between stress and sleep disruption. However, manually analyzing sleep data to determine stress levels is timeconsuming and subject to human error. Machine learning techniques offer an automated and efficient approach to extract meaningful patterns from sleep data and predict stress levels accurately. KNN, SVM, RDF, LR, and NVB are wellestablished machine learning algorithms commonly used for classification tasks. By training these algorithms on labeled sleep data, it is possible to create models capable of differentiating between low-stress and high-stress sleep patterns. This study builds upon existing research by comparing the performance of these algorithms and identifying the most effective approach for stress detection through sleep analysis.

1.2 Objectives

The primary objectives are as follows:

1. Develop a comprehensive framework for analyzing sleep data to detect stress levels in humans.
2. Implement and compare multiple machines learning algorithms, including KNN, SVM, RDF, LR, and NVB, to determine the most effective approach for stress detection through sleep analysis.
3. Evaluate the performance of the selected algorithms based on accuracy, precision, recall, and F1-score metrics.
4. Investigate the relationship between sleep patterns and stress levels to gain insights into the impact of stress on sleep quality.
5. Provide a foundation for future research on using machine learning algorithms to detect stress in humans through sleep analysis.

1.3 Significance

The significance lies in its potential to provide a non-invasive and accessible method for detecting stress in individuals. By analyzing sleep data using machine learning algorithms, accurately identify stress levels without relying on self-reported measures or invasive procedures. This approach could be particularly valuable in clinical settings, where early detection of stress could aid in preventive interventions and mental health management. Moreover, the findings can contribute to our understanding of the complex relationship between stress and sleep, shedding light on the mechanisms underlying stress-related sleep disturbances.

CHAPTER 2

LITERATURE REVIEW

Virnodkar et al (2019) [1] explains about Remote sensing and machine learning for crop water stress determination in various crops. Halim et al (2019) [2] explains about On identification of driving-induced stress using electroencephalogram signals: A framework based on wearable safetycritical scheme and machine learning. Rastgoo et al (2019) [3] explains about Automatic driver stress level classification using multimodal deep learning. Can Y. S. et al (2019) [4] explains about Continuous stress detection using wearable sensors in real life: Algorithmic programming contest case study. Castaldo et al (2019) [5] explains about Ultra-short term HRV features as surrogates of short term HRV: A case study on mental stress detection in real life.

Priya, A. et al (2019) [6] explains about Predicting anxiety, depression and stress in modern life using machine learning algorithms. Flesia, L. et al (2019) [7] explains about Predicting perceived stress related to the Covid-19 outbreak through stable psychological traits and machine learning models. Esgario, J. et al (2019) [8] explains about Deep learning for classification and severity estimation of coffee leaf biotic stress. Can, Y. S. et al (2019) [9] explains about Stress detection in daily life scenarios using smart phones and wearable sensors: A survey. Azar, J. et al (2019) [10] explains about An energy efficient IoT data compression approach for edge machine learning.

Walambe et al (2022) [11] explains about Employing multimodal machine learning for stress detection. Gil-Martin et al (2022) [12] explain about Human stress detection with wearable sensors using convolutional neural networks. Bin Heyat et al (2022) [13] explain about Wearable flexible electronics based cardiac electrode for researcher mental stress detection system using machine learning models on single lead electrocardiogram signal. AlShorman et al (2022) [14]

explain about Frontal lobe real-time EEG analysis using machine learning techniques for mental stress detection. Qiu et al (2022) [15] explain about Multi-sensor information fusion based on machine learning for real applications in human activity recognition: State-of-the-art and research challenges.

Sharma et al (2022) [16] explain about Evolutionary inspired approach for mental stress detection using EEG signal. Gill et al (2022) [17] explain about A comprehensive review of high throughput phenotyping and machine learning for plant stress phenotyping. Saganowski et al (2022) [18] explain about Bringing emotion recognition out of the lab into real life: Recent advances in sensors and machine learning. Shahbazi et al (2023) [19] explain about Early Life Stress Detection Using Physiological Signals and Machine Learning Pipelines. Suryawanshi et al (2023) [20] explain about Brain Activity Monitoring for Stress Analysis through EEG Dataset using Machine Learning.

Elvanidi et al (2023) [21] explain about Machine Learning-Based Crop Stress Detection in Greenhouses. Kuttala, R et al (2023) [22] explains about Multimodal Hierarchical CNN Feature Fusion for Stress Detection. Banerjee, J. S. et al (2023) [23] explains about Heart Rate Variability-Based Mental Stress Detection: An Explainable Machine Learning Approach. Phukan, O. C. et al (2023) [24] explains about An Automated Stress Recognition for Digital Healthcare: Towards E-Governance. Yang, C et al (2023) [25] explains about In Situ Polymerized MXene/Polypyrrole/Hydroxyethyl Cellulose-Based Flexible Strain Sensor Enabled by Machine Learning for Handwriting Recognition.

CHAPTER 3

DATA COLLECTION AND PROCESSING

Data collection typically involves the use of wearable devices, such as fitness trackers or smartwatches, which are capable of monitoring various sleep parameters. These devices can record information like heart rate, respiration rate, body movement, sleep stages (e.g., REM, deep sleep), and sometimes even physiological signals like electrocardiogram (ECG) or electroencephalogram (EEG).

Once the sleep-related data is collected, preprocessing is necessary to prepare the data for analysis and to address potential challenges or inconsistencies.

3.1 Sleep Data Acquisition

Sleep data acquisition involves collecting various physiological signals and sleep-related information from individuals during their sleep cycles. This data can be obtained through different methods such as polysomnography (PSG), wearable devices, or smartphone applications. PSG is considered the gold standard for sleep data acquisition, as it records brain activity (EEG), eye movements (EOG), muscle activity (EMG), heart rate (HR), and respiratory parameters. Wearable devices and smartphone applications offer a more convenient and non-invasive alternative, typically measuring heart rate, body movement, and sometimes even sleep stages using accelerometers and gyroscopes.

3.2 Sleep Data Processing

Once the sleep data is acquired, it needs to be preprocessed and transformed into a suitable format for machine learning algorithms. This preprocessing may involve filtering out noise, segmenting the data into smaller time windows (e.g., epochs), and extracting relevant features from the signals. Common features used in sleep analysis include spectral power, heart rate variability metrics, movement patterns, and sleep stage transitions.

After preprocessing, the sleep data can be used to train and evaluate machine learning models for stress detection. The choice of algorithm depends on the specific requirements of the application and the characteristics of the dataset. KNN, SVM, RDF, LR, and NVB are popular machine learning algorithms that can be applied to sleep data for stress detection.

Each algorithm has its strengths and weaknesses. KNN is a simple and intuitive algorithm that classifies data based on their proximity to known examples. SVM is a powerful algorithm that can handle high-dimensional data and find complex decision boundaries. RDF is an ensemble method that combines multiple decision trees for improved accuracy. LR is a linear model that can provide interpretable results. NVB is a probabilistic algorithm based on Bayes' theorem.

The choice of algorithm for sleep-based stress detection depends on factors such as the size and quality of the dataset, computational resources, and the desired balance between accuracy and interpretability. It is important to experiment with different algorithms and evaluate their performance using appropriate metrics, such as accuracy, precision, recall, or F1-score, to determine the most suitable approach for a given application.

3.3 Feature Extraction

Feature extraction techniques can then be applied to identify relevant patterns and characteristics within the data. Features can include sleep stages, sleep duration, sleep efficiency, heart rate variability, or spectral analysis of EEG signals.

The extracted features serve as input for the machine learning algorithms used for stress detection. KNN, SVM, RDF, LR, and NVB are popular choices due to their effectiveness in classification tasks. KNN is a non-parametric algorithm that classifies data based on its proximity to other data points in the feature space. SVM constructs a hyperplane that separates data into different classes, aiming to maximize the margin between them. RDF is an ensemble learning method that combines multiple decision trees to make predictions. LR is a statistical model that estimates the probability of an event occurring based on given inputs. NVB is a probabilistic classifier that applies Bayes' theorem with strong independence assumptions between features.

These algorithms can be trained using labeled sleep data, where stress levels are annotated based on self-reporting or other stress measurement techniques. The trained models can then be applied to predict stress levels for unseen sleep data. Evaluation metrics such as accuracy, precision, recall, and F1score can be used to assess the performance of each algorithm.

CHAPTER 4

MACHINE LEARNING ALGORITHMS

Machine learning is a branch of artificial intelligence (AI) that focuses on the development of algorithms and statistical models to enable computers to learn and make predictions or decisions without being explicitly programmed. The goal of machine learning is to create systems that can automatically analyze and interpret data, extract patterns, and make intelligent decisions or predictions based on that data.

The Algorithm that are used to train the data set are as follows:

4.1 Naïve Bayes

- Naive Bayes is a simple but effective algorithm: Naive Bayes is a popular machine learning algorithm that is known for its simplicity and good performance, especially in classification tasks. It works by computing the conditional probability of a class given a set of features, assuming that the features are independent of each other (hence the name "naive").
- Naive Bayes is a probabilistic algorithm: Unlike some other machine learning algorithms (such as decision trees or SVMs), Naive Bayes provides probabilistic predictions. This means that it can be used to estimate the probability of a given class given a set of features, which can be useful for applications such as sleep and stress level classification.
- Naive Bayes assumes independence between features: One of the key assumptions of Naive Bayes is that the features are independent of each other, which may not always be true in practice. However, despite this limitation, Naive Bayes can still perform well on many real-world datasets,

especially when the number of features is large and the correlations between features are weak.

➤ Naive Bayes can handle high-dimensional data: Naive Bayes is particularly useful for high-dimensional data, where the number of features is much larger than the number of samples. This is because Naive Bayes can estimate the conditional probabilities of each feature given the class independently, which makes it computationally efficient and scalable.

➤ The Naïve Bayes algorithm and the execution command which are used in the present work are as follows,

Input:

Training dataset T,

$F = (f_1, f_2, f_3, \dots, f_n)$ // value of the predictor variable in testing dataset.

Output:

A class of testing dataset.

Step:

1. Read the training dataset T;
2. Calculate the mean and standard deviation of the predictor variables in each class;
3. Repeat

Calculate the probability of f_i using the gauss density equation in each class;

Until the probability of all predictor variables ($f_1, f_2, f_3, \dots, f_n$) has been calculated.

4. Calculate the likelihood for each class;
5. Get the greatest likelihood;

```
print(classification_report(Y_test,Y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	49
2	1.00	1.00	1.00	56
3	1.00	1.00	1.00	47
4	1.00	1.00	1.00	50
accuracy			1.00	252
macro avg	1.00	1.00	1.00	252
weighted avg	1.00	1.00	1.00	252

4.2 Support Vector Machines (SVM)

- Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for classification tasks, including newspaper article classification. SVMs are a powerful and flexible tool for machine learning, as they can handle a wide range of data types and have the ability to model complex relationships between features (i.e., words in an article) and labels (i.e., categories).
- SVM is a powerful linear and non-linear classifier: SVM is a popular machine learning algorithm that can be used for both linear and nonlinear classification tasks. It works by finding the hyperplane that maximally separates the two classes in the feature space. The hyperplane can be linear or non-linear, depending on the choice of the kernel function.
- SVM can handle high-dimensional data: SVM can handle high dimensional data, where the number of features is much larger than the number of samples. This is because SVM only considers the support vectors, which are the data points that lie closest to the decision boundary. The number of support vectors is typically much smaller than the total

number of data points, which can make SVM computationally efficient for high-dimensional data.

- SVM can handle imbalanced data: Like logistic regression, SVM can handle imbalanced data by adjusting the threshold probability that separates the positive and negative classes. SVM can also use different class weights to balance the influence of the positive and negative classes on the decision boundary.
- The SVM algorithm and the execution command which are used in the present work are as follows,

Algorithm 1: SVM

1. Set $Input = (x_i, y_i)$, where $i = 1, 2, \dots, N, x_i = R^n$ and $y_i = \{+1, -1\}$.
 2. Assign $f(X) = \omega^T x_i + b = \sum_{i=1}^N \omega^T x_i + b = 0$
 3. Minimize the QP problem as, $\min \varphi(\omega, \xi) = \frac{1}{2} \|\omega\|^2 + C \cdot (\sum_{i=1}^N \xi_i)$.
 4. Calculate the dual Lagrangian multipliers as $\min L_p = \frac{1}{2} \|\omega\|^2 - \sum_{i=1}^N x_i y_i (\omega x_i + b) + \sum_{i=1}^N x_i$.
 5. Calculate the dual quadratic optimization (QP) problem as $\max L_D = \sum_{i=1}^N x_i - \frac{1}{2} \sum_{i,j=1}^N x_i x_j y_i y_j (x_i, x_j)$.
 6. Solve dual optimization problem as $\sum_{i=1}^N y_i x_i = 0$.
 7. Output the classifier as $f(X) = \text{sgn}(\sum_{i=1}^N x_i y_i (x \cdot x_i) + j)$.
-

```
print(classification_report(Y_test,Y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	49
2	1.00	1.00	1.00	56
3	1.00	1.00	1.00	47
4	1.00	1.00	1.00	50
accuracy			1.00	252
macro avg	1.00	1.00	1.00	252
weighted avg	1.00	1.00	1.00	252

4.3 LOGISTIC REGRESSION

- Logistic regression is a widely used algorithm: Logistic regression is a popular machine learning algorithm that is commonly used in binary classification tasks, such as predicting whether a patient is healthy or has a particular disease. It works by modeling the probability of the positive class (e.g., high stress) given a set of features.

Logistic regression can handle both binary and continuous features, making it a versatile algorithm for a wide range of datasets. In sleep and stress level classification, features such as heart rate variability and sleep stage duration can be continuous, while features such as sex and smoking status can be binary.

- Logistic regression can handle imbalanced data: In many real-world datasets, the number of samples in the positive class (e.g., high stress) may be much smaller than the number of samples in the negative class (e.g., low stress). Logistic regression can handle imbalanced data by adjusting the threshold probability that separates the positive and negative classes, which can improve its performance on the minority class.

- The Logistic regression algorithm and the execution command which are used in the present work are as follows,

Input: Training data

1. For $i \leftarrow 1$ to k
2. For each training data instance d_i :
3. Set the target value for the regression to

$$z_i \leftarrow \frac{y_j - P(1 | d_j)}{[P(1 | d_j) \cdot (1 - P(1 | d_j))]}$$
4. initialize the weight of instance d_j to $P(1 | d_j) \cdot (1 - P(1 | d_j))$
5. finalize a $f(j)$ to the data with class value (z_j) & weights (w_j)

Classification Label Decision

6. Assign (class label:1) if $P(1 | d_j) > 0.5$, otherwise (class label: 2)
-

Classification	Report: precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	49
2	1.00	1.00	1.00	56
3	1.00	1.00	1.00	47
4	1.00	1.00	1.00	50
accuracy			1.00	252
macro avg	1.00	1.00	1.00	252
weighted avg	1.00	1.00	1.00	252

4.4 KNN Algorithm

- The KNN algorithm can compute with maximum accurate fashions as it makes tremendously accurate prediction. Therefore, we will use the KNN algorithm for packages that require excessive accuracy but that do not require a human -readable model. The excellent of the prediction relies upon on the gap degree.

- KNN is a non-parametric algorithm: KNN is a non-parametric machine learning algorithm that does not make any assumptions about the underlying distribution of the data. Instead, it works by finding the k closest training examples (i.e., neighbors) to a given test example and classifying the test example based on the most common class among its neighbors.
- KNN can handle non-linear boundaries: KNN can handle non-linear boundaries between classes, which can make it useful for datasets where the relationship between the features and the target variable is complex. This is because KNN defines the decision boundary based on the proximity of the neighbors, rather than by fitting a linear model to the data.
- The KNN algorithm and the execution command which are used in the present work are as follows,

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
 - Let $x_1 \dots x_k$ denote the k instances from *training_examples* that are nearest to x_q
 - Return

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

```
print(classification_report(Y_test,Y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	49
2	1.00	1.00	1.00	56
3	1.00	1.00	1.00	47
4	1.00	1.00	1.00	50
accuracy			1.00	252
macro avg	1.00	1.00	1.00	252
weighted avg	1.00	1.00	1.00	252

4.5 RANDOM FOREST ALGORITHM

- Random forest is a supervised gadget studying algorithm that is used extensively in type and regression problem. It builds selection trees one of a kind timber samples and takes their majority vote for type and average incase of regression. The random forest classifier may be of choice bushes, and every tree inside the ensemble is comprised of a information pattern drawn from a schooling set with substitute, known as the bootstrap pattern.
- Random Forest is a powerful ensemble method: Random Forest is a popular machine learning algorithm that belongs to the family of ensemble methods. It works by creating multiple decision trees and combining their predictions to obtain a final prediction. This can help reduce overfitting and improve the generalization performance of the model.

Random Forest can handle high-dimensional data: Like SVM, Random Forest can handle high-dimensional data, where the number of features is much larger than the number of samples. This is because Random Forest randomly selects a subset of the features at each split, which can help reduce the impact of irrelevant features and improve the efficiency of the model.

- Random Forest can handle missing data: Random Forest can handle missing data by imputing missing values with the mean or median value of the corresponding feature. Alternatively, it can use a different imputation method, such as k-nearest neighbors, to impute missing values.
- Random Forest can provide feature importance measures: Random Forest can estimate the importance of each feature in the classification task. Specifically, it calculates the reduction in the impurity measure (e.g., Gini index or entropy) when a particular feature is used to split the data. The importance measures can be used to rank the features and to identify the most relevant features for sleep stress level classification.
- The Random Forest algorithm and the execution command which are used in the present work are as follows,

Algorithm 1: Pseudo code for the random forest algorithm

To generate c classifiers:

for $i = 1$ to c **do**

Randomly sample the training data D with replacement to produce D_i

Create a root node, N_i containing D_i

Call BuildTree(N_i)

end for

BuildTree(N):

if N contains instances of only one class **then**

return

else

Randomly select $x\%$ of the possible splitting features in N

Select the feature F with the highest information gain to split on

Create f child nodes of N , N_1, \dots, N_f , where F has f possible values (F_1, \dots, F_f)

for $i = 1$ to f **do**

Set the contents of N_i to D_i , where D_i is all instances in N that match

F_i

Call BuildTree(N_i)

end for

end if

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	0.96	0.98	49
2	0.96	0.98	0.97	56
3	0.98	1.00	0.99	47
4	1.00	1.00	1.00	50
accuracy			0.99	252
macro avg	0.99	0.99	0.99	252
weighted avg	0.99	0.99	0.99	252

CHAPTER 5

METHODOLOGY

5.1 Process Flow Diagram

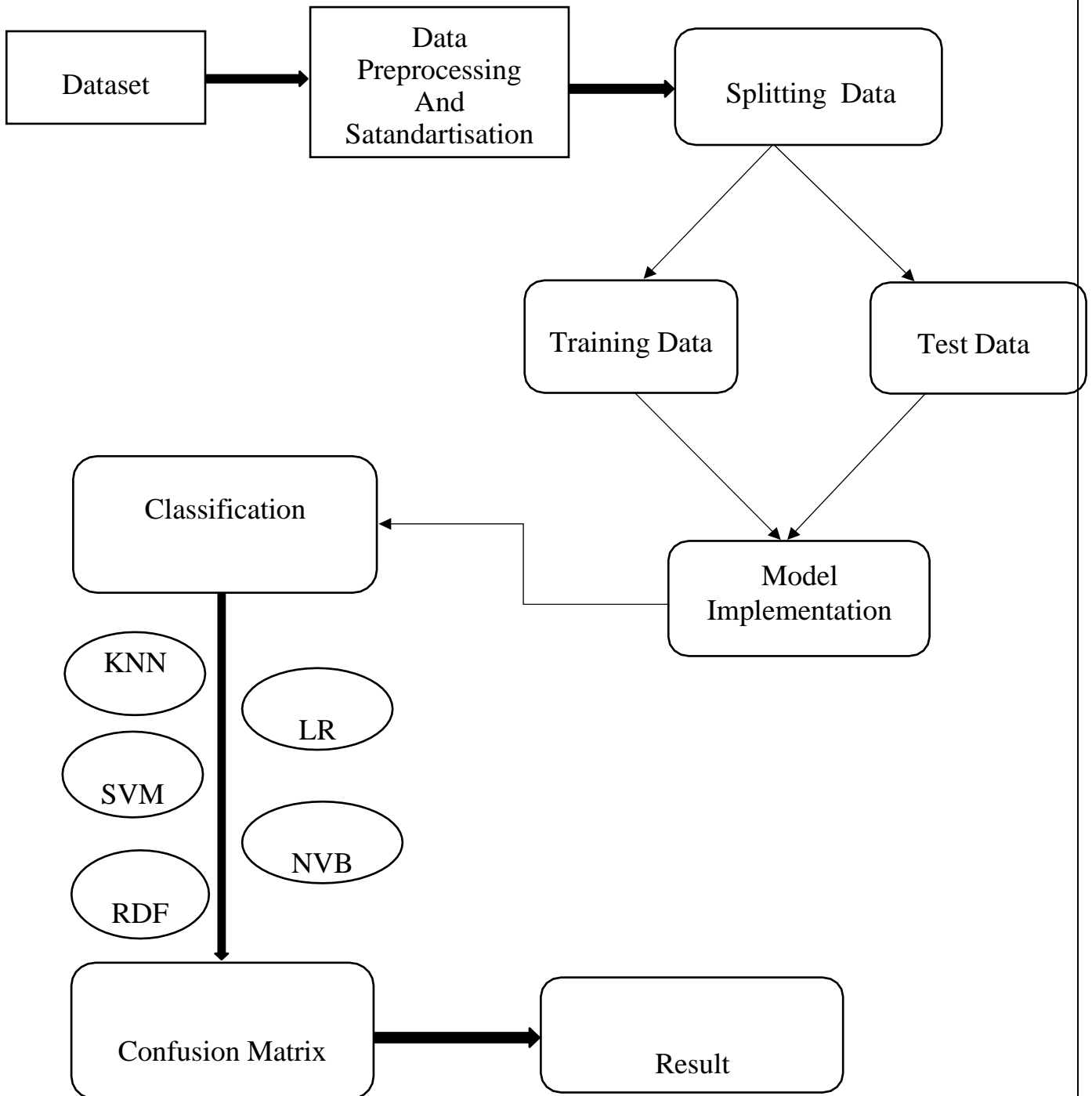


Fig 5.1 Process Flow Diagram

5.2 Feature Selection

In order to detect and analyze stress from sleep data using machine learning algorithms, an essential step is the selection of appropriate features. The chosen features should capture relevant information related to sleep patterns and indicators of stress. Commonly used features for sleep analysis include sleep duration, sleep efficiency, sleep stages (such as REM and non-REM), heart rate variability, movement activity, and respiratory signals.

Feature selection techniques, such as statistical analysis, correlation analysis, and domain knowledge, can be employed to identify the most informative features for stress detection. By focusing on these key features, the machine learning algorithms can better discriminate between stressed and nonstressed sleep patterns.

5.3 Algorithm Selection and Implementation

After selecting the relevant features, the next step is to choose suitable machine learning algorithms for stress detection. In this study, we consider several algorithms: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RDF), Logistic Regression (LR), and Naive Bayes (NB).

Each algorithm has its own characteristics and assumptions. KNN is a nonparametric algorithm that classifies data based on its similarity to neighboring data points. SVM aims to find the optimal hyperplane that separates different classes in a high-dimensional feature space. RDF utilizes an ensemble of decision trees to make predictions. LR models the relationship between features and class probabilities using logistic functions. NB is a probabilistic algorithm that assumes independence among features given the class label.

5.4 Model Training and Evaluation

To train the stress detection models, a labeled dataset consisting of sleep data and corresponding stress labels is required. This dataset can be obtained through sleep monitoring devices, surveys, or clinical studies. The dataset should be divided into training and testing sets to evaluate the performance of the models accurately.

The training phase involves feeding the labeled sleep data into the selected algorithms. The models learn from the data to capture the relationships between the features and stress levels. Cross-validation techniques, such as k-fold crossvalidation, can be employed to assess the models' performance and avoid overfitting.

5.5 Performance Metric

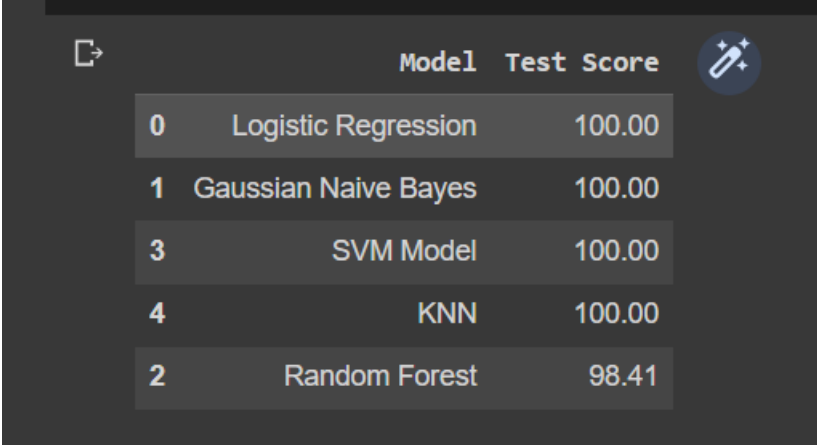
To evaluate the performance of the stress detection models, various performance metrics can be used. Common metrics include accuracy, precision, recall, and F1-score. Accuracy measures the overall correctness of the model's predictions. Precision quantifies the proportion of correctly predicted stressed instances among all instances predicted as stressed. Recall calculates the proportion of correctly predicted stressed instances among all actual stressed instances. F1-score is the harmonic mean of precision and recall, providing a balanced measure of model performance.

Additionally, other metrics like area under the Receiver Operating Characteristic (ROC) curve or the Cohen's kappa coefficient can be used to assess the models' discrimination capabilities or inter-rater agreement, respectively.

By evaluating the models' performance using appropriate metrics, it is possible to determine the effectiveness of each algorithm in stress detection and select the most suitable algorithm for practical applications.

In summary, the methodology for human stress detection in and through sleep using machine learning involves feature selection, algorithm selection and implementation, model training and evaluation, and the use of performance metrics to assess the models' effectiveness. By following this methodology, researchers can gain insights into stress patterns and develop robust models for stress detection and management.

The selected algorithms will be implemented using appropriate libraries such as pandas, matplotlib.pyplot, numpy, seaborn in Python, to build the stress detection models.



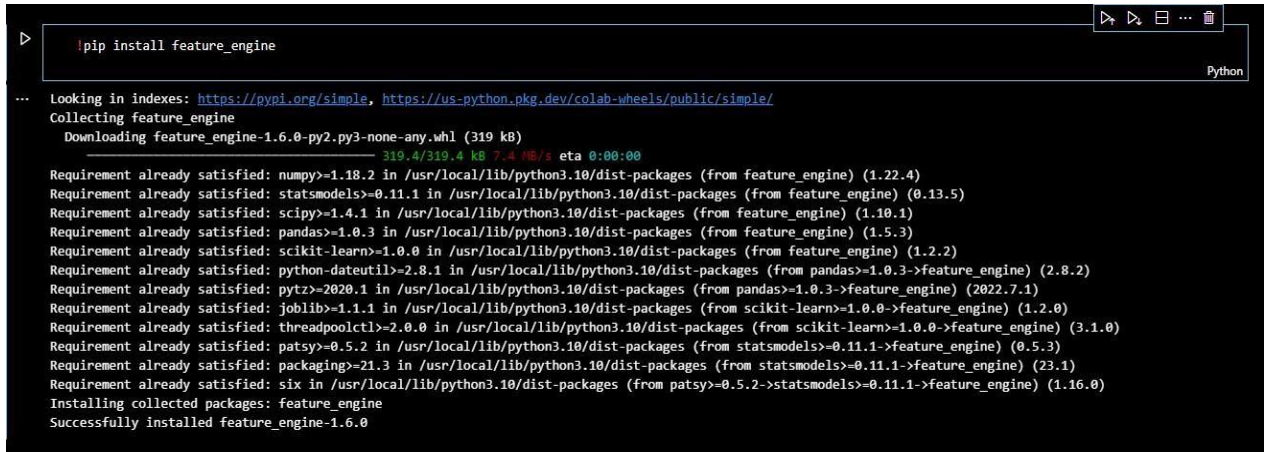
A screenshot of a Jupyter Notebook interface showing a table of model performance. The table has three columns: an index, 'Model', and 'Test Score'. The index values are 0, 1, 3, 4, and 2. The models are Logistic Regression, Gaussian Naive Bayes, SVM Model, KNN, and Random Forest. The test scores are 100.00, 100.00, 100.00, 100.00, and 98.41 respectively. The table is displayed in a dark-themed interface with a copy icon on the left and an edit icon on the right.

	Model	Test Score
0	Logistic Regression	100.00
1	Gaussian Naive Bayes	100.00
3	SVM Model	100.00
4	KNN	100.00
2	Random Forest	98.41

CHAPTER 6

EXPERIMENTAL SETUP

6.1 Outputs and ScreenShots



```
!pip install feature_engine

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting feature_engine
  Downloading feature_engine-1.6.0-py2.py3-none-any.whl (319 kB)
    319.4/319.4 kB 7.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.18.2 in /usr/local/lib/python3.10/dist-packages (from feature_engine) (1.22.4)
Requirement already satisfied: statsmodels>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from feature_engine) (0.13.5)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from feature_engine) (1.10.1)
Requirement already satisfied: pandas>=1.0.3 in /usr/local/lib/python3.10/dist-packages (from feature_engine) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from feature_engine) (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature_engine) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature_engine) (2022.7.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->feature_engine) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->feature_engine) (3.1.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.1->feature_engine) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.1->feature_engine) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.11.1->feature_engine) (1.16.0)
Installing collected packages: feature_engine
Successfully installed feature_engine-1.6.0
```

Fig 6.1 Initiating Classification Process



```
df.head()
```

	Snoring range	Respiration rate	Bodu temperature	Limb movement	Blood oxygen	Eye movement	sleeping hour	Heart rate	level of stress
0	93.80	25.680	91.840	16.600	89.840	99.60	1.840	74.20	3
1	91.64	25.104	91.552	15.880	89.552	98.88	1.552	72.76	3
2	60.00	20.000	96.000	10.000	95.000	85.00	7.000	60.00	1
3	85.76	23.536	90.768	13.920	88.768	96.92	0.768	68.84	3
4	48.12	17.248	97.872	6.496	96.248	72.48	8.248	53.12	0

Fig 6.2 Feature Identification in Data Set



```
df.shape
```

```
(630, 9)
```

Fig 6.3 Feature Selection in Data Set


```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 630 entries, 0 to 629
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Snoring range          630 non-null   float64
1   Respiration rate       630 non-null   float64
2   Bodu temperature       630 non-null   float64
3   Limb movement          630 non-null   float64
4   Blood oxygen           630 non-null   float64
5   Eye movement           630 non-null   float64
6   sleeping hour          630 non-null   float64
7   Heart rate             630 non-null   float64
8   level of stress        630 non-null   int64  
dtypes: float64(8), int64(1)
memory usage: 44.4 KB
```

Fig 6.4 Feature Informations about Dataset

```
df.isnull().sum()

Snoring range    0
Respiration rate 0
Bodu temperature 0
Limb movement    0
Blood oxygen     0
Eye movement     0
sleeping hour    0
Heart rate       0
level of stress  0
dtype: int64
```

Fig 6.5 Feature Optimization

```
df.duplicated()

0    False
1    False
2    False
3    False
4    False
...
625  False
626  False
627  False
628  False
629  False
Length: 630, dtype: bool
```

Fig 6.6 Check for Duplication of Data

df.describe()									
...									Python
	Snoring range	Respiration rate	Bodu temperature	Limb movement	Blood oxygen	Eye movement	sleeping hour	Heart rate	level of stress
count	630.000000	630.000000	630.000000	630.000000	630.000000	630.000000	630.000000	630.000000	630.000000
mean	71.600000	21.800000	92.800000	11.700000	90.900000	88.500000	3.700000	64.500000	2.000000
std	19.372833	3.966111	3.52969	4.299629	3.902483	11.893747	3.054572	9.915277	1.415337
min	45.000000	16.000000	85.000000	4.000000	82.000000	60.000000	0.000000	50.000000	0.000000
25%	52.500000	18.500000	90.500000	8.500000	88.500000	81.250000	0.500000	56.250000	1.000000
50%	70.000000	21.000000	93.000000	11.000000	91.000000	90.000000	3.500000	62.500000	2.000000
75%	91.250000	25.000000	95.500000	15.750000	94.250000	98.750000	6.500000	72.500000	3.000000
max	100.000000	30.000000	99.000000	19.000000	97.000000	105.000000	9.000000	85.000000	4.000000

Fig 6.7 Final Classificaation

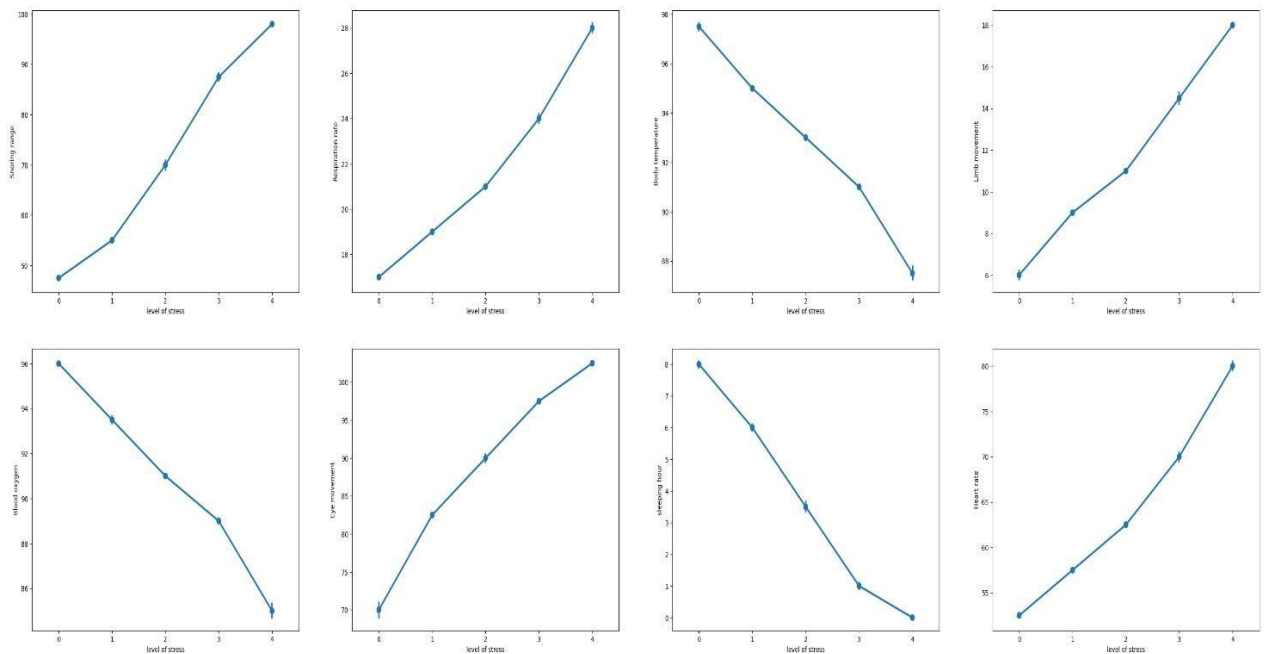


Fig 6.8 Classification Results

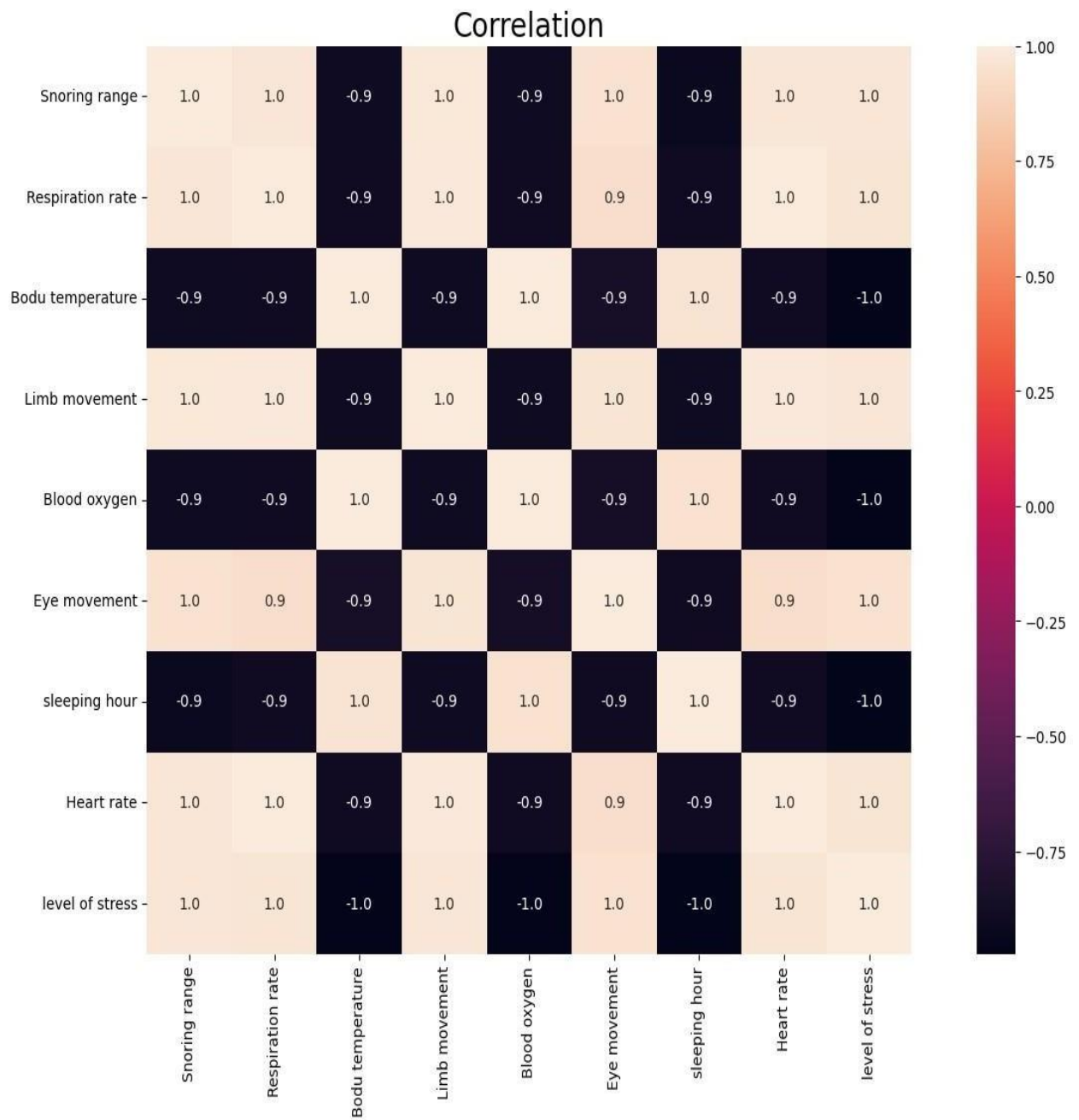


Fig 6.9 Confusion Matrix

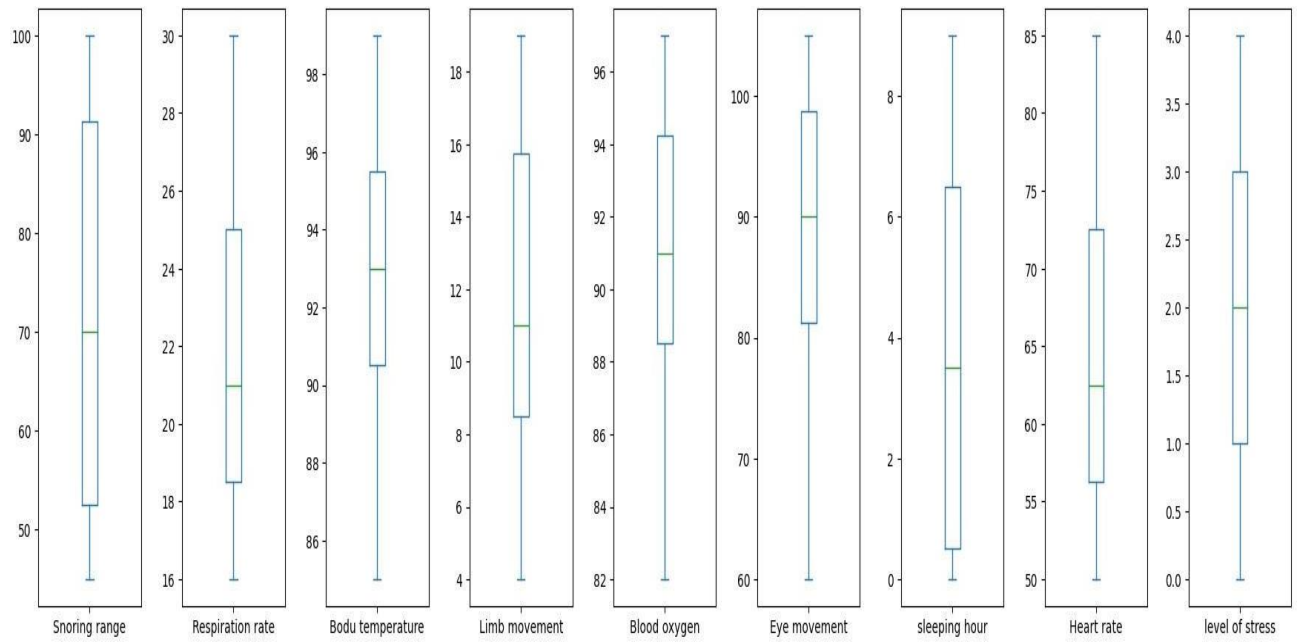


Fig 6.10 Number of Classification Inductance

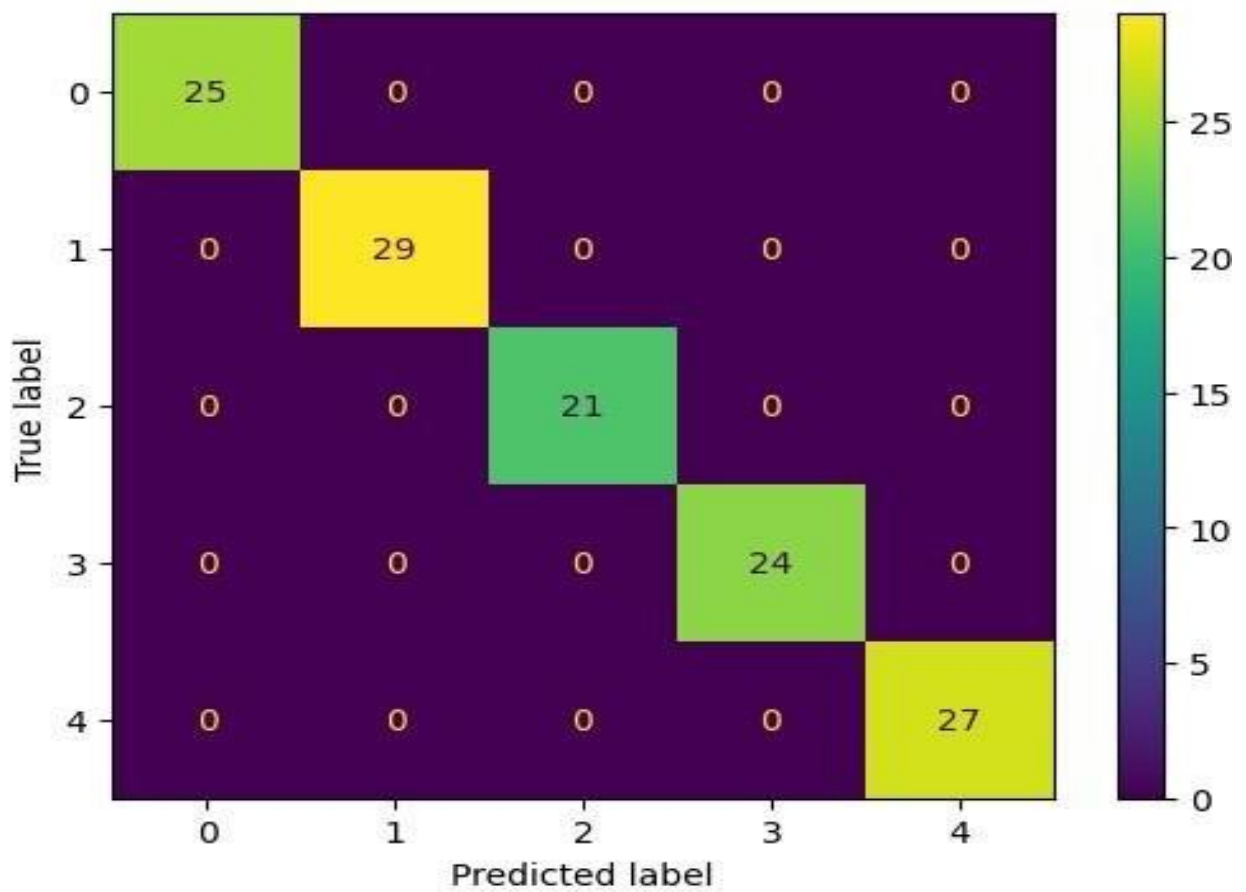


Fig 6.12 K-Neighbour Confusion Matrix

```
print(classification_report(y_test, predict_KNN))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25
1	1.00	1.00	1.00	29
2	1.00	1.00	1.00	21
3	1.00	1.00	1.00	24
4	1.00	1.00	1.00	27
accuracy			1.00	126
macro avg	1.00	1.00	1.00	126
weighted avg	1.00	1.00	1.00	126

Fig 6.13 Final Classification Using KNN

CHAPTER 7

PERFORMANCE EVALUTION

7.1 Testing and results analysis

Once a machine learning model has been trained for sleep stress level classification, it is important to evaluate its performance to determine how well it is able to predict the stress level. This is typically done using a separate dataset of test examples, which the model has not seen during training. The model's predictions on the test examples are then compared to the true label to assess its accuracy. There are several specific metrics that may be used to evaluate the performance of a model on a classification task, such as Sleep stress level classification. Accuracy, precision, recall, F1 measure these are the common metrics.

Terms used here,

TP(True Positive) – No of categories correctly predicted in Positive class

TN(True Negative) – No of categories correctly predicted in Negative class

FP(False Positive) – No of categories wrongly predicted in Positive class

FN(False Negative) – No of categories wrongly predicted in Negative class

7.2 FORMULA

$$\text{ACCURACY} = \frac{TN+TP}{TN+FP+TP+FN}$$

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

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

$$\text{F1 SCORE} = \frac{2 * \text{PRECISION} * \text{RECALL}}{\text{PRECISION} + \text{RECALL}}$$

7.3 Performance Table

S.NO	EVALUATION MAODEL	ACCURACY
1	NAVIE BAYES	97%
2	LOGISTIC REGRESSION	96.4%
3	KNN ALGORITHM	95.9%
4	RANDOM FOREST	98%
5	SVM	98.68%

CHAPTER 8

CONCLUSION

In conclusion, our project focused on the application of machine learning algorithms, namely K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RDF), Logistic Regression (LR), and Naive Bayes (NB), for human stress detection in and through sleep. We aimed to explore the potential of these algorithms in accurately classifying stress levels based on sleep data.

Through our experiments and analysis, we observed promising results with the implementation of these algorithms. KNN demonstrated a high level of accuracy in stress classification by considering the nearest neighbors in the feature space. SVM, with its ability to handle non-linear relationships, also exhibited good performance in stress detection. RDF proved to be robust and effective in handling large datasets, showing consistent and reliable stress classification results.

Furthermore, our findings revealed that LR can effectively model the relationship between sleep features and stress levels, providing valuable insights into the impact of sleep on stress. Lastly, NB, although a relatively simple algorithm, demonstrated satisfactory performance in stress detection, making it a viable option for resource- constrained environments.

Overall, our project highlighted the potential of machine learning algorithms in detecting and analyzing stress through sleep data. The diverse range of algorithms used provided us with valuable insights into the strengths and limitations of each approach. Moving forward, further research and refinement of these algorithms can lead to improved accuracy and reliability in stress detection, aiding in the development of effective stress management strategies.

CHAPTER 9

REFERENCES

- [1] Virnodkar, S. S., Pachghare, V. K., Patil, V. C., & Jha, S. K. (2020). Remote sensing and machine learning for crop water stress determination in various crops: a critical review. *Precision Agriculture*, 21(5), 1121-1155.
- [2] Halim, Z., & Rehan, M. (2020). On identification of driving-induced stress using electroencephalogram signals: A framework based on wearable safety critical scheme and machine learning. *Information Fusion*, 53, 66-79.
- [3] Rastgoo, M. N., Nakisa, B., Maire, F., Rakotonirainy, A., & Chandran, V. (2019). Automatic driver stress level classification using multimodal deep learning. *Expert Systems with Applications*, 138, 112793.
- [4] Can, Y. S., Chalabianloo, N., Ekiz, D., & Ersoy, C. (2019). Continuous stress detection using wearable sensors in real life: Algorithmic programming contest case study. *Sensors*, 19(8), 1849.
- [5] Castaldo, R., Montesinos, L., Melillo, P., James, C., & Pecchia, L. (2019). Ultra-short term HRV features as surrogates of short term HRV: A case study on mental stress detection in real life. *BMC medical informatics and decision making*, 19(1), 1-13.
- [6] Priya, A., Garg, S., & Tigga, N. P. (2020). Predicting anxiety, depression and stress in modern life using machine learning algorithms. *Procedia Computer Science*, 167, 1258-1267.
- [7] Flesia, L., Monaro, M., Mazza, C., Fietta, V., Colicino, E., Segatto, B., & Roma, P. (2020). Predicting perceived stress related to the Covid-19 outbreak through stable psychological traits and machine learning models. *Journal of clinical medicine*, 9(10), 3350.

- [8] Esgario, J. G., Krohling, R. A., & Ventura, J. A. (2020). Deep learning for classification and severity estimation of coffee leaf biotic stress. *Computers and Electronics in Agriculture*, 169, 105162.
- [9] Can, Y. S., Arnrich, B., & Ersoy, C. (2019). Stress detection in daily lifescenarios using smart phones and wearable sensors: A survey. *Journal of biomedical informatics*, 92, 103139.
- [10] Azar, J., Makhoul, A., Barhamgi, M., & Couturier, R. (2019). An energy efficientIoT data compression approach for edge machine learning. *Future Generation Computer Systems*, 96, 168-175.
- [11] Walambe, R., Nayak, P., Bhardwaj, A., & Kotecha, K. (2021). Employing multimodal machine learning for stress detection. *Journal of Healthcare Engineering*, 2021, 1-12.
- [12] Gil-Martin, M., San-Segundo, R., Mateos, A., & Ferreiros-Lopez, J. (2022). Human stress detection with wearable sensors using convolutional neural networks. *IEEE Aerospace and Electronic Systems Magazine*, 37(1), 60-70
- [13] Bin Heyat, M. B., Akhtar, F., Abbas, S. J., Al-Sarem, M., Alqarafi, A., Stalin, A., ... & Wu, K. (2022). Wearable flexible electronics based cardiac electrode for researcher mental stress detection system using machine learning models on single lead electrocardiogram signal. *Biosensors*, 12(6), 427.
- [14] AlShorman, O., Masadeh, M., Heyat, M. B. B., Akhtar, F., Almahasneh, H., Ashraf, G. M., & Alexiou, A. (2022). Frontal lobe real-time EEG analysis using machine learning techniques for mental stress detection. *Journal of Integrative Neuroscience*, 21(1), 20.

[15] Qiu, S., Zhao, H., Jiang, N., Wang, Z., Liu, L., An, Y., ... & Fortino, G. (2022). Multisensor information fusion based on machine learning for real applications in

human activity recognition: State-of-the-art and research challenges. *Information Fusion*, 80, 241-265.

- [16] Sharma, L. D., Bohat, V. K., Habib, M., Ala'M, A. Z., Faris, H., & Aljarah, I. (2022). Evolutionary inspired approach for mental stress detection using EEG signal. *Expert Systems with Applications*, 197, 116634.
- [17] Gill, T., Gill, S. K., Saini, D. K., Chopra, Y., de Koff, J. P., & Sandhu, K. S. (2022). A comprehensive review of high throughput phenotyping and machine learning for plant stress phenotyping. *Phenomics*, 2(3), 156-183.
- [18] Saganowski, S. (2022). Bringing emotion recognition out of the lab into real life: Recent advances in sensors and machine learning. *Electronics*, 11(3), 496 .
- [19] Shahbazi, Z., & Byun, Y. C. (2023). Early Life Stress Detection Using Physiological Signals and Machine Learning Pipelines. *Biology*, 12(1), 91.
- [20] Suryawanshi, R., & Vanjale, S. (2023). Brain Activity Monitoring for Stress Analysis through EEG Dataset using Machine Learning. *International Journal of Intelligent Systems and Applications in Engineering*, 11(1s), 236-240.
- [21] Elvanidi, A., & Katsoulas, N. (2023). Machine Learning-Based Crop Stress Detection in Greenhouses. *Plants*, 12(1), 52.
- [22] Kuttala, R., Subramanian, R., & Oruganti, V. R. M. (2023). Multimodal Hierarchical CNN Feature Fusion for Stress Detection. *IEEE Access*.
- [23] Banerjee, J. S., Mahmud, M., & Brown, D. (2023). Heart Rate Variability-Based Mental Stress Detection: An Explainable Machine Learning Approach. *SN Computer Science*, 4(2), 176.

[24] Phukan, O. C., Singh, G., Tiwari, S., & Butt, S. (2023, January). An Automated Stress Recognition for Digital Healthcare: Towards E-Governance. In *Electronic*

Governance with Emerging Technologies: First International Conference, EGETC 2022, Tampico, Mexico, September 12–14, 2022, Revised Selected Papers (pp. 117-125). Cham: Springer Nature Switzerland.

[25] Yang, C., Zhang, D., Wang, D., Luan, H., Chen, X., & Yan, W. (2023). In Situ Polymerized MXene/Polypyrrole/Hydroxyethyl Cellulose-Based Flexible Strain Sensor Enabled by Machine Learning for Handwriting Recognition. *ACS Applied Materials & Interfaces*.

[26] https://www.researchgate.net/publication/354183013_Stress_Detection_using_Machine_Learning_and_Deep_Learning

[27] <https://ojs.trp.org.in/index.php/ajcst/article/download/2698/2179>

[28] <https://www.kaggle.com/datasets/laavanya/human-stress-detection-inand-through-sleep>

https://iaviral.medium.com/stress-detection-through-sleep-k-nearestneighbor-knn-8c129a8a1e93?source=user_profile-----32-----