Cognitive Stress Level Prediction based on Sensory wearables and IOT data

Shikha Singh

Department of Applied Data Science, San Jose State University

DATA 270: Data Analytics Process

Dr. Linsey Pang

May 13, 2022

Abstract

Rising stress levels have caught everyone’s attention like never before. Stress can originate from multitude of sources like academic institutions, workplaces, personal relationships, and society. Increased stress is associated with anxiety issues, chronic heart diseases, high blood pressure and nervous breakdown. Through Leveraging supervised machine learning techniques, Healthier lifestyle habits will be encouraged by providing recommendations on ways to combat increased stress levels. The scope of this project is to create a stress monitoring system using wearable devices which will examine the Change in blood volume pulse (BVP), heart rate variation (HRV) with the device during stress inducing task. Logistic regression (LR), KNN, Random Forest, and Xtreme Gradient Boosting traditional machine learning algorithms along with Deep Neural Network (DNN) will be used to classify stress levels. The performance of the model could be enhanced using k-fold cross validation.

Also, previous studies showed that with linear and non-linear HRV features and using Random Forest machine learning model higher accuracy was achieved. We are aiming to predict stress levels with 80% accuracy approximately. The performance and results of the algorithms will be compared based on Accuracy, Precision, Recall, AUC-ROC, log-loss, F1-score, confusion matrix. The Project will positively impact individuals’ well-being by detecting stress in the initial stages and preventing long-term health issues. Based on classification and detection of stress levels, suggestions would be provided to individuals to facilitate their mental wellness.

Keywords: *Stress level, Heart rate variation, heart disease*

Introduction

Project Background and Execute Summary

In this present world where life is accelerating at a fast pace, stress has become a global issue that is impacting millions of lives. Stress may arise from a variety of places like social expectations, professional responsibilities personal relationships, and academic constraints.

Longer exposure to stress has many adverse effects which include anxiety disorders, hypertension, chronic heart illnesses, and neurological breakdowns. Now is a very important time to track and manage stress levels which reveals a big void in today's wellness and healthcare programs, especially when it comes to integrating technology with individual health monitoring.

Understanding the importance of early stress detection, this project attempts to utilize the advancements of the Internet of Things (IOT) and sensor wearable technology to close the gap. Continuous monitoring of stress indicators like Heart Rate Variability (HRV), Blood Volume Pulse (BVP), etc. offers a great way to identify each person's stress level, allowing early treatments and a better lifestyle.

The project's goal is to estimate and classify people's stress levels using the information from the wearable devices. The accessibility and reliability of traditional stress assessment methods are limited since they depend on self-reports.

This project aims to develop a stress monitoring system with an accuracy of about 80% in predicting stress levels, driven by the possibility of having a real impact on people's mental health. By doing this, it aims to detect stress at an early stage, which helps in preventing the development of more serious health issues.

As part of the process, data obtained from wearable devices is analyzed using supervised machine learning techniques, such as Deep Neural Network (DNN), Logistic Regression, XG Boost, Random Forest, and k nearest neighbors (KNN) algorithms. The study will use previous research that shows how effectively Random functions at high accuracy to investigate both linear and non-linear HRV characteristics. Furthermore, by employing k-fold cross- validation, the model's performance will be enhanced, ensuring its dependability and robustness.

The projected benefits of this research go much beyond technology; It has important uses cases for both individual and national health. Through providing practical insights on stress levels and the recommendation of stress relieving measures, the initiative contributes to the larger objective of promoting mental wellness. Enhancing the subject of predictive health analytics will be the comparative examination of different machine learning algorithms based on metrics like F1 Score and confusion matrix.

In summary, this project addresses a pressing health issue and lays the groundwork for future developments in individual health monitoring and management. With this preventive healthcare may enter a new era in which people are empowered to take early actions to maintain their mental health.

Project Requirements

The basic functional requirement for a stress monitor system is that the system can detect the stress condition of the target efficiently and accurately according to the measurement data. Since the data comes from various sources, standardization or normalization applied on input features ensures consistent scaling. Simultaneously, to the consideration of time series features, the data records should be over consistent intervals. In that case, time-series analysis techniques are required.

As for the AI-Powered requirements, this paper involves five machine learning techniques: Logistic Regression (LR), K nearest neighbors (KNN), XG Boost, Random Forest (RF) and Deep Neural Network (DNN) to construct the stress monitor system. The final performance is determined by several metrics, including Accuracy, Precision, Recall and F1-Score. The necessary python packages in this project include Scikit-learn, torch et al.

Regarding the robustness of the system, the metadata must cover a diverse target population, including different age, sex and region groups et al. In that case, the dataset used in this project is combined based on four data sources. Each data source contains several separated csv files, consisting of various physiological measurement data, such as participants' heart rate (HR), electrodermal activity (EDA) and blood volume pulse (BVP) et al. Due to the diversity of data, missing data should be handled and to ensure the data consistency, sampling methods should be utilized. Meanwhile, partial unlabeled data will be labeled manually.

Project Deliverables

Report

Our report consists of detailed information about multiple topics which includes the abstract, an introduction to the topic, and clear descriptions of the sensory wearables whose information is used in our analysis. As far as the data cleaning and preparation is concerned, our report would explain the various steps that are used as part of the data preprocessing, and various relevant visualizations that rightly represent the data. With respect to data analysis, feature engineering, clear demonstration of the various machine learning models that are used, evaluation of their performance, and eventually the results and the subsequent impact along the various references that are used as part of our cognitive stress level prediction project will be clearly documented in our report. As part of documenting the various machine learning models that we implemented in this project, in addition to describing how the features engineering approach for each individual model is implemented, the report would also contain explanation about the approach of model training along with detailed description about the functionality of the algorithms, and the various measures taken and implemented to improve the performance of the models which is part of the tuning process to achieve best outcomes. The report would also contain information about the performance evaluation of the stress prediction models. This would include attributes such accuracy, precision, recall, F1 score, and other relevant metrics.

Prototypes

A prototype is a model of a product that is developed during the early essential phase of product development and the design process. Wearable device prototypes are made up of sensors that collect user vitals such as heart rate, accelerometer data, electrodermal activity, blood volume pulse data, respiration rate and temperature. All this data is used to develop preliminary machine learning models of prototype type. This gives one the big picture performance of different models as well as what steps or routes can be taken to achieve the most accurate model possible in terms of output. This is done by very carefully studying the prototypes and then finding any ways accuracy can be improved. After looking at the various possible improvements- each of these machine learning models will be deeply analyzed and optimized until better results can be achieved.

Development applications

There are multiple development applications, consisting of sensory data collected from wearables and IoT devices used for preprocessing. The process of preprocessing includes data cleaning, dimensionality reduction, and normalization, in addition to feature extraction, which plays an important role. Development of machine learning models for stress level prediction, which includes logistic regression, K nearest neighbors' random forests, XG Boost and Deep Neural Network (DNN) are some of the other applications. In addition to the above applications, development of visualizations using various parameters of stress level is another utility.

Production Applications

The practical applications of real-time stress monitoring and management are varied and have significant impact on users' health. Wearable sensors and IoT devices continuously collect user data, which is helpful in measurement of stress. The stress prediction solutions, when applied in real-world scenarios such as that of workplaces, schools, and healthcare facilities, with continuous monitoring and evaluation will help in promptly identifying users who are in stress.

This can provide the user and concerned authorities a chance to mitigate it as well. Above all, integration of stress prediction capabilities into currently used wearable devices beside health monitoring platforms can provide personalized recommendations to individuals about their stress levels. This considers the individual's physiological responses and factors affecting their immediate surroundings.

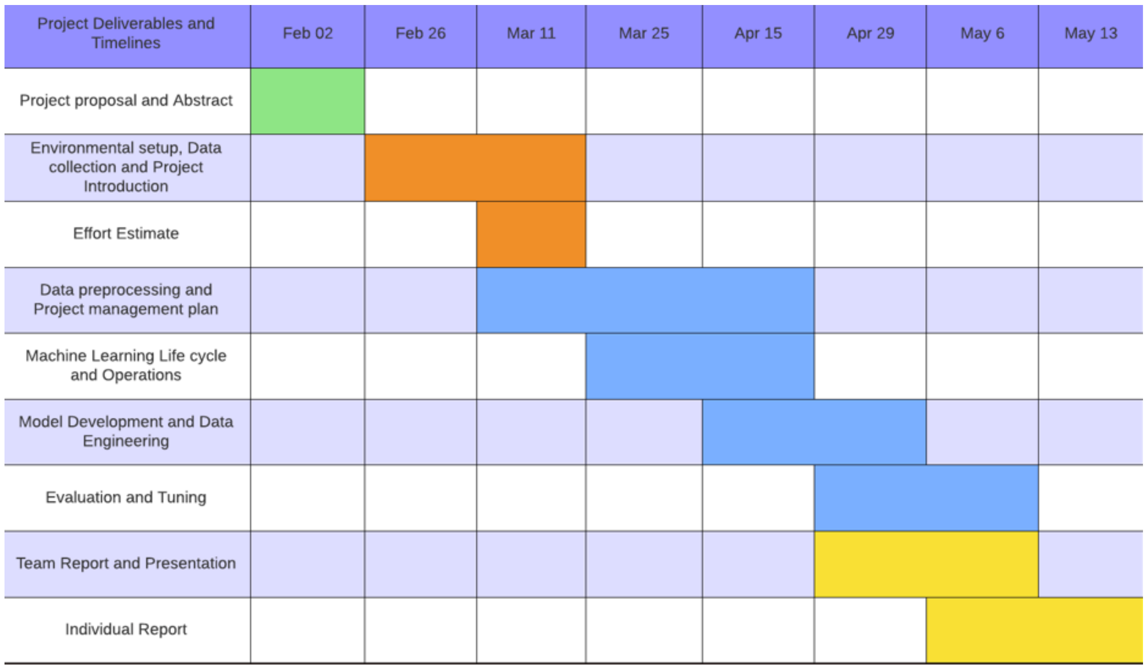
Gantt Chart

The Gantt chart shows the timeline, tasks, sub-tasks, percentage of progress, dates, milestones, deadlines, and resource allocated to the task/sub-task in the form of bar. Such a chart will show all the sub-tasks and how they are co-related to give you a bigger picture. Figure 1 contains the list of all the project deliverables and their due dates

Project Deliverables and Timelines

Figure 1

list of all the project deliverables and their due dates.



Project Deliverables and Timelines	Feb 02	Feb 26	Mar 11	Mar 25	Apr 15	Apr 29	May 6	May 13
Project proposal and Abstract	🟩							
Environmental setup, Data collection and Project Introduction		🟧	🟧					
Effort Estimate			🟧					
Data preprocessing and Project management plan			🟦	🟦	🟦			
Machine Learning Life cycle and Operations				🟦	🟦			
Model Development and Data Engineering					🟦	🟦		
Evaluation and Tuning						🟦	🟦	
Team Report and Presentation						🟨	🟨	
Individual Report							🟨	🟨

Technology and Solution Survey

Existing research and studies took different statistical approaches to solve the stress detection problem. Ç. Çöpürkaya et al. (2023) conducted an experiment with groups of university students in Istanbul, Turkey, where 48 individuals participated in survey-based questionnaire, 20 individuals participated in physiological test in presence of a psychologist. This physiological data was captured by Empatica E4 wearable device which captured different measurements like – blood volume pulse, electrodermal activity, body temperature, interbeat-interval, heart rate, 3-axis accelerometer data (Ç. Çöpürkaya et al.,2023).

Sara Campanella et al. (2023) proposed a machine learning approach to differentiate stressful and non-stressful situations based on photoplethysmography and electrodermal activity signals collected from 29 individuals via Empatica E4 bracelet. The subjects went through an experiment inspired by Montreal Image Stress task which aids the study of psycho-social stress in human brain. E4 bracelet provides temperature, accelerometer, EDA and PPG sensor data. PPG and EDA signals were manually segmented and filtered for feature extraction. Using statistical methods like mean, median, mode, standard deviation meaningful features were extracted from the input sensor signals. Random forest, SVM and Logistic regression machine learning algorithms were used here to classify stress. With Random Foret the model achieved 76.5% accuracy using all the features, but with Pearson’s Correlation coefficient it achieved 75.4% accuracy. SVM model achieved 74.5% accuracy and logistic regression achieved 76.4% accuracy using all the features (Sara Campanella et al., 2023).

Iqbal, T. et al. (2022) performed a study on physiological signals from wrist-worn watches with a photoplethysmogram (PPG) sensor. A study was conducted on 35 healthy volunteers who answered a few questions which were designed to introduce stress levels and then they went through tasks including the Stroop color test, Trier Social stress test, and Hyperventilation Provocation test. They also had a rest period in between two tasks. This study used two statistical analyses -1) Linear mixed model analysis and 2) Adaptive reference range analysis to determine the predictive power of the features collected through wearables. Analysis was performed on Heat Rate (HR) and Respiratory Rate (RR) parameters. Both the parameters result in small p values (<0.001) with 95% confidence. Both are statistically significant during stress period (Iqbal, T. et al., 2022).

Md. Rafiul Amin et al. (2022) performed a study on undergraduate students and predicted examination performance based on Empatica E4 records. The study was based on Yerkes-Dodson law, which states that when stress level is too high or low, an individual’s task at hand is negatively impacted. So, a student might perform poorly when under high stress. They predicted students' grades by building classification model and enhanced the performance using k=10-fold cross validation technique. k-Nearest Neighbors (kNN) model achieved 80% with 5-min data duration window. Also, Ensemble bagged tree model achieved 80% accuracy with 15 min data duration window. However, Ensemble subspace kNN achieved 60% accuracy. With SVM (Support Vector Machine), 70% accuracy was achieved for 30 min data duration window (Md. Rafiul Amin et al., 2022).

Anu Priya et al. (2020) predicted anxiety, depression and stress based on survey data on DASS-21, the Depression, Anxiety and Stress Scale questionnaire. The modeling approach classified anxiety, stress, and depression into five different severity levels. Decision tree, random forest, Support vector machine, Naïve Bayes, and k-Nearest Neighbor (kNN) algorithms were implemented for prediction. Naïve Bayes classifier achieved highest accuracy of 85% to classify Depression level along with 74% accuracy in classifying Stress. However, the decision tree algorithm performed poorly with 62% accuracy to predict stress level (Anu Priya et al., 2020).

P. Bobade et al. (2020) worked on WESAD dataset – multimodal data collected from wearable sensory devices. They build classification models to predict stress and no-stress state along with predicted three different physiological conditions- amusement -neutral-stress state. With different input features like ACC, ECG, BVP, TEMP they performed feature extraction techniques which involved computing mean, standard deviation, maximum and minimum of each feature. Also, to build both types of classification models, they performed Principal Component Analysis (PCA) and standardized the features. Decision Tree, Random Forest, K-Nearest Neighbor, Linear Discriminant Analysis, AdaBoost and Kernel Support Vector Machine Classification machine learning algorithms were implemented along with deep learning

– Artificial Neural Network (ANN) to perform binary and three class classification. For Binary classification, ANN achieved 95% accuracy whereas Decision Tree performed 68% accuracy for three-class classification (P. Bobade et al., 2020).

Research work surveyed here on stress prediction process, mostly used manually surveyed data and in few cases sensory wearable data is used to build machine learning models. Both traditional machine learning and artificial neural network techniques are used to build classification models to predict stress. Support Vector Machine, kNN, Naïve Bayes traditional algorithms achieved high accuracy in few papers whereas ANN achieved highest accuracy in the work published by P. Bobade et al. (2020). Sara Campanella et al. (2023) work achieved similar accuracy with SVM, logistic regression and random forest. So, both traditional and deep learning can achieve high accuracy depending on the dataset used and feature engineering process.

Based on the Technology and Research Survey, the current project will focus on implementing both traditional machine learning and deep learning algorithms. This project aims to explore a more generalized way to classify stress based on sensory wearable data collected from different sources with the highest accuracy level. To achieve highest accuracy, feature engineering processes like PCA, data standardization will be implemented.

Literature Survey of Existing Research

Stress can be induced by several external factors such as work pressure, personal relationships, and material desires. Gradually it deteriorates an individual’s physical and mental well-being.

The research aims at studying effects of stress on professional and academic performance. It collects physiological data through survey questionnaires such as blood volume pulse (BVP), inter-beat intervals (IBI), electrodermal activity (EDA) and heart rate (HR) through wearable sensors. This will enable us to study the relationship between stress and physiological signals. Data preprocessing steps were implemented based on signal labels and frequency. It will be categorized as data collection, preprocessing and data exploratory analysis through survey questionnaire, visual representations of data such as line and bar charts. The questionnaire included personal information, educational qualifications, anxiety issues and smoking habits. This research contributed towards the creation of an exhaustive and complete dataset which can be further utilized in future studies for analysis, prediction, and classification of stress levels.

17.24% of men have high test anxiety whereas 30% of women appear to have high test anxiety. Survey participants are related to child development, law, tourism, business administration, biomedical engineering, and public finance. 60.42% of students plan to pursue higher master's studies. Based on stress levels, strategies can be formulated to reduce stress levels. This will help in understanding stress stimulating factors (Copurkaya et al. 2023).

Stress monitoring is done through E4 bracelet. It avoids any kind of health-related complications induced by stress. Heart rate variability (HRV), galvanic skin response (GSR), blood oxygen saturation, cortisol level, blood pressure (BP), and brain signals are primary features under consideration. This is employing machine learning algorithms such as K nearest neighbors KNN, Random Forest and logistic regression for analyzing physiological signals obtained through the bracelet. Feature importance is performed through Pearson coefficient and chi test in random forest algorithm. Implementation of machine learning techniques and statistical tools for personalized stress monitoring was carried out. This literature uses binary classification of stress levels as stress exists or not. The ranking of feature importance was carried out using two methods: chi-square tests (chi-test), in MATLAB, and the Pearson's correlation with the Waikato Environment for Knowledge Analysis (WEKA). All the findings were consistent with the previously available work. Precision, Recall and F1 metric were used to determine machine learning algorithms effectiveness. The classifiers produced an accuracy of approximately 70 percent in all cases. Tuning the hyperparameters has a positive impact on performance. The random forest provided highest accuracy as it generalizes well. This is due to its random nature. The accuracy, along with various other metrics of Logistic Regression, was not satisfactory, particularly for label 0, where precision and recall values were 0.68 and 0.53, respectively. This literature provided enough evidence towards using random forest as a robust algorithm to predict with high accuracy. Despite the limited over the rest of baseline condition, which were responsible for the balancing the database, an accuracy of 76.5%, 75.4%, and 75.7% was achieved, using all the features and both Pearson and chi-test approaches, respectively (Campanella et al. 2022).

Stress level data is collected from the smartwatch. This literature compares the Individual, centralized and federated learning methods of machine learning techniques. For the analysis purposes, data from smart watch must be sent to a centralized server, but in these cases, user privacy can be compromised. This data might contain sensitive information which is not supposed to be shared. In these cases, federated learning can be utilized, and machine learning models can be trained on data without leaving the user's device. Stress levels are classified through the application of logistic regression. Individual, centralized, and federated learning strategies are used on logistic regression. Logistic regression is chosen due to its low complexity and good performance. The research utilized Accuracy, Precision, Recall and F1 score as performance metrics to evaluate individual, centralized, and federated learning strategies.

Federated learning performed poorest on stress level detection, followed by centralized. Individual learning emerged as the best performing model. The individual learning model proved to be best performing with an average accuracy of 0.9998 and an average F1-measure of 0.9996. Individual machine learning models of all the participants proved to be 100% accurate and F1- measure. Even the poorest individual model provided an accuracy of 0.9970 and F1-measure of 0.9951, which can be considered almost perfect (Fauzi et al.2022). Table 1 depicts the comparison among the relevant literatures cited in the references.

Table 1

Comparison among relevant research papers

Relevant Literature	Objective	Methods Employed	Outcome
Copurkaya et al. 2023	Data Collection and Analysis	Statistical tools such as variance of signals and frequency distribution	A detailed and exhaustive dataset for future studies. 17.24% of men have high test anxiety whereas 30% of women appear to have high test anxiety
Campanella et al. 2022	Stress Level Classification	SVM, logistic Regression and Random Forest	Random Forest Model provided highest accuracy. An accuracy of 76.5%, 75.4%, and 75.7% was achieved, using all the features and both Pearson and chi-test approaches, respectively.

Fauzi et al.2022	Compare Individual, Centralized and federated learning	LR in combination with 3 learning approaches	The individual learning model proved to be best performing with an average accuracy of 0.9998 and an average F1-measure of 0.9996

Data and Project Management Plan

Data Management Plan

Availability and processing data for building a machine learning model is utmost crucial for this project. Data Collection is the most challenging part of any machine learning project. Stress data collection from the subjects comes from limited closed group study with consents. Also, only a few sources are available where data was collected using sensory wearable devices.

Data Collection Approaches

This project focuses on data collected from sensory wearable devices only. The project will be using a biophysiological dataset, named the “Stress-Predict Dataset”. Iqbal, T et al. (2022) conducted a pilot study where they collected physiological signals from 35 healthy volunteers using wrist-worn watches with a photoplethysmogram (PPG) sensor. These subjects went through a series of stress tests (i.e., Stroop color test, Trier Social Stress Test and Hyperventilation Provocation Test) along with intermittent rest time. This dataset captures accelerometer (x, y, z axes), raw BVP, skin conductance EDA, heart rate, inter-beat-interval, skin temperature data and timestamp tags for each task start and end. Also, the data set has a label indicating presence or absence of stress for each timestamp for each participant. The dataset is shared by Iqbal. T (2022) in his GitHub repository in comma-separated values (CSV) format for each feature for each volunteer. The size of the entire data is approximately 70MB.

This project also will be using BioStress dataset collected through a survey conducted by KOCAÇINAR, B et al. (2023). 48 volunteers took part in this 15-minutes long stress inducing session in the presence of a psychologist expert and data was collected from Empatica E4 device worn by these volunteers. The collected data is in CSV format and the entire size of the dataset is around 81MB. Unlike “Stress-Predict Dataset”, this dataset is ready for Machine learning model as it has combined all the physiological signals along with timestamp and stress labels in upstream and downstream datasets. (KOCAÇINAR, B et al. 2023).

Along with the above mentioned two data sources this project will also include Empatica E4 Stress data collected by Campanella, S. et al. (2023). This study collected photoplethysmographic and electrodermal activity signals from 29 subjects from their Empatica E4 device. The size of this dataset is around 54MB. Also, the data is in CSV format and is not labelled. For each subject all the sensory signals were collected in separate CSV file formats.

Data Management, Storage Methods, and Usage Mechanism

In this project Amazon web services (AWS) will be used for data storage and management. The project is using multiple data sources, and all CSV files will be stored in Amazon S3 bucket. Also, the project is aiming to build Extraction-Transformation-Load (ETL) pipeline using Amazon Redshift. Amazon Redshift is a fully managed scalable data warehouse service and integrated with storage bucket S3. Though this project is initially considering around 200MB of data from multiple sources, using Amazon Redshift there will be a future scope to scale up this project. Transformed and labelled dataset will be merged at Redshift and will create central data repository for this project.

Data Access Management (IAM roles) will be implemented for data access in Amazon services. This will ensure data security throughout the project tenure. Also, data recovery will be managed by Redshift.

Data Processing Requirement

Data collected from multiple sources for this project is partially labelled. We will label the data based on the timestamp and interval during which stress was induced as recorded in the study. Also, raw data needs to be cleaned and merged for each timestamp for each participant.

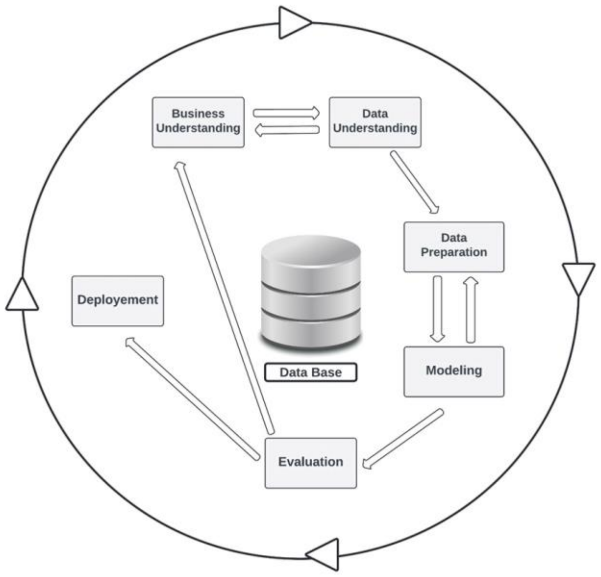
We will be focusing on different signal-based data labelling and the project aims to build a generalized machine learning model which will be able to predict presence or absence of stress directly from signals captured through sensory device. We are planning to perform feature engineering including mean, median, mode, standard deviation, maximum and minimum calculation based on moving time window. Also, Principal component analysis (PCA) will be performed on the data.

Project Development Methodology

There are many approaches that can be used for this project - The waterfall model, The agile Methodology, and the CRISP-DM model. Waterfall Model is not the best suited because in this each step must be completed to start the next step, But the project requires lot of experimentation and revisiting which cannot be done with this methodology. The project requires a balanced approach that integrates flexibility with rigorous testing and compliance measures. CRISP-DM is best suited for this project because it provides a structured framework for data analysis and model development as shown in Figure 2.

Figure 2

Represents CRISP-DM diagram for project development.



Given the complexity of analyzing data from IoT wearable devices and the iterative nature of building accurate stress detection models, CRISP-DM offers clear guidelines for understanding data, preprocessing, feature engineering, and model evaluation. Its analytical and clear technique guarantees the accuracy of stress detection, and its focus on collaboration and stakeholder participation makes it possible to incorporate feedback and domain knowledge all the way through the project. Additionally, the iterative nature of CRISP-DM works well with the ongoing development and enhancement required to meet the project's objective of early stress detection and significant health issue avoidance.

Business Understanding

The first phase in CRISP-DM methodology is Business Understanding phase for this project involves recognizing the significance of stress detection in today's fast-paced world and the importance of early intervention to prevent adverse health effects like anxiety and depression. Like the behavior analysis in various industries, the project aims to utilize the advancements in IoT and wearable technology to monitor stress levels continuously. Understanding various causes of stress, including expectations from society, professional responsibilities, and interpersonal connections, highlights the importance of precise and rapid data gathering, cleansing, and processing. The project specifically focuses on recognizing stress indicators like Heart Rate Variability (HRV) and Blood Volume Pulse (BVP) from wearable devices to classify stress levels accurately. To account for the wide range of participant backgrounds, it is essential to balance sample representation across different demographics and ensure accurate detection of stress despite changes in individual characteristics. By addressing these requirements, the project aims to contribute to the larger objective of promoting mental wellness through early stress detection and intervention.

Data Understanding

The second phase in the CRISP-DM methodology is understanding the data, which follows the first phase of Business Understanding. In this project, data collection primarily revolves around sensory wearable devices, with multiple datasets contributing to the analysis. The primary dataset, termed the "Stress-Predict Dataset," originates from a pilot study conducted by Iqbal, T et al. (2022). This dataset encompasses physiological signals gathered from 35 healthy volunteers using wrist-worn watches equipped with a photoplethysmogram (PPG) sensor. Participants underwent various stress tests, including the Stroop color test, Trier Social Stress Test, and Hyperventilation Provocation Test, alongside rest intervals. The dataset includes accelerometer readings (x, y, z axes), raw blood volume pressure (BVP), electrodermal activity (EDA), heart rate, inter-beat interval (IBI), skin temperature data, and timestamps for task initiation and conclusion. Stress labels are provided for each timestamp and participant. Similarly, the BioStress dataset, acquired from a survey by KOCAÇINAR, B et al. (2023), includes data from 48 volunteers during a 15-minute stress-inducing session using Empatica E4 wearable devices. Unlike the "Stress-Predict Dataset," the BioStress dataset is ready for machine learning models, combining physiological signals with timestamps and stress labels. Lastly, the project incorporates the Empatica E4 Stress data by Campanella, S. et al. (2023), featuring photoplethysmographic and electrodermal activity signals from 29 subjects. These datasets provide a comprehensive basis for exploring the relationship between physiological signals and stress levels, essential for stress detection and management analysis.

Data Preparation

In the data preparation phase of our project, we will address several key tasks to ensure that the collected data is well-prepared for subsequent modeling and analysis. Since the data is collected from multiple sources and is only partially labeled, our first step will be to label the data based on the timestamp and interval during which stress was induced, as recorded in study.

Additionally, we will perform data cleaning and merging for each timestamp and participant to ensure consistency and accuracy in the dataset. Our focus will be on signal-based data labeling, as we aim to build a generalized machine learning model capable of predicting the presence or absence of stress directly from signals captured through sensory devices. To enhance the predictive power of our model, we plan to perform feature engineering, which will involve calculating various statistical measures such as mean, median, mode, standard deviation, maximum, and minimum values based on moving time windows. These features will provide valuable insights into the underlying patterns and trends present in the data. Furthermore, we intend to employ Principal Component Analysis (PCA) to reduce the dimensionality of the data and extract the most informative features. By performing these data preparation steps, we will ensure that our dataset is well-structured, labeled, and enriched with meaningful features, setting the stage for the successful construction and evaluation of our stress detection model. The preprocessed data is split into training and testing sets in an 80:20 ratio while maintaining class balance in order to guarantee unbiased model performance. The datasets are prepared for the next stages of model construction and assessment by carefully recording these processes, which paves the way for perceptive stress detection analysis.

Modeling

In the modeling phase, machine learning models will be built using a range of Python tools, like Scikit-learn, and torch. We use a variety of algorithms, each with a unique set of benefits for stress detection, including Random Forest (RF), XG Boost, K nearest neighbors (KNN), Logistic Regression (LR), and Deep Neural Network (DNN). Random Forest performs best with small datasets and high accuracy, Random forest works best with bigger datasets avoids overfitting, DNN works well on non-linearity and capture the complex pattern of features, and XG boost guarantees consistent results. Standard metrics including F1-Score, Accuracy, Precision, and Recall will be used in performance evaluation, and testing datasets will be used to evaluate model outputs. The results' interpretation will direct the selection and improvement of the model for subsequent iterations. We will also investigate ensemble techniques like voting, boosting, averaging, and bagging to build ensemble models that use the advantages of individual algorithms. Our goal is to conduct a thorough review to determine the best model for stress detection using IoT devices.

Evaluation

During the assessment stage, the models' performance is evaluated based on predefined metrics focusing on accuracy, F1-score, and confusion matrix analysis. By enabling early stress identification and the prevention of long-term health difficulties, the goal is to forecast stress levels with about 80% accuracy. This will have a positive influence on people's well-being. We will choose the best model for deployment by comparing the models according to these assessment measures. We will assess measures like F1 score, Accuracy, Specificity, and Sensitivity to determine model performance for classification algorithms like ours. Furthermore, an assessment report is generated that contrasts our machine learning models' performance with that of deep learning models, guaranteeing that the model selected for testing and implementation exhibits the highest levels of validation and training accuracy. The careful evaluation of models is intended to determine the most reliable and accurate stress detection method, contributing to the improvement of people's mental health.

Deployment

The last step in CRISP-DM methodology is the deployment phase which involves utilizing AWS services such as S3 for data storage and Amazon Redshift for data warehousing. All CSV files will be stored in an Amazon S3 bucket, utilizing AWS's scalable and secure storage solution. Additionally, an Extraction-Transformation-Load (ETL) pipeline will be constructed using Amazon Redshift, a fully managed data warehousing service integrated with

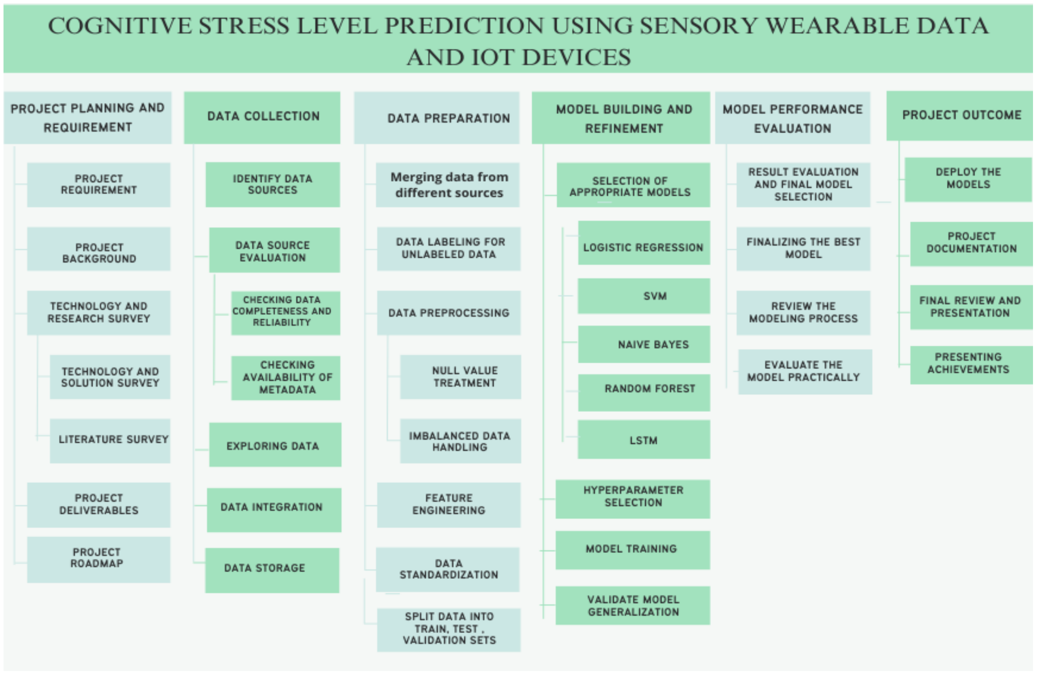
S3 storage. Despite handling about 200MB of data from multiple sources, Amazon Redshift offers scalability for future expansion of the project. The transformed and labeled dataset will be consolidated within Redshift, establishing a central data repository. To ensure data security throughout the project's lifecycle, Data Access Management will be implemented for data access in Amazon services, while Redshift manages data recovery. The deployment process will be thoroughly documented to ensure transparency and understanding, facilitating the regular monitoring, re-training, and analysis of deployed models to maintain performance and incorporate user feedback for continual improvement.

Project Organization Plan

To achieve the objective of a convincing stress monitoring system, The Work Breakdown Structure (WBS) was created to provide a blueprint of hierarchical decomposition of the project. It consists of six phases: Project Planning, Data Collection, Data Preprocessing, Model Building, Final Evaluation, and Project Outcome. The structure details are shown in Figure 3:

Figure 3

WBS Chart for Stress Prediction Project


COGNITIVE STRESS LEVEL PREDICTION USING SENSORY WEARABLE DATA AND IOT DEVICES
PROJECT PLANNING AND REQUIREMENT
PROJECT REQUIREMENT
PROJECT BACKGROUND
TECHNOLOGY AND RESEARCH SURVEY
TECHNOLOGY AND SOLUTION SURVEY
LITERATURE SURVEY
PROJECT DELIVERABLES
PROJECT ROADMAP
DATA COLLECTION
IDENTIFY DATA SOURCES
DATA SOURCE EVALUATION
CHECKING DATA COMPLETENESS AND RELIABILITY
CHECKING AVAILABILITY OF METADATA
EXPLORING DATA
DATA INTEGRATION
DATA STORAGE
DATA PREPARATION
Merging data from different sources
DATA LABELING FOR UNLABELED DATA
DATA PREPROCESSING
NULL VALUE TREATMENT
IMBALANCED DATA HANDLING
FEATURE ENGINEERING
DATA STANDARDIZATION
SPLIT DATA INTO TRAIN, TEST , VALIDATION SETS
MODEL BUILDING AND REFINEMENT
SELECTION OF APPROPRIATE MODELS
LOGISTIC REGRESSION
SVM
NAIVE BAYES
RANDOM FOREST
LSTM
HYPERPARAMETER SELECTION
MODEL TRAINING
VALIDATE MODEL GENERALIZATION
MODEL PERFORMANCE EVALUATION
RESULT EVALUATION AND FINAL MODEL SELECTION
FINALIZING THE BEST MODEL
REVIEW THE MODELING PROCESS
EVALUATE THE MODEL PRACTICALLY
PROJECT OUTCOME
DEPLOY THE MODELS
PROJECT DOCUMENTATION
FINAL REVIEW AND PRESENTATION
PRESENTING ACHIEVEMENTS

Project Planning

The project planning aims to review the literature, define the scope of the project, identify the risk, and develop a reasonable schedule. This phase is separated into tasks given as Related literature review, Comparative analysis between individual, centralized, and federated learning-based stress detection literature review, stress detection with machine learning and deep learning literature review, Stress detection using Empatica E4 Bracelet and machine-learning techniques literature review, define project objectives and deliverables, Risk management and establishes milestones and a flexible timeline.

Data Collection

Data Collection involves the process of gathering and examining data from various sources. The phase two includes five tasks as identify data sources, data source evaluation, check data completeness and reliability, check the availability of necessary metadata, explore the data, data integration and data storage.

Data Preprocessing

The Data Preprocessing phase is the most crucial part, which encompasses activities related to the preparation of raw data for analysis and model construction. The structure of phase three includes manual data labeling for unlabeled data, handling missing values and outliers, feature standardization, data validation, data split prepared for model building and documentation of data engineering.

Model Building, Refinement and Evaluation

Phase four will build several models, including Logistic Regression, K nearest neighbors, XG Boost, Random Forest, and DNN, based on the preprocessed data, and tune them to ensure each individual model achieves the best performance. This phase is decomposed into five tasks as define model architecture, hyperparameter selection for each model, model training, validate model generalization, cross validation and model tuning, model performance evaluation.

Project Outcome

The project outcome mainly shows the achievement result and accomplishes the final documentation of this project.

- presentation preparation
- present achievement to stakeholders
- final project documentation

Project Resource Requirements and Plan

Amazon Web Services (AWS) are utilized for implementing this project. AWS is a cloud-based platform which offers services for data storage, machine learning model development, databases, visualizations, and service management. These applications offer scalability, high availability and pay as you use pricing estimates. Jupyter Notebook is used for writing Python scripts which are carrying out data preprocessing and exploratory data analysis tasks. Python version 3.7 was used for statistical analysis.

A local machine with 64-bit processor and 16 GB RAM is utilized for AWS services to work efficiently without any interruptions. AWS QuickSight empowers organizations to include business intelligence capabilities in their decision-making process. Interactive dashboards and visualizations are created with AWS QuickSight. AWS S3 is used to store data files and other analysis files related to the project. Machine learning algorithms are used to build, train and deploy the models through AWS SageMaker. AWS SageMaker offers a cost effective and high-performing platform to preprocess data for modeling, selecting the algorithms, comparing the performance of the models, and creating machine learning pipelines. Machine learning libraries such as scikit learn, NumPy and pandas are integrated with AWS SageMaker to perform machine learning modeling. AWS CLI is a one stop solution for all the services in the AWS. With the single AWS Command Line Interface, we can manage all the services within AWS.

Zoom and Microsoft Office 365 were used for collaboration among team members to share the project's progress and prepare deliverables. Through zoom, team members have regular weekly check-ins to stay updated with each other's progress to make timely deliverables.

Table 2 shows the hardware and software requirements necessary to complete the project and achieve its objectives. Cost Estimates have been provided to depict the cost incurred towards the project.

Table 2

Hardware and Software Resources and their cost estimation

Resource Name	Resource categorization	Total duration (in months)	Cost Estimate	Usage
AWS S3	Hardware	2 months	25 USD	Data Storage
AWS SageMaker	Software	2 months	120 USD	Machine Learning Model Development
Windows 64-bit version machine	Hardware	2 months	900 USD	Local Device
Jupyter Notebook (Python)	Software	2 months	Free of Cost	Data Preprocessing and Exploratory Data Analysis

Microsoft Office 365	Software	2 months	Free of Cost (Student Subscription)	Report Writing, Presentation and Statistical Analysis
Zoom	Software	2 months	Free of Cost (Student Subscription)	Collaboration among team
AWS Quicksight	Software	2 months	30 USD	Data Visualization
Scikit Learn, NumPy, Pandas	Software	2 months	Free of Cost	Libraries for model development
MySQL	Software	2 months	Free of Cost	Database
AWS CLI	Software	2 months	Free of Cost	AWS Service Management tool

Project Schedule

PERT Chart

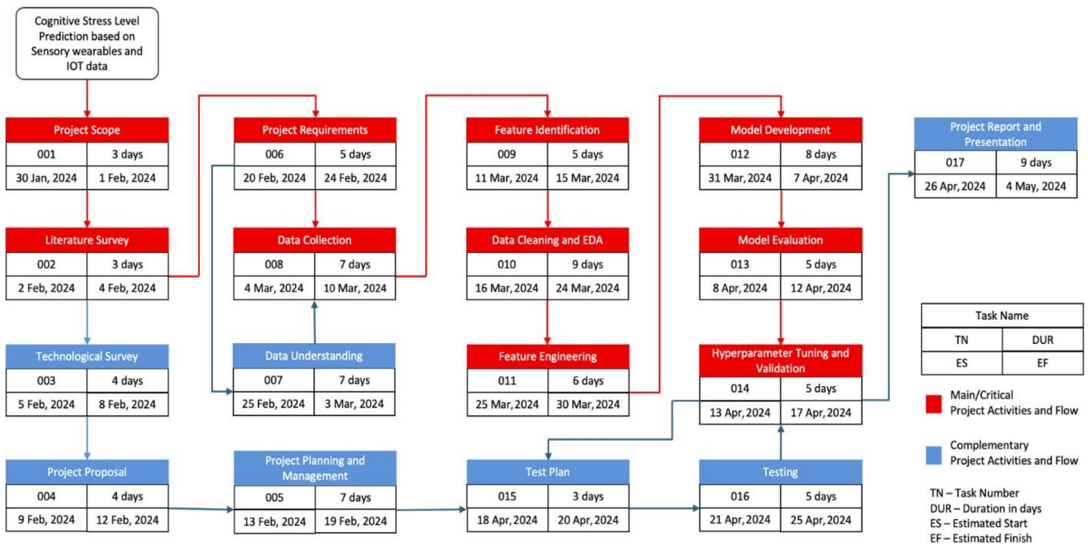
Among various visual project management tools, which are used to track project tasks and timelines, PERT chart (program evaluation and review technique) is one such tool. It is a methodology of project duration by building a network. It enables the scheduling, organization, and coordination of tasks within a project. Figure 4 shows the PERT chart for cognitive stress level prediction based on Sensory wearables and IOT data. Here the nodes represent tasks for the project, and arrows represent the flow of the project. Inside the nodes we mentioned task number, duration of days which tells amount of time the team will have to complete each task, estimated start, estimated finish. Here in the PERT chart the critical project activities are mentioned in red color and complementary project activities are mentioned in blue color.

To create a PERT chart, it's considered good practice to follow the following steps, first we need to identify all the project's activities like major phases, milestones/tasks, then identify dependencies to gauge the project sequence and the draw the PERT chart using these milestones/tasks as the numbered nodes and then add in directional arrows accordingly, the final task is to establish timelines for when tasks need to be completed.

Visual representation of PERT Chart

Figure 4

PERT Chart for Cognitive Stress Level Prediction based on Sensory wearables and IOT data.



Gantt chart

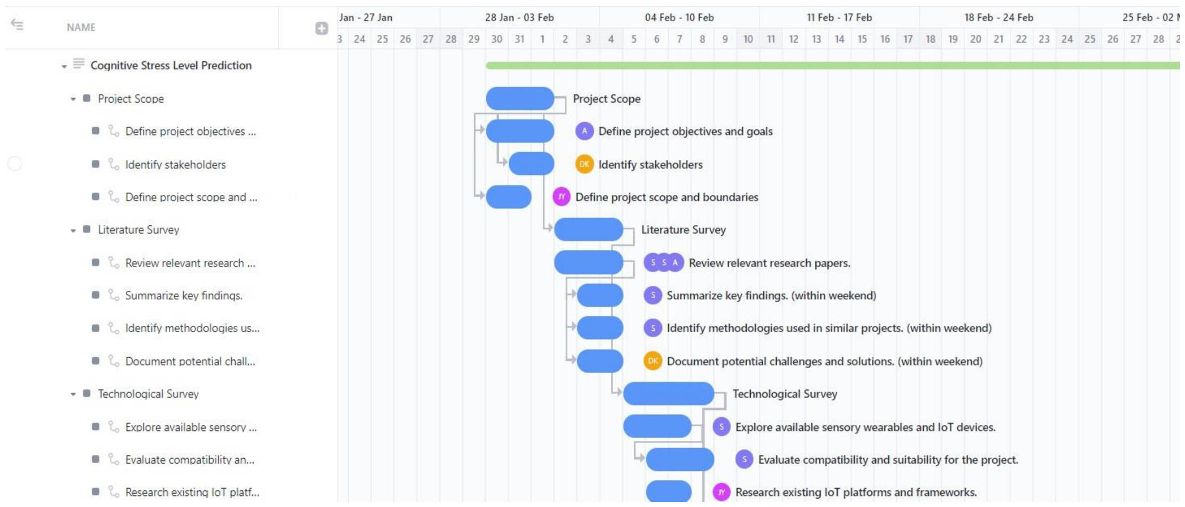
Gantt chart is produced during the planning phase of project management process. It is an agile project management tool that showcases the various project activities and their inter-dependencies along with the time allocated for executing them and the respective actual completion time. The schedule contains high level, low-level milestones, and activities within the project from start to finish. The start and end dates of tasks, milestones, dependencies between tasks, assignees, in how many days a task can be completed etc. are included in the Gantt chart. The following figures help to denote visual representation of Gantt chart which shows all the tasks and deadlines and how they are segregated among the team members equally to share the word load. This ensures that the workload is distributed equally and aim to finish the tasks according to the deadlines assigned to the respective team members.

In Gantt chart we can see tasks organized in a way where we can see how each one is depending on each other. The low-level milestones (sub tasks) help us ensure that no internal steps are missed and thus help in getting the best accuracy in the model. Figure 5 represents the project scope, literature survey, technological survey which helps in understanding the scope of the project. Figure 6 represents the project proposal, the project planning and management project requirements helps in following an agile methodology in implementing the project. Figure 7 represents data understanding, data collection, feature identification, data cleaning and EDA which helps in understanding the data better prior to modelling. Figure 8 represents feature engineering, model development, and model evaluation which helps in model building. Figure 9 represents hyperparameter tuning and validation, test plan, testing, project report and presentation. We used ‘clickup’ tool for building the Gantt chart.

Visual representation of Gantt Chart for scheduling the project of stress prediction

Figure 5

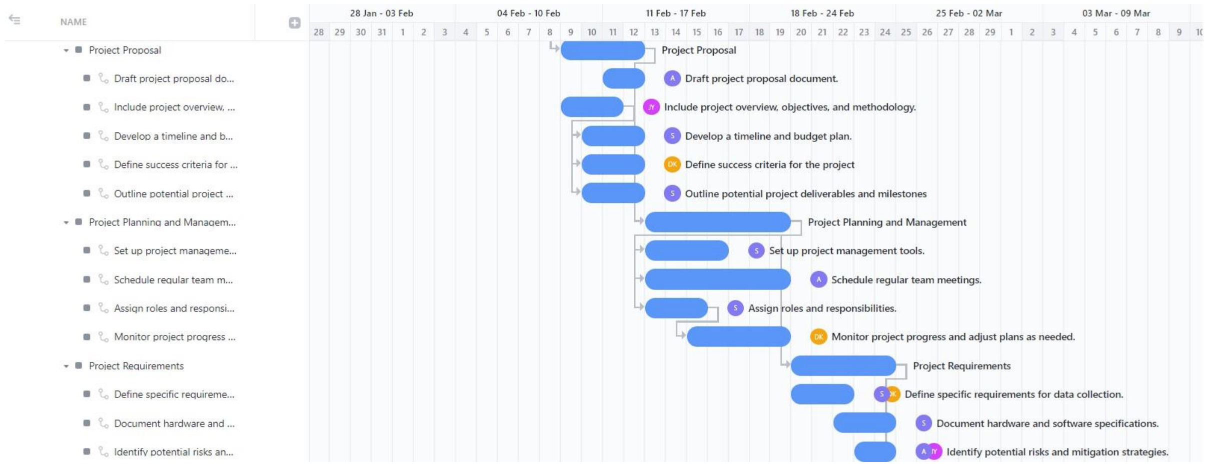
Gantt chart representing the Project scope, Literature survey, Technological survey.



Cognitive Stress Level Prediction
Project Scope
Define project objectives and goals
Identify stakeholders
Define project scope and boundaries
Literature Survey
Review relevant research papers.
Summarize key findings. (within weekend)
Identify methodologies used in similar projects. (within weekend)
Document potential challenges and solutions. (within weekend)
Technological Survey
Explore available sensory wearables and IoT devices.
Evaluate compatibility and suitability for the project.
Research existing IoT platforms and frameworks.

Figure 6

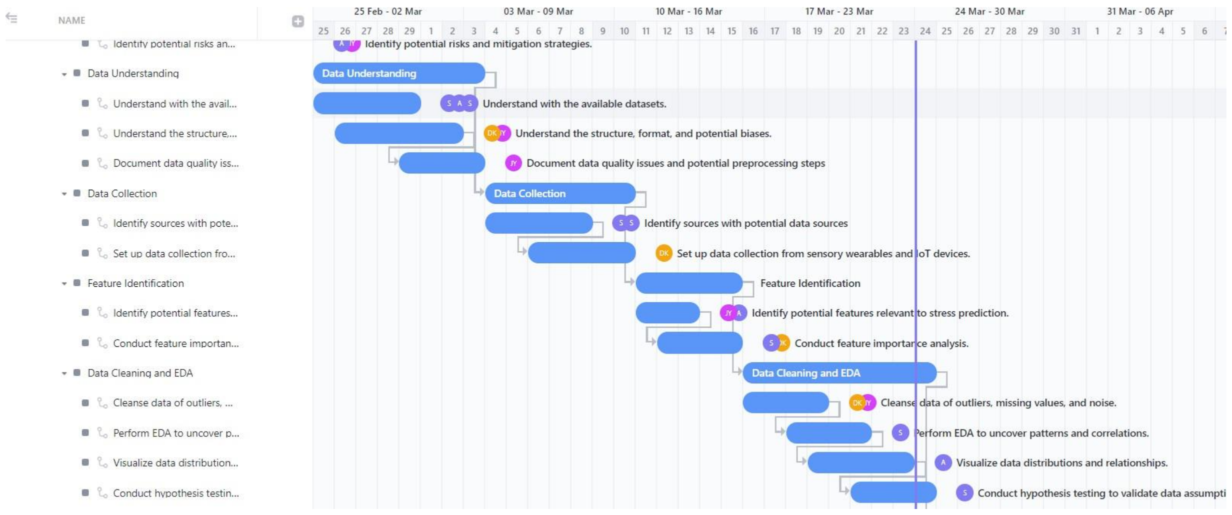
Represents the project proposal, project planning and management project requirements.



Project Proposal
Draft project proposal document.
Include project overview, objectives, and methodology.
Develop a timeline and budget plan.
Define success criteria for the project
Outline potential project deliverables and milestones
Project Planning and Management
Set up project management tools.
Schedule regular team meetings.
Assign roles and responsibilities.
Monitor project progress and adjust plans as needed.
Project Requirements
Define specific requirements for data collection.
Document hardware and software specifications.
Identify potential risks and mitigation strategies.

Figure 7

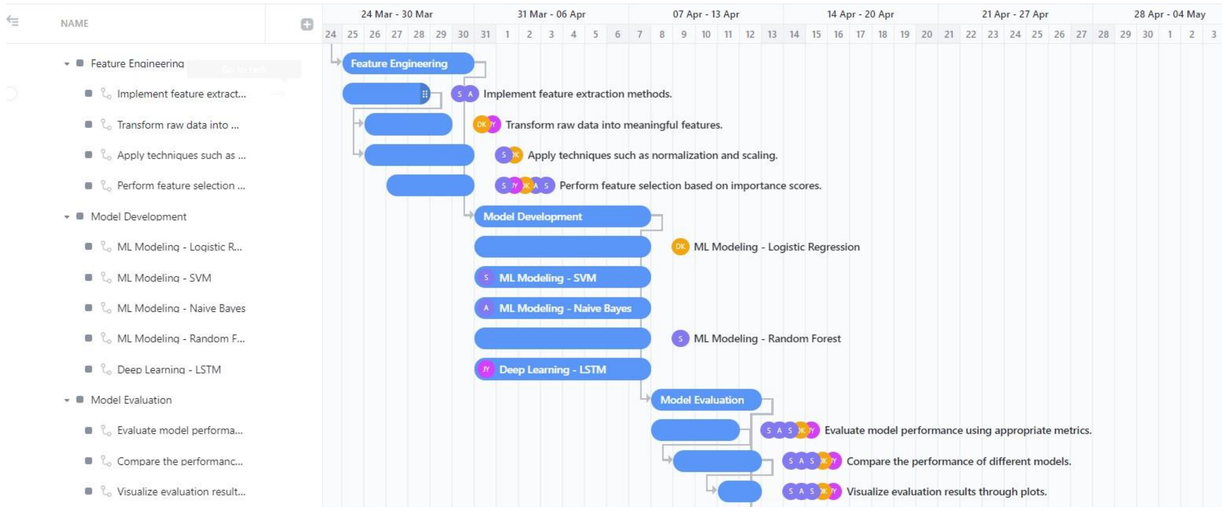
It represents Data understanding, Data collection, Feature identification, Data cleaning and EDA.



Identify potential risks an... — Identify potential risks and mitigation strategies.
Data Understanding — Data Understanding
Understand with the avail... — Understand with the available datasets.
Understand the structure.... — Understand the structure, format, and potential biases.
Document data quality iss... — Document data quality issues and potential preprocessing steps
Data Collection — Data Collection
Identify sources with pote... — Identify sources with potential data sources
Set up data collection fro... — Set up data collection from sensory wearables and IoT devices.
Feature Identification — Feature Identification
Identify potential features... — Identify potential features relevant to stress prediction.
Conduct feature importan... — Conduct feature importance analysis.
Data Cleaning and EDA — Data Cleaning and EDA
Cleanse data of outliers, ... — Cleanse data of outliers, missing values, and noise.
Perform EDA to uncover p... — Perform EDA to uncover patterns and correlations.
Visualize data distribution... — Visualize data distributions and relationships.
Conduct hypothesis testin... — Conduct hypothesis testing to validate data assumpti

Figure 8

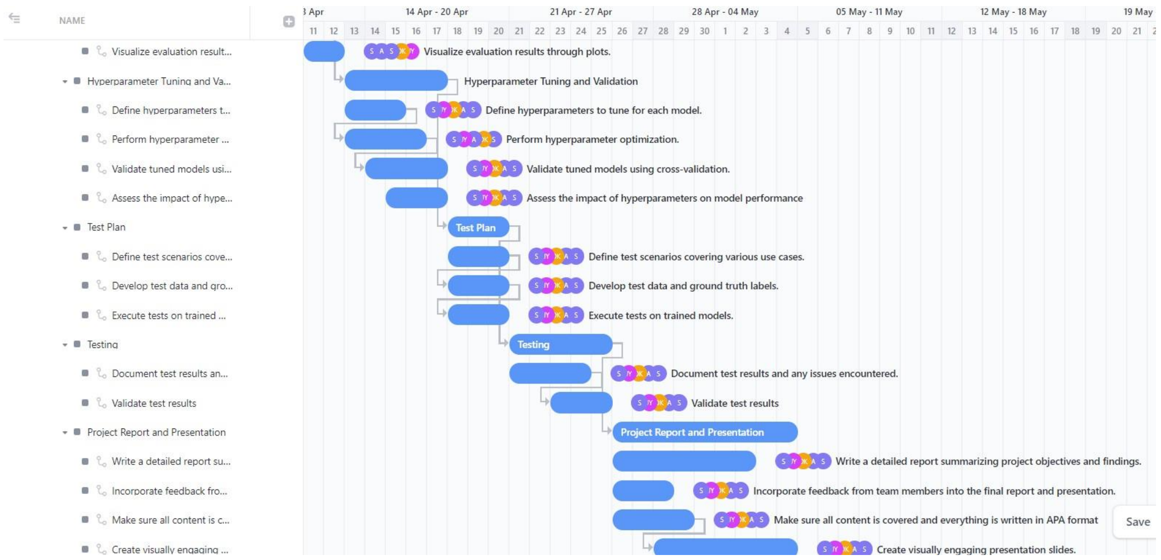
Represents Feature engineering, Model development, Model evaluation



Feature Engineering — Feature Engineering
Implement feature extract... — Implement feature extraction methods.
Transform raw data into ... — Transform raw data into meaningful features.
Apply techniques such as ... — Apply techniques such as normalization and scaling.
Perform feature selection ... — Perform feature selection based on importance scores.
Model Development — Model Development
ML Modeling - Logistic R... — ML Modeling - Logistic Regression
ML Modeling - SVM — ML Modeling - SVM
ML Modeling - Naive Bayes — ML Modeling - Naive Bayes
ML Modeling - Random F... — ML Modeling - Random Forest
Deep Learning - LSTM — Deep Learning - LSTM
Model Evaluation — Model Evaluation
Evaluate model performa... — Evaluate model performance using appropriate metrics.
Compare the performanc... — Compare the performance of different models.
Visualize evaluation result... — Visualize evaluation results through plots.

Figure 9

Represents Hyperparameter tuning and Validation, Test plan, Testing, Project Report and Presentation



Visualize evaluation result...
Hyperparameter Tuning and Va...
Define hyperparameters t...
Perform hyperparameter ...
Validate tuned models usi...
Assess the impact of hype...
Test Plan
Define test scenarios cove...
Develop test data and gro...
Execute tests on trained ...
Testing
Document test results an...
Validate test results
Project Report and Presentation
Write a detailed report su...
Incorporate feedback fro...
Make sure all content is c...
Create visually engaging ...

Visualize evaluation results through plots.
Hyperparameter Tuning and Validation
Define hyperparameters to tune for each model.
Perform hyperparameter optimization.
Validate tuned models using cross-validation.
Assess the impact of hyperparameters on model performance
Test Plan
Define test scenarios covering various use cases.
Develop test data and ground truth labels.
Execute tests on trained models.
Testing
Document test results and any issues encountered.
Validate test results
Project Report and Presentation
Write a detailed report summarizing project objectives and findings.
Incorporate feedback from team members into the final report and presentation.
Make sure all content is covered and everything is written in APA format
Create visually engaging presentation slides.

Data Engineering

Data Process

When it comes to the data acquired, several sources are used to collect data on stress level prediction such as the Mendeley has an exam of examining psychological stress from bio-stress dataset that can help get some insight and another source is a git hub. We have taken this source as a data set for different articles like Stress Monitoring Using Wearable Sensors. Stress levels have been increasing rapidly, and there is an urgent need to develop a stress monitoring system supported by wearable devices. We used supervised machine learning methods and a deep learning technique to solve this problem. Stress prediction classification that could be achieved using a physiological stress signal such as Change in Blood Volume pulse and HRV, which is collected by the wearable signals.

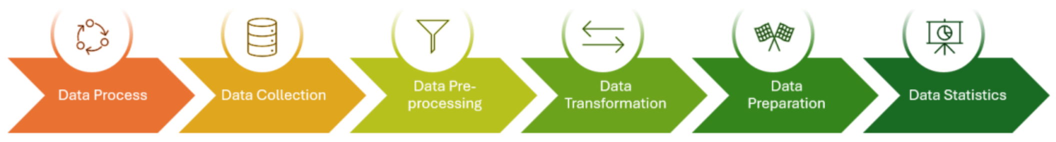
In the preprocessing phase, the focus was on cleaning the collected data in terms of quality and consistency for further modeling. We employed ADASYN techniques for data balancing. Regarding the imputation of zero values, we make sure they were not due to sensor failure, poor skin-to-sensor contact, and other technical problems. Certain data, which consists of raw formatted signals for every participant used in datasets, are combined into a single dataset on timestamp and all signals were collected at different frequencies. For instance, some of them are 32HZ, 4HZ, etc. We converted all different frequencies into 1HZ and combined them since similar activities exist between the two datasets. After doing some features engineering on x, y, z of accelerometer data applies the formula on the magnitude and implements the correlation between them to draw meaning insights out of it. Along with this preprocessing includes the treatment of missing values, imputing zero values based on the domain knowledge of the data with consideration on the contextual factors, standardization and normalization of the features and processing raw signals from the wearable device that were used to remove noise and optimize the signal quality which would in turn be used as input in our model. Through this we expect that the reliability and the accuracy of our model will be enhanced.

For optimal training and evaluation of the models, the dataset needs to be divided into the training, validation, and test sets. Eventually, the extracted features datasets are divided into training, validation and and the test set, and the train and validation is again split in 70:30. 30% is again split in to 15% for validation and 15% for test dataset. The dataset is divided in this ratio 70:15:15 for training, validation and testing. This allows the models to be trained on an acceptable dataset and validated to ascertain some hyperparameters and its generalization performance. We use a methodical approach to induce datasets hereby we hope the models will meet the design objects concerning the capability to forecast stress situations with concrete accuracy. We have mentioned the flow of data process in Figure 10.

During the final stages, the trained models are tested using the available metrics concerning the performance of algorithms such as Accuracy, precision, Recall, F1-score, confusion matrix. By evaluating the performance of each, the best algorithm is selected for stress level prediction. Then, classification and detection of stress levels take place. This project intends to help improve mental wellness among stressed individuals which can further help in creating personalized recommendations for the clients are recommended to enhance a healthier living style and reduce stress.

Figure 10

Flow of Data process



Data Collection

Stress may be defined within the context of stress classification and its various types of data sources that give a varied view into the physical and mental condition of a person. Understandably, in this regard, the understanding of these varied data sources—speech, facial expression, and IoT data, to be specific—is of prime importance for a system that seeks to develop the determination of comprehensive stress classification.

Speech data can provide useful information on the level of stress through its voice modulation, pitch, and speech patterns. However, speech data can be affected by accent, language proficiency, and context of speech.

Stress is always indicated in facial expressions like furrowing the brows, tightening the jawline, or opening the eyes wide. Analyzing facial expressions is non-obtrusive and very easy to capture on-camera or by facial recognition software. Interpretation of facial expressions, however, could be quite subjective with reference to individual differences and cultural norms.

The IoT data, and particularly the biophysiological signals from wearable devices, provide an objective and quantifiable measure of the physiological response to stress. Parameters like accelerometer data, PPG sensor readings, skin conductance, heart rate, etc., indicate direct bodily responses toward the stressors. IoT data has real-time continuity and is less likely to be tampered with compared to speaking and facial expressions. It gives a fuller and truer description of the physiological status of the person so that a much clearer classification of stress can be made.

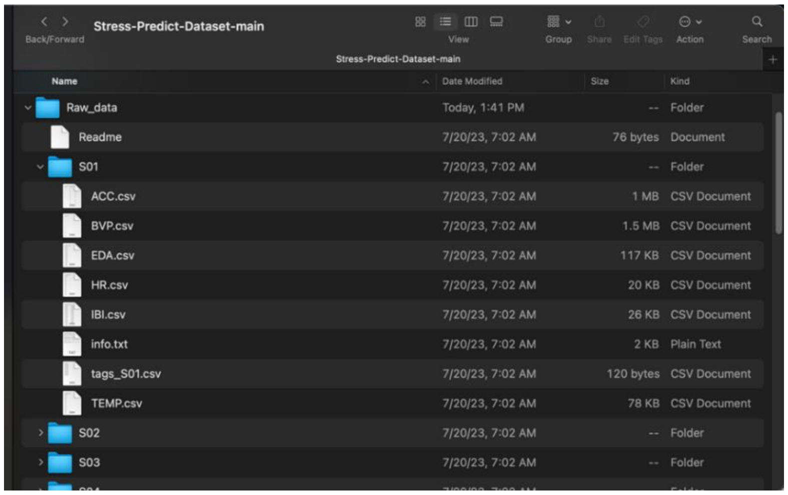
The reason we selected IoT data for classifying stress in class is that the IoT has grown up in interdisciplinary fields of science. In other words, these physiological responses—as measured directly by the IoT data provide more accurate indicators of stress than the indirect behavioral cues. Signals from IoT devices are less prone to subjective and external inflations of facts.

Real-time Monitoring: The wearable IoT devices make it possible for the dynamic and real-time monitoring of stress levels. IoT devices are non-invasive and can be worn comfortably, enabling naturalistic data collection without disruption of ordinary activities. IoT sources of healthcare have a variety of physiological signals that would certainly provide the extra depth and richness of the dataset necessary for machine learning models. Other than speech and facial expression data, among IoT data, it is the most objective, accurate, and real-time monitored, so it is the best choice in this project for stress recognition.

For the project, based on the requirements 2 different datasets have been chosen. All 2 datasets include data collected from IoT wearable devices. Figure 11 shows the dataset one, that is the 'Stress-Predict Dataset', sourced from Iqbal et al., encompasses physiological signals gathered from 35 healthy individuals via wrist-worn watches equipped with PPG sensors. These individuals went through series of stress inducing tests which include Stroop color test, Trier social test and hyperventilation provocation test along with intermittent rest time. The dataset, shared by Iqbal, T (2022), is available in comma-separated values (CSV) format on his GitHub repository. The dataset size is approximately 70MB.

Figure 11

Represents Stress-Predict Dataset storage structure in the system

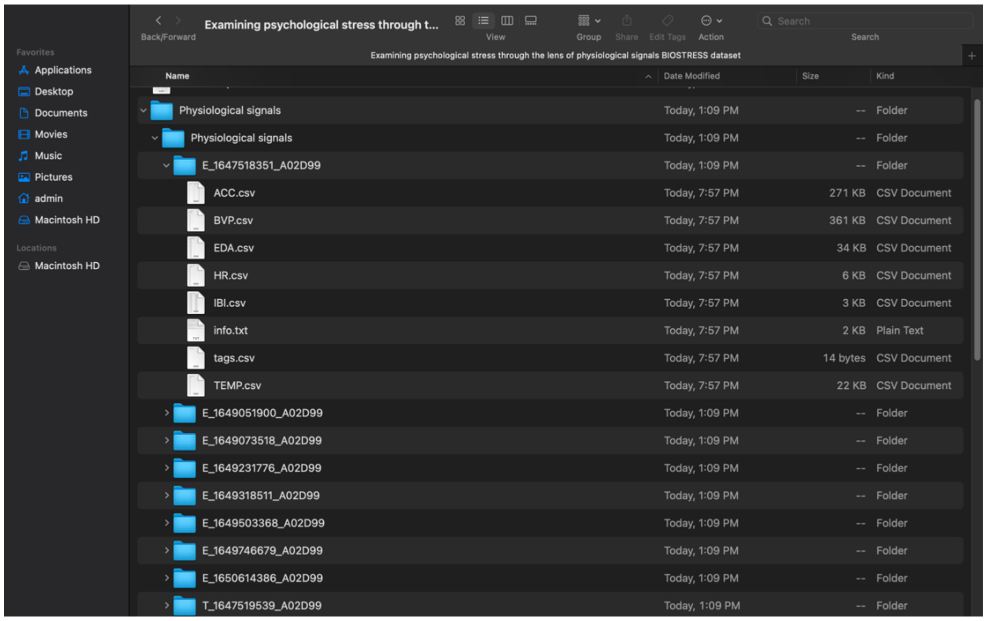


Name	Date Modified	Size	Kind
Raw_data	Today, 1:41 PM	--	Folder
Readme	7/20/23, 7:02 AM	76 bytes	Document
S01	7/20/23, 7:02 AM	--	Folder
ACC.csv	7/20/23, 7:02 AM	1 MB	CSV Document
BVP.csv	7/20/23, 7:02 AM	1.5 MB	CSV Document
EDA.csv	7/20/23, 7:02 AM	117 KB	CSV Document
HR.csv	7/20/23, 7:02 AM	20 KB	CSV Document
IBI.csv	7/20/23, 7:02 AM	26 KB	CSV Document
info.txt	7/20/23, 7:02 AM	2 KB	Plain Text
tags_S01.csv	7/20/23, 7:02 AM	120 bytes	CSV Document
TEMP.csv	7/20/23, 7:02 AM	78 KB	CSV Document
S02	7/20/23, 7:02 AM	--	Folder
S03	7/20/23, 7:02 AM	--	Folder

Figure 12 shows the second data set, that is BIOSTRESS dataset was collected through a survey conducted by KOCAÇINAR, B et al. (2023). It is like the Stress-Predict dataset since it has the same parameters of the physiological signals that the dataset contains. The dataset contains information on 48 subjects who participated in an experiment, where 15-minutes long stress was induced and the is data collected with Empatica E4 device. A series of human physiological signals are recorded using the empirical E4 device. These include Blood Volume Pressure, Electrodermal Activity, Body Temperature, Heart Rate, Inter-Beat Interval, and Accelerometer Data. The dataset from this comprises around 54MB of data and is provided in the CSV format. The source of the dataset is available at Mendeley Data. (KOCAÇINAR, B et al. 2023).

Figure 12

Represents Biostress Dataset storage structure in the system.



Examining psychological stress through t...

Examining psychological stress through the lens of physiological signals BIOSTRESS dataset

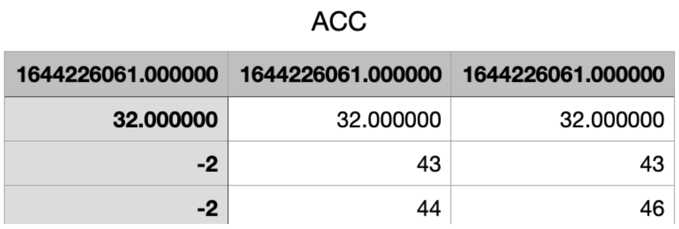
Name	Date Modified	Size	Kind
Physiological signals	Today, 1:09 PM	--	Folder
Physiological signals	Today, 1:09 PM	--	Folder
E_1647518351_A02D99	Today, 1:09 PM	--	Folder
ACC.csv	Today, 7:57 PM	271 KB	CSV Document
BVP.csv	Today, 7:57 PM	361 KB	CSV Document
EDA.csv	Today, 7:57 PM	34 KB	CSV Document
HR.csv	Today, 7:57 PM	6 KB	CSV Document
IBI.csv	Today, 7:57 PM	3 KB	CSV Document
info.txt	Today, 7:57 PM	2 KB	Plain Text
tags.csv	Today, 7:57 PM	14 bytes	CSV Document
TEMP.csv	Today, 7:57 PM	22 KB	CSV Document
E_1649051900_A02D99	Today, 1:09 PM	--	Folder
E_1649073518_A02D99	Today, 1:09 PM	--	Folder
E_1649231776_A02D99	Today, 1:09 PM	--	Folder
E_1649318511_A02D99	Today, 1:09 PM	--	Folder
E_1649503368_A02D99	Today, 1:09 PM	--	Folder
E_1649746679_A02D99	Today, 1:09 PM	--	Folder
E_1650614386_A02D99	Today, 1:09 PM	--	Folder
T_1647519539_A02D99	Today, 1:09 PM	--	Folder

The dataset is organized into individual CSV files for each physiological signal as

ACC.csv: Figure 13 and 14 shows the file contains accelerometer data, which includes measurements of acceleration along three axes X, Y and Z. This data can provide insights into body movement and activity levels.

Figure 13

Represents Sample accelerometer data from Stress predict dataset.

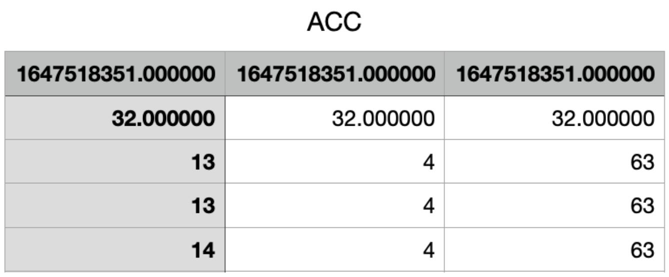


ACC

1644226061.000000	1644226061.000000	1644226061.000000
32.000000	32.000000	32.000000
-2	43	43
-2	44	46

Figure 14

Represents Sample accelerometer data from Bio stress dataset.



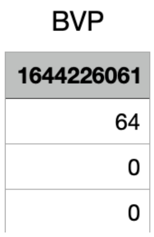
ACC

1647518351.000000	1647518351.000000	1647518351.000000
32.000000	32.000000	32.000000
13	4	63
13	4	63
14	4	63

BVP.csv: Figure 15 and 16 shows the file contains changes in blood volume in the microvascular bed of tissue.

Figure 15

Represents Sample BVP data from Stress predict dataset

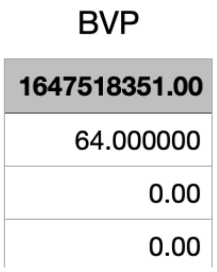


BVP

1644226061
64
0
0

Figure 16

Represents Sample BVP data from Biostress dataset

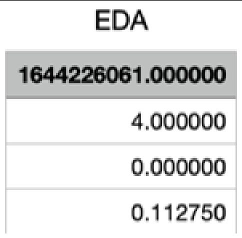


BVP
1647518351.00
64.000000
0.00
0.00

EDA.csv: Figure 17 and 18 shows file contains electrodermal activity data. EDA detects changes in the electrical conductivity of the skin, which may be an indicator of emotional and physiological arousal.

Figure 17

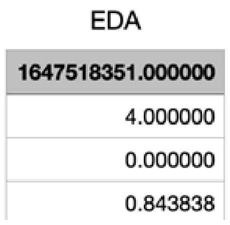
Represents Sample EDA data from stress predict dataset.



EDA
1644226061.000000
4.000000
0.000000
0.112750

Figure 18

Represents Sample EDA data from Biostress dataset.

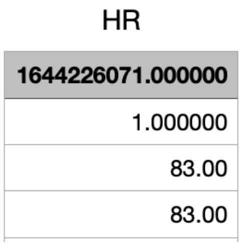


EDA
1647518351.000000
4.000000
0.000000
0.843838

HR.csv: Figure 19 and 20 shows the file contains heart rate data, it represents the number of heartbeats per unit of time. HR data is a fundamental measure of cardiovascular activity and can vary in response to stress, physical activity, and other factors.

Figure 19

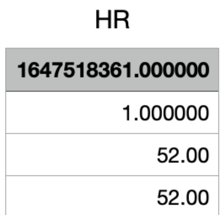
Represents Sample HR data from Stress Predict dataset


HR

1644226071.000000
1.000000
83.00
83.00

Figure 20

Represents Sample HR data from Biostress dataset.

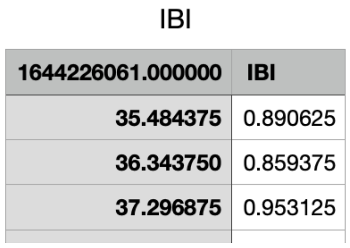

HR

1647518361.000000
1.000000
52.00
52.00

IBI.csv: Figure 21 and 22 shows the time intervals between successive heartbeats. IBI data can provide light on heart rate variability as it is obtained from HR data.

Figure 21

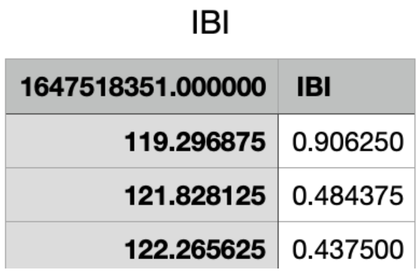
Represents Sample IBI data from Stress Predict dataset.


IBI

1644226061.000000	IBI
35.484375	0.890625
36.343750	0.859375
37.296875	0.953125

Figure 22

Represents Sample IBI data from Biostress dataset.

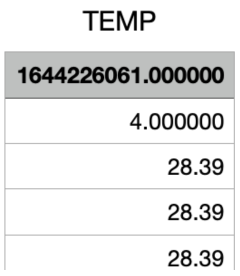

IBI

1647518351.000000	IBI
119.296875	0.906250
121.828125	0.484375
122.265625	0.437500

Temp.csv: Figure 23 and 24 shows the file contains the temperature data, which is a collection of participant body temperature readings.

Figure 23

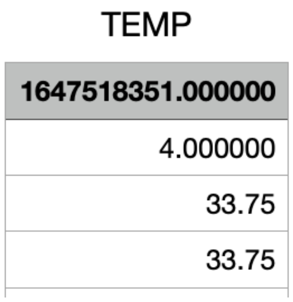
Represents Sample TEMP data from Stress Predict dataset.



TEMP
1644226061.000000
4.000000
28.39
28.39
28.39

Figure 24

Represents Sample TEMP data from Bio stress dataset.



TEMP
1647518351.000000
4.000000
33.75
33.75

It's important to remember that, to preserve consistency and enable comparison analysis, the data for each participant is gathered during the same time period and under comparable circumstances. Dataset structure is given in Table 3.

Table 3

Dataset overview and structure

Key Variables	**Stress-Predict Dataset**	**BIOSTRESSS Dataset**

Source	Iqbal, T etal.(2022)	KOCAÇINAR, B et al. 2023).
Method	Wrist-worn devices with PPG sensors	Empatica E4 device
Participants	35 volunteers	48 volunteers
Tests	Stroop, Trier Social Stress Test, Hyperventilation	TOAD stress
Data Collected	Accelerometer, BVP, EDA, HR, IBI, Skin Temp, Timestamps	Accelerometer, BVP, EDA, HR, IBI, Skin Temp, Timestamps
Labels	Not labelled	Presence or absence of stress
Format	CSV for each feature for each volunteer	CSV for each feature for each volunteer

Size	~70MB	~81MB
Availability	GitHub Repository	Mendeley Data

Data Pre-processing

We are utilizing datasets from two different sources for this project. One is biostress dataset from Mendeley data website and other one is stress predict dataset. Variations exist in both datasets in terms of number of columns, sampling frequency, outliers, presence of null values and timestamp. We are handling both files separately to bring both files in same format. In this way we can avoid anomalies in the final dataset. Each file is preprocessed individually which gives us the opportunity to dive deeper into intricacies of both dataset and deal with difference in sampling frequencies for all columns. Once initial preprocessing is done, we will merge both files and start with data statistics of combined data. This will provide information regarding the structure of the complete dataset.

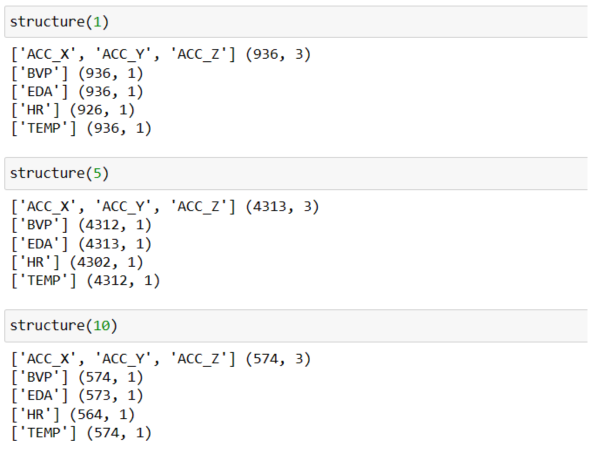
Data Preprocessing comprises of handling missing values, outlier detection, removing duplicate values and inconsistent data. These steps are extremely crucial for the performance of machine learning models. The quality of dataset directly impacts machine learning model’s performance. Exploratory Data Analysis (EDA) is carried out to understand dataset distribution, statistics, and frequencies.

Biostress dataset from Mendeley

Figure 25 shows each participant data who is undergoing stress inducing test. The physiological signals of these participants are stored in separate folders for each participant. Data belonging to all participants is combined to one file and then preprocessing is done.

Figure 25

Represents Sample participants data from Biostress dataset.

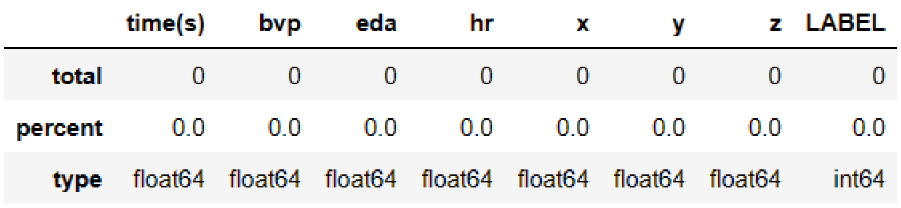


```
structure(1)

['ACC_X', 'ACC_Y', 'ACC_Z'] (936, 3)
['BVP'] (936, 1)
['EDA'] (936, 1)
['HR'] (926, 1)
['TEMP'] (936, 1)

structure(5)

['ACC_X', 'ACC_Y', 'ACC_Z'] (4313, 3)
['BVP'] (4312, 1)
['EDA'] (4313, 1)
['HR'] (4302, 1)
['TEMP'] (4312, 1)

structure(10)

['ACC_X', 'ACC_Y', 'ACC_Z'] (574, 3)
['BVP'] (574, 1)
['EDA'] (573, 1)
['HR'] (564, 1)
['TEMP'] (574, 1)
```

Figure 26 shows missing values in biostress dataset for each column time, blood volume pressure (BVP), electrodermal activity (EDA), heart rate (HR), temperature and 3-axis (x, y, z) accelerometer (ACC) and label. We can see clearly that there are no missing values in the dataset. This means all columns contain meaningful data with respect to their respective columns.

Figure 26

Shows missing values in Biostress dataset

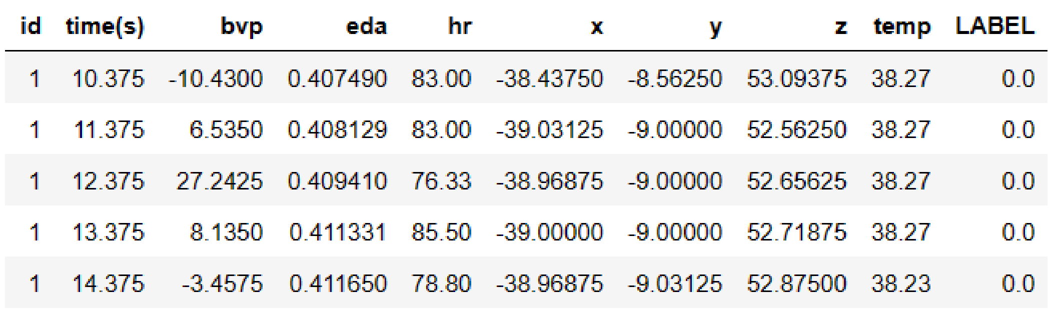


	time(s)	bvp	eda	hr	x	y	z	LABEL
total	0	0	0	0	0	0	0	0
percent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
type	float64	float64	float64	float64	float64	float64	float64	int64

Figure 27 shows the data after frequency conversion of feature variables to 1 Hz. Frequency sampling rate for columns are different. Frequency sampling rate for BVP is 64 Hz, 4Hz for EDA, 32 Hz for ACC and 4 Hz for temperature signal. For analyzing these variables together, it is vital to convert them to one frequency sampling rate which 1Hz here.

Figure 27

Shows data converted to 1 Hz frequency in Biostress dataset

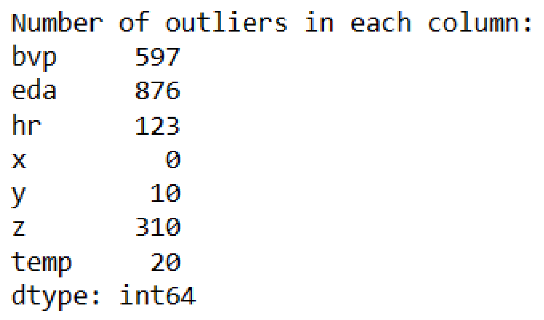


id	time(s)	bvp	eda	hr	x	y	z	temp	LABEL
1	10.375	-10.4300	0.407490	83.00	-38.43750	-8.56250	53.09375	38.27	0.0
1	11.375	6.5350	0.408129	83.00	-39.03125	-9.00000	52.56250	38.27	0.0
1	12.375	27.2425	0.409410	76.33	-38.96875	-9.00000	52.65625	38.27	0.0
1	13.375	8.1350	0.411331	85.50	-39.00000	-9.00000	52.71875	38.27	0.0
1	14.375	-3.4575	0.411650	78.80	-38.96875	-9.03125	52.87500	38.23	0.0

Figure 28 shows the results for outlier detection in the dataset. The outliers are defined as values which are not in normal range of data and falling above or below of regular range. It is important to detect and remove the outliers as it will not provide correct data distribution. Accelerometer x-axis has zero outliers, temperature has 20 outlier values, heart rate has 123 and followed by other columns. We will be removing these values.

Figure 28

Shows outliers present in Biostress dataset.

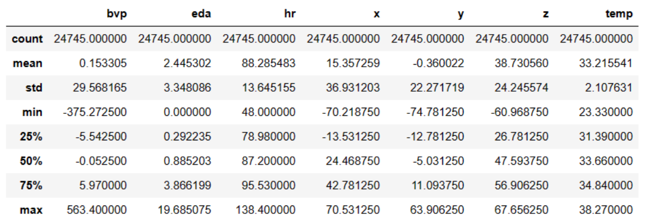


```
Number of outliers in each column:
bvp      597
eda      876
hr       123
x          0
y         10
z        310
temp      20
dtype: int64
```

Figure 29 shows the summary statistics of the dataset. There are 24745 number of total rows. The minimum and maximum value of all the columns used in the project are listed along with standard deviation and mean across all column labels. This would give statistical description which can be leveraged at advanced stages for model development and creating visualizations.

Figure 29

Shows summary statistics for Biostress dataset

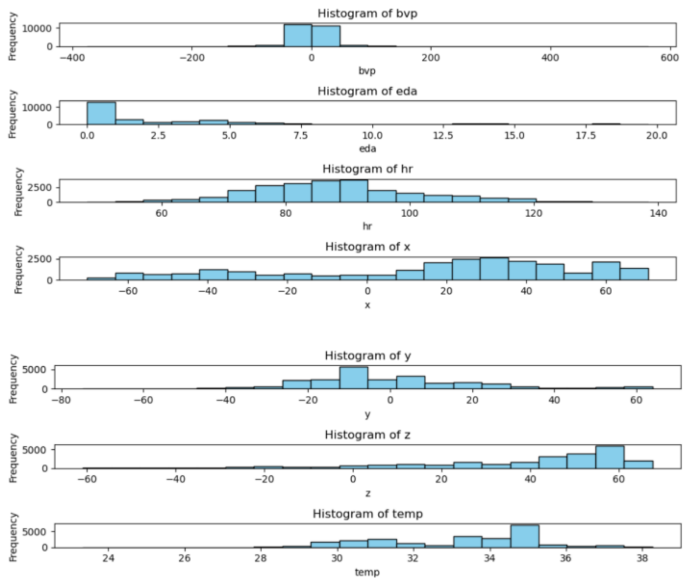


	bvp	eda	hr	x	y	z	temp
count	24745.000000	24745.000000	24745.000000	24745.000000	24745.000000	24745.000000	24745.000000
mean	0.153305	2.445302	88.285483	15.357259	-0.360022	38.730560	33.215541
std	29.568165	3.348086	13.645155	36.931203	22.271719	24.245574	2.107631
min	-375.272500	0.000000	48.000000	-70.218750	-74.781250	-60.968750	23.330000
25%	-5.542500	0.292235	78.980000	-13.531250	-12.781250	26.781250	31.390000
50%	-0.052500	0.885203	87.200000	24.468750	-5.031250	47.593750	33.660000
75%	5.970000	3.866199	95.530000	42.781250	11.093750	56.906250	34.840000
max	563.400000	19.685075	138.400000	70.531250	63.906250	67.656250	38.270000

Figure 30 shows the frequency distribution of data points in dataset. Histogram plot is created for BVP, EDA, HR, x, y, z and temperature. These provide information regarding range of values, max and min values, number of data points falling in each bin.

Figure 30

Shows frequency distribution of data points in Biostress dataset

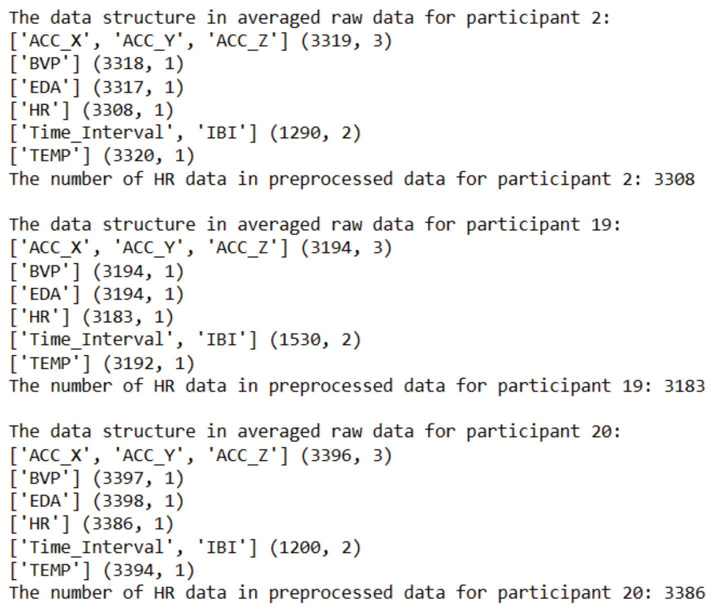


Stress-Predict dataset

Figure 31 shows sample of participants data who are undergoing stress test. The physiological signals of these participants are stored in separate folders for each participant. Data belonging to all participants is combined to one file and then preprocessing is done.

Figure 31

Shows sample of participants data in Stress Predict dataset

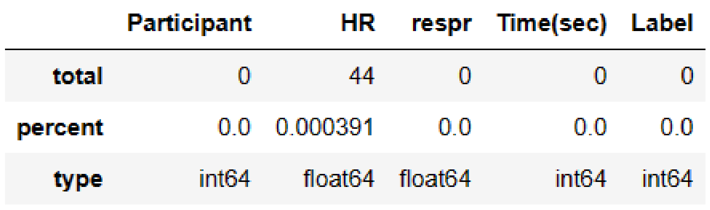


```
The data structure in averaged raw data for participant 2:
['ACC_X', 'ACC_Y', 'ACC_Z'] (3319, 3)
['BVP'] (3318, 1)
['EDA'] (3317, 1)
['HR'] (3308, 1)
['Time_Interval', 'IBI'] (1290, 2)
['TEMP'] (3320, 1)
The number of HR data in preprocessed data for participant 2: 3308

The data structure in averaged raw data for participant 19:
['ACC_X', 'ACC_Y', 'ACC_Z'] (3194, 3)
['BVP'] (3194, 1)
['EDA'] (3194, 1)
['HR'] (3183, 1)
['Time_Interval', 'IBI'] (1530, 2)
['TEMP'] (3192, 1)
The number of HR data in preprocessed data for participant 19: 3183

The data structure in averaged raw data for participant 20:
['ACC_X', 'ACC_Y', 'ACC_Z'] (3396, 3)
['BVP'] (3397, 1)
['EDA'] (3398, 1)
['HR'] (3386, 1)
['Time_Interval', 'IBI'] (1200, 2)
['TEMP'] (3394, 1)
The number of HR data in preprocessed data for participant 20: 3386
```

Figure 32 depicts missing values in Stress-Predict dataset for every column time, blood volume pressure (BVP), electrodermal activity (EDA), heart rate (HR), temperature (TEMP) and 3-axis (ACC_X, ACC_Y, ACC_Z) accelerometer (ACC) and label. There are 44 missing values in the heart rate HR column and zero missing values in other columns.

Figure 32

Shows missing values in Stress Predict dataset.

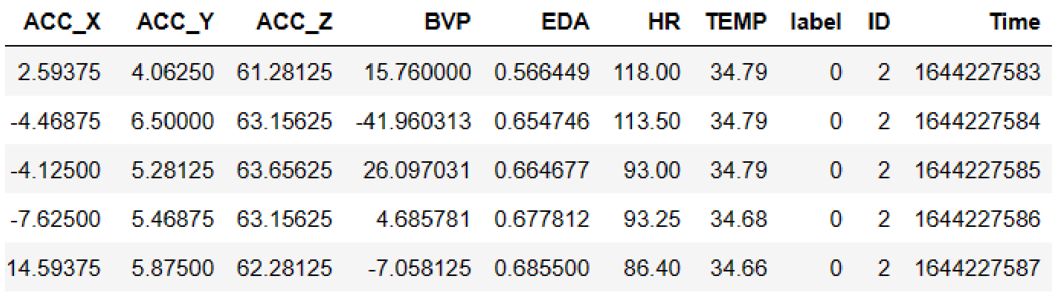


	Participant	HR	respr	Time(sec)	Label
total	0	44	0	0	0
percent	0.0	0.000391	0.0	0.0	0.0
type	int64	float64	float64	int64	int64

Figure 33 shows the dataset after converting sampling frequencies of feature variables to 1 Hz. Frequency sampling rate for columns are different. Frequency sampling rate for features is 64 Hz, 4Hz, 32 Hz, and 4 Hz. It is vital to convert them to one frequency sampling rate which is 1Hz for utilizing the columns in our project.

Figure 33

Shows data converted into 1Hz frequency in Stress Predict dataset.

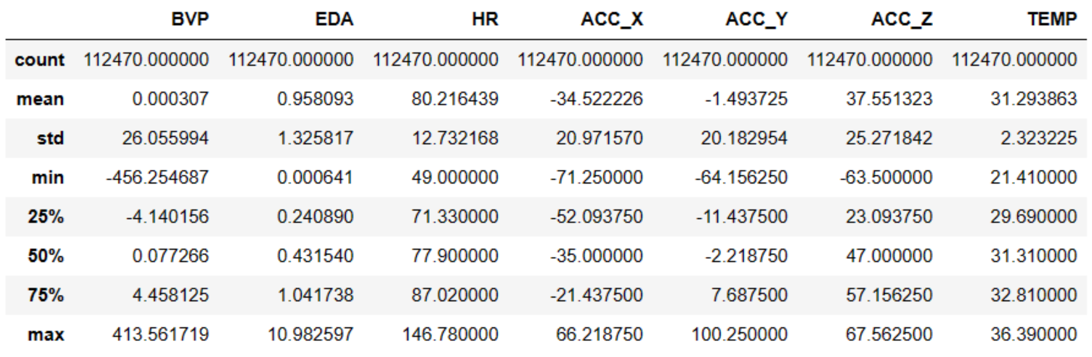


ACC_X	ACC_Y	ACC_Z	BVP	EDA	HR	TEMP	label	ID	Time
2.59375	4.06250	61.28125	15.760000	0.566449	118.00	34.79	0	2	1644227583
-4.46875	6.50000	63.15625	-41.960313	0.654746	113.50	34.79	0	2	1644227584
-4.12500	5.28125	63.65625	26.097031	0.664677	93.00	34.79	0	2	1644227585
-7.62500	5.46875	63.15625	4.685781	0.677812	93.25	34.68	0	2	1644227586
14.59375	5.87500	62.28125	-7.058125	0.685500	86.40	34.66	0	2	1644227587

Figure 34 shows the summary statistics of the dataset. There are 112470 number of total rows. The minimum and maximum data values in the dataset are listed column wise along with standard deviation std and mean for features. This would provide statistical information for creating insightful visualizations and machine learning models.

Figure 34

Shows summary statistics of Stress Predict dataset

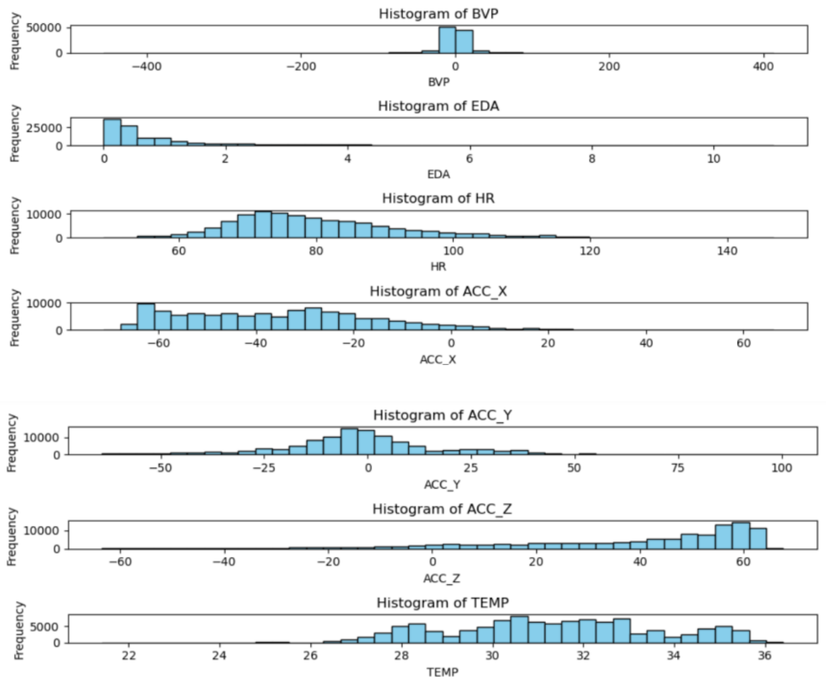


	BVP	EDA	HR	ACC_X	ACC_Y	ACC_Z	TEMP
count	112470.000000	112470.000000	112470.000000	112470.000000	112470.000000	112470.000000	112470.000000
mean	0.000307	0.958093	80.216439	-34.522226	-1.493725	37.551323	31.293863
std	26.055994	1.325817	12.732168	20.971570	20.182954	25.271842	2.323225
min	-456.254687	0.000641	49.000000	-71.250000	-64.156250	-63.500000	21.410000
25%	-4.140156	0.240890	71.330000	-52.093750	-11.437500	23.093750	29.690000
50%	0.077266	0.431540	77.900000	-35.000000	-2.218750	47.000000	31.310000
75%	4.458125	1.041738	87.020000	-21.437500	7.687500	57.156250	32.810000
max	413.561719	10.982597	146.780000	66.218750	100.250000	67.562500	36.390000

Figure 35 depicts the frequency distribution of values in data. Histogram Frequency distribution plot is created for BVP, EDA, HR, x, y, z and TEMP columns. These show information regarding range of values, max and min values, number of data points falling in each bin, distribution of data uniform, right skewed or left skewed.

Figure 35

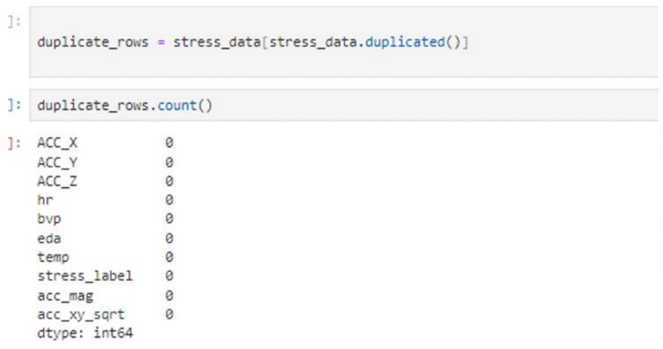
Shows frequency distribution of data points in Stress Predict dataset



Once the dataset is merged into one file from both the data sources biostress and stress predict, file is checked for duplicate values. Our dataset contains no duplicate values as shown in Figure 36.

Figure 36

Shows duplicate value check.



```
]:  duplicate_rows = stress_data[stress_data.duplicated()]

]:  duplicate_rows.count()

]:  ACC_X           0
    ACC_Y           0
    ACC_Z           0
    hr              0
    bvp             0
    eda             0
    temp            0
    stress_label    0
    acc_mag         0
    acc_xy_sqrt     0
    dtype: int64
```

Data Transformation

Feature Engineering Process

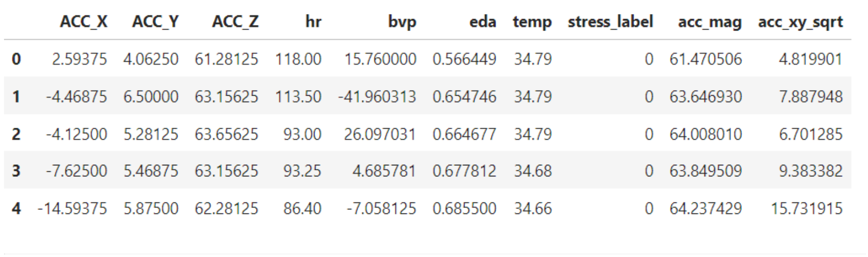
Features from both the Bio stress dataset and Stress-Predict Dataset are extracted and processed to create the combined generic dataset for stress prediction level. Irrelevant features are removed while new derived features are added to enhance dimensionality as well as the model performance. The combined dataset has student identification numbers and timestamps for each signal recorded. As part of feature engineering these features are dropped. Studies conducted by Rogers, M et al. (1997) on accelerometer data analysis suggest that with x, y and z-axis data cumulative acceleration can be computed as a function of frequency (Rogers, M et al., 1997). The cumulative acceleration is calculated using Equation 1.

$$accel = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

In addition to it another similar feature is added where x and y data is considered only as z data has high variation with the other axes. Figure 37 shows the data snippet post-addition of derived features for accelerometer data.

Figure 37

Derived feature addition data snippet from combined stress dataset

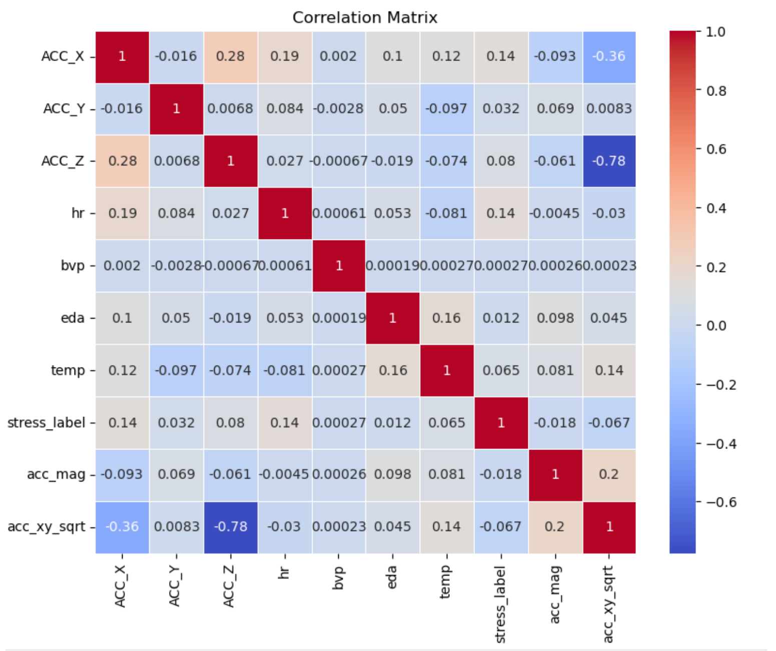


	ACC_X	ACC_Y	ACC_Z	hr	bvp	eda	temp	stress_label	acc_mag	acc_xy_sqrt
0	2.59375	4.06250	61.28125	118.00	15.760000	0.566449	34.79	0	61.470506	4.819901
1	-4.46875	6.50000	63.15625	113.50	-41.960313	0.654746	34.79	0	63.646930	7.887948
2	-4.12500	5.28125	63.65625	93.00	26.097031	0.664677	34.79	0	64.008010	6.701285
3	-7.62500	5.46875	63.15625	93.25	4.685781	0.677812	34.68	0	63.849509	9.383382
4	-14.59375	5.87500	62.28125	86.40	-7.058125	0.685500	34.66	0	64.237429	15.731915

Correlation matrix was plotted to visualize the correlation between features and target label as shown in Figure 38. Correlation matrix shows the positively and negatively correlated features. Features which have the highest positive correlation value will be strongly related to the target variable. This means features ACC_X and hr are highly positively correlated to stress label.

Figure 38

Represents Correlation Matrix for dataset.

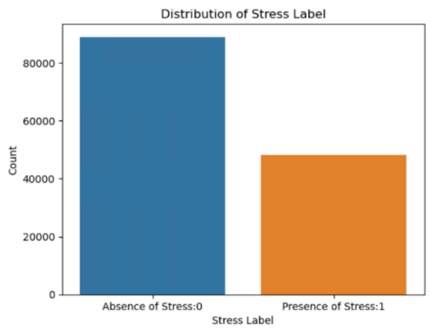


Data Imbalance Handling

The combined stress dataset has a class imbalance in the target variable – stress label. The signal recorded for stress is labelled as 1, which is an indicator of the presence of stress in the participant based on other input signals from the wearables. The absence of stress in the signal is labelled as 0, which is an indicator that the participant is not going through stress. Figure 39 shows the class imbalance in predictor variable – stress label in the combined dataset.

Figure 39

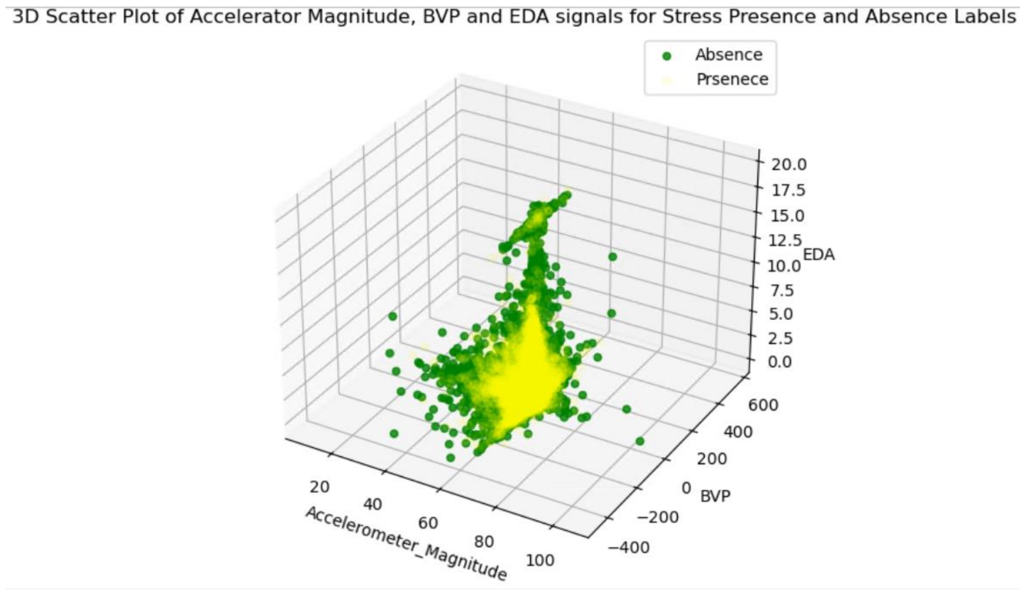
Stress Label distribution in the combined stress dataset.



As per the plot, the current data set has the presence of stress indicator data around 40000 whereas the absence of stress data present in the dataset is nearly 90000. This skewness in the data will negatively impact the building of concrete decision boundaries to build a classification model. The machine learning models will be used further down in this project depending on the class distribution and could perform poorly due to overpowering the majority class. Figure 40 shows the imbalance in Stress Absence and Presence labels in a 3-dimensional plot before performing the oversampling technique on the combined stress dataset.

Figure 40

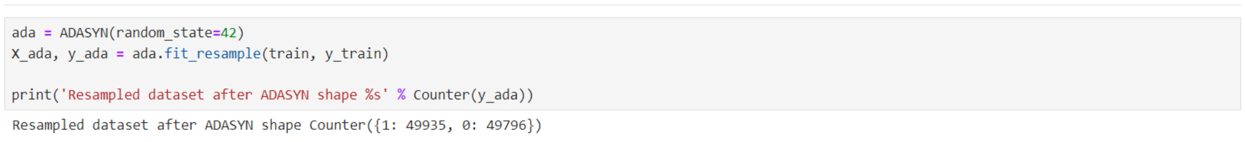
3D Scatter Plot of Accelerometer Magnitude, BVP and EDA signals for different Stress Labels before oversampling technique.



Oversampling technique – Adaptive Synthetics Sampling (ADASYN) is applied to the combined stress dataset to balance the target variable, stress labels. ADASYN algorithm generates synthetic minority class data without copying the data. This algorithm uses density distribution to decide the number of samples required to generate for each minority class sample. Figure 41 shows the code snippet for the ADASYN technique implementation on the stress dataset. (Haibo He et al., 2008). After the sampling class label distribution is 49935 records for stress label 1 and 49796 records for class label 0. code snippet used for the project.

Figure 41

ADASYN oversampling code snippet



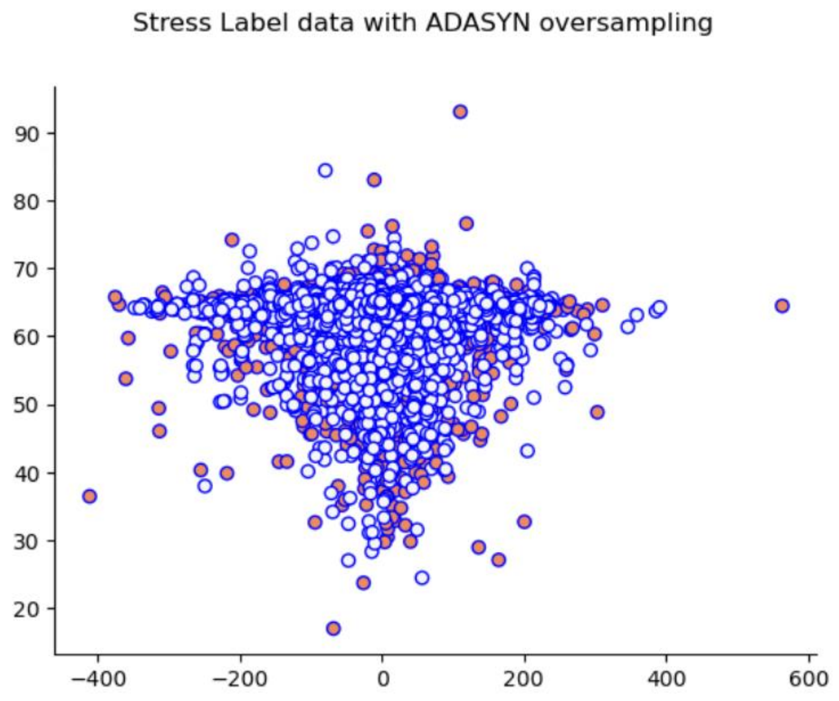
```
ada = ADASYN(random_state=42)
X_ada, y_ada = ada.fit_resample(train, y_train)

print('Resampled dataset after ADASYN shape %s' % Counter(y_ada))
```

```
Resampled dataset after ADASYN shape Counter({1: 49935, 0: 49796})
```

Figure 42 shows the scatter plot for BVP and Accelerometer magnitude signals after ADASYN oversampling for minority target class label stress predict 0.

Figure 42

Scatter plot for BVP signal and Accelerometer magnitude after ADASYN over sampling

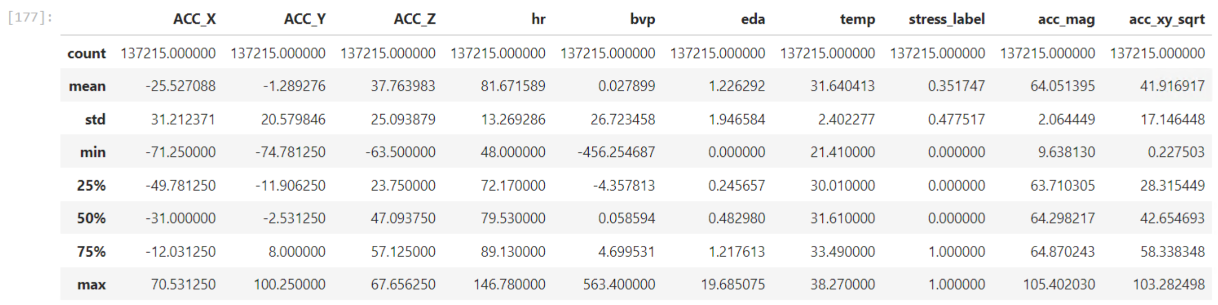


Data Standardization

As a part of the preprocessing step, features are scaled using Scalar standardization. The initial dataset has features with huge differences in their ranges of values. The figure shows the combined stress dataset statistics. EDA signal has a range from 0 to 19.68 whereas the BVP signal value ranges from -456.25 to 563.4. Figure 43 shows stress dataset statistics before standardization.

Figure 43

Stress dataset statistics.



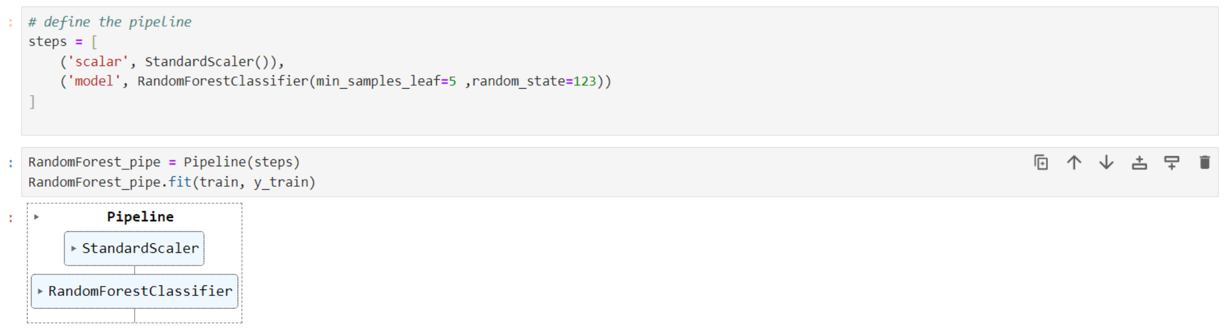
```
[177]:
```

	ACC_X	ACC_Y	ACC_Z	hr	bvp	eda	temp	stress_label	acc_mag	acc_xy_sqrt
count	137215.000000	137215.000000	137215.000000	137215.000000	137215.000000	137215.000000	137215.000000	137215.000000	137215.000000	137215.000000
mean	-25.527088	-1.289276	37.763983	81.671589	0.027899	1.226292	31.640413	0.351747	64.051395	41.916917
std	31.212371	20.579846	25.093879	13.269286	26.723458	1.946584	2.402277	0.477517	2.064449	17.146448
min	-71.250000	-74.781250	-63.500000	48.000000	-456.254687	0.000000	21.410000	0.000000	9.638130	0.227503
25%	-49.781250	-11.906250	23.750000	72.170000	-4.357813	0.245657	30.010000	0.000000	63.710305	28.315449
50%	-31.000000	-2.531250	47.093750	79.530000	0.058594	0.482980	31.610000	0.000000	64.298217	42.654693
75%	-12.031250	8.000000	57.125000	89.130000	4.699531	1.217613	33.490000	1.000000	64.870243	58.338348
max	70.531250	100.250000	67.656250	146.780000	563.400000	19.685075	38.270000	1.000000	105.402030	103.282498

Data Standardization rescales the features around a mean of 0 and a standard deviation of 1. It prevents the impact of outliers and allows direct comparison of model coefficients. For data standardization, the pipeline is built to perform standard scalar transformation and random forest classification. Figure 44 shows code snippet for Standardization and random forest classification pipeline for the data.

Figure 44

Code snippet for Standardization and random forest classification pipeline.



```
# define the pipeline
steps = [
    ('scalar', StandardScaler()),
    ('model', RandomForestClassifier(min_samples_leaf=5 ,random_state=123))
]
```

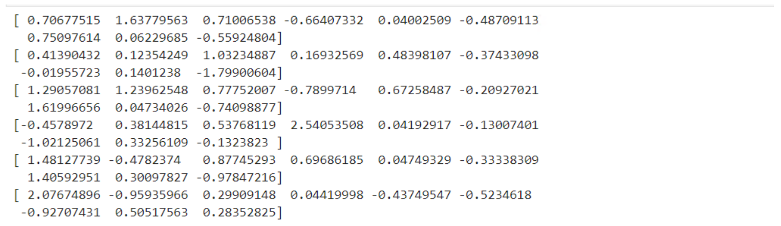
```
RandomForest_pipe = Pipeline(steps)
RandomForest_pipe.fit(train, y_train)
```

```
Pipeline
  StandardScaler
  RandomForestClassifier
```

After standardization, feature values will be transformed. Figure 45 shows the transformed data after scalar standardization.

Figure 45

Data after performing scalar standardization.



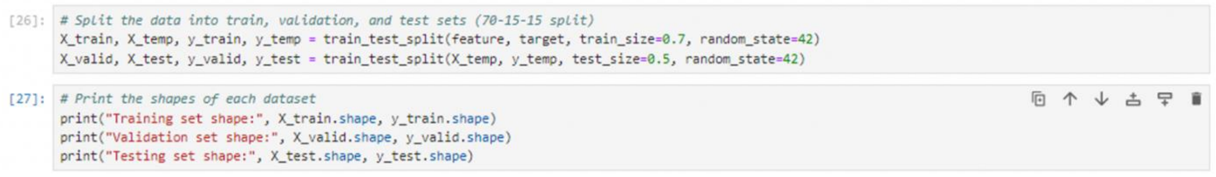
```
[ 0.70677515  1.63779563  0.71006538 -0.66407332  0.04002509 -0.48709113
  0.75097614  0.06229685 -0.55924804]
[ 0.41390432  0.12354249  1.03234887  0.16932569  0.48398107 -0.37433098
 -0.01955723  0.1401238  -1.79900604]
[ 1.29057081  1.23962548  0.77752007 -0.7899714   0.67258487 -0.20927021
  1.61996656  0.04734026 -0.74098877]
[-0.4578972   0.38144815  0.53768119  2.54053508  0.04192917 -0.13007401
 -1.02125061  0.33256109 -0.1323823 ]
[ 1.48127739 -0.4782374   0.87745293  0.69686185  0.04749329 -0.33338309
  1.40592951  0.30097827 -0.97847216]
[ 2.07674896 -0.95935966  0.29909148  0.04419998 -0.43749547 -0.5234618
 -0.92707431  0.50517563  0.28352825]
```

Data Preparation

After data preprocessing and data transformation steps transformed data is split into test and train sets. 70% of the entire data is used for training the model while 15% of the data is kept as test data and 15% of the data is used as validation purpose. This test data is used for model evaluation. Figure 46 shows the train and test and validation data split code snippet.

Figure 46

Code snippet for train, test and validation data split.

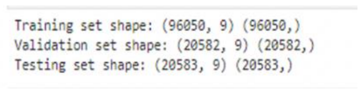


```
[26]: # Split the data into train, validation, and test sets (70-15-15 split)
      X_train, X_temp, y_train, y_temp = train_test_split(feature, target, train_size=0.7, random_state=42)
      X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

[27]: # Print the shapes of each dataset
      print("Training set shape:", X_train.shape, y_train.shape)
      print("Validation set shape:", X_valid.shape, y_valid.shape)
      print("Testing set shape:", X_test.shape, y_test.shape)
```

After performing the first train test split, train data has 96050 rows and 9 columns whereas test data has 20582 records with 9 columns. The test set data portion is kept for the final evaluation of the models. The validation data has 20583 records with 9 columns and this portion is used for the model validation. Figure 47 shows the result from train, test and validation data split done on the dataset.

Figure 47

Code snippet for train, test and validation data split result.

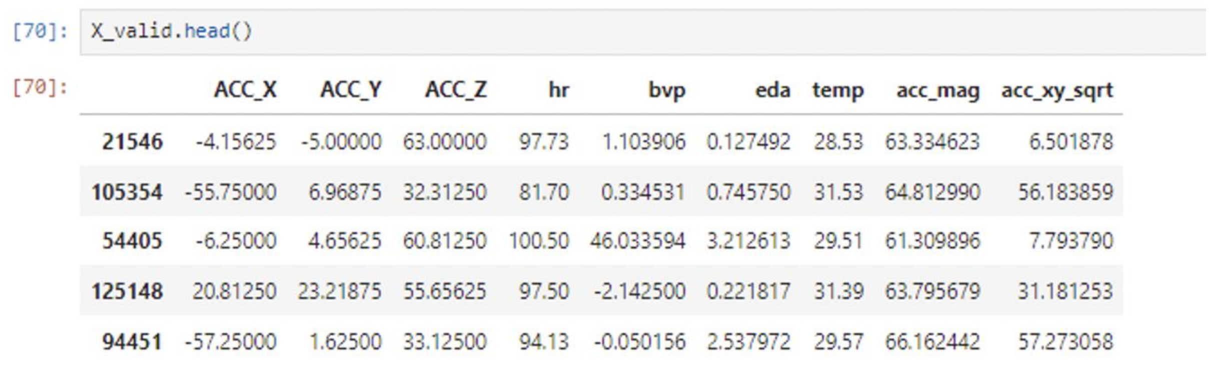


```
Training set shape: (96050, 9) (96050,)
Validation set shape: (20582, 9) (20582,)
Testing set shape: (20583, 9) (20583,)
```

This train dataset will be used to build various machine learning models and valid dataset will be used for model validation. Figure 48 shows sample data from a valid dataset.

Figure 48

Sample data from valid dataset.



```
[70]: X_valid.head()
```

	ACC_X	ACC_Y	ACC_Z	hr	bvp	eda	temp	acc_mag	acc_xy_sqrt
21546	-4.15625	-5.00000	63.00000	97.73	1.103906	0.127492	28.53	63.334623	6.501878
105354	-55.75000	6.96875	32.31250	81.70	0.334531	0.745750	31.53	64.812990	56.183859
54405	-6.25000	4.65625	60.81250	100.50	46.033594	3.212613	29.51	61.309896	7.793790
125148	20.81250	23.21875	55.65625	97.50	-2.142500	0.221817	31.39	63.795679	31.181253
94451	-57.25000	1.62500	33.12500	94.13	-0.050156	2.537972	29.57	66.162442	57.273058

Data Statistics

Table 4 shows how the dataset’s size changed from the data collection to the data preparation phases of the project. Two raw datasets, namely the Stress-predict dataset and Biostress datasets were collected initially. Each of these files contains data on 6 different physiological signals, namely accelerometer data, BVP data, EDA data, HR data, IBI data, and Temp data of each volunteer. Each of these physiological signals is stored in different CSV files.

After the pre-processing step, all 6 individual files of the physiological signals were merged for both Stress Predict datasets. Now the signal dataset for the Stress predict dataset has 24745 rows and 11 columns. Similarly, the files from the Biostress dataset are also combined. The merged Biostress dataset has 112470 rows and 12 columns.

Table 4

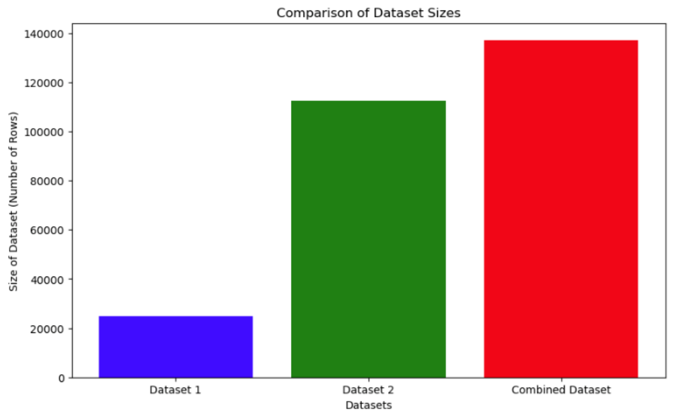
Summary of Dataset sizes after different stages

Stage	Process	Size(Rows X columns)
Raw	Stress predict Dataset	Individual CSV files for each feature
	BIOSTRESS dataset	Individual CSV files for each feature
	Stress Predict Dataset	24745 X 11
	BIOSTRESS Dataset	112470 X 12
	Combined Dataset	137215 X 11
Pre-Processing	Combined dataset after dropping columns	137215 X 8
	Feature engineering (Acc_Mag)	137215 X 9
Transformation	Feature Engineering (acc_XY_sqrt)	137215 X 10
	Training Data	96050 X 9
	Testing Data	20582 X 9
Preparation	Validation Data	20583 X 9

These two processed datasets must be merged to make a single dataset. A single combined dataset has been created. Figure 49 shows that the newly created file has 137215 rows and 11 columns. The 11 columns include unnamed 0, id, time, X, Y, Z, HR, BVP, EDA, Temp, label. Out of 11 columns unnamed 0, id, time are unnecessary columns. So, the 3 columns have been dropped. Now the effective size of the new dataset is 137215 X 8.

Figure 49

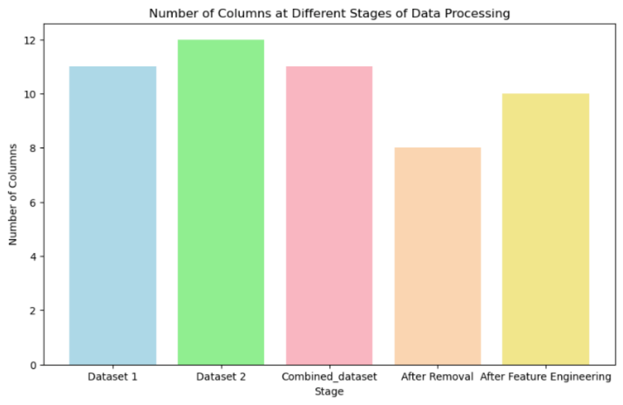
Shows number of rows present in the dataset



During data transformation, feature engineering has been applied to the combined dataset. To enhance the dimensionality and model performance two additional new features have been added. One feature was added using the X, Y, and Z axis of accelerometer data. Another feature is also added using the same accelerometer data in which only X, and Y axis data is used, and the Z axis is ignored in this. Figure 50 shows that the newly formed dataset with the added feature has 137215 rows and 10 columns.

Figure 50

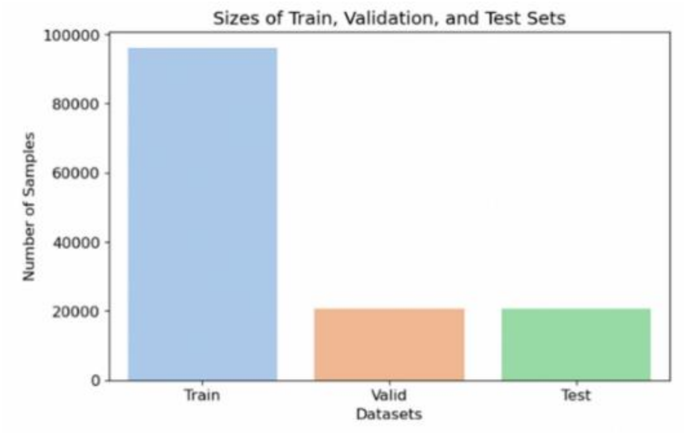
Shows number of columns in the dataset after each phase



Finally, in the data preparation step, the dataset is split into test and train sets. 70% of the entire data is for processing and 30% of the remaining data is kept for evaluation. After this train test split dataset has 96050 rows and 9 columns. The test data has 41165 rows and 9 columns. This train dataset is further split into train and validation sets with an 80:20 ratio. The train data is used for the model building and the validation data is used for the validation. After this split the train data has 76840 rows and 9 columns. Figure 51 shows that validation datasets have 19210 rows and 9 columns.

Figure 51

Shows Bar chart showing size after train test split



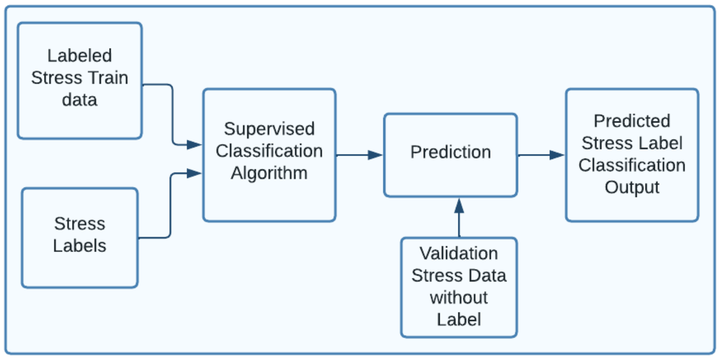
Model Development

Model Proposals

The Cognitive stress level prediction project aims to predict stress level of an individual through the signals collected from the sensory wearable device. Stress levels are categorized in two types- 1 as presence of stress and 0 as absence of stress. We propose to develop K-nearest neighbor (KNN) as supervised classification model to predict stress levels in the individuals. KNN works on the principle of similarity or proximity. Whenever a new sample or test data point arrives, its distance is calculated from the training samples and accordingly it is classified into classes. Both the datasets - the Stress-Predict Dataset and the Biostress Dataset have different signals recorded for each participant like Accelerometer data, heart rate and respiratory rate, skin conductance EDA, skin temperature data etc. We cleaned and transformed all these different sensory signals recorded at multiple frequencies into one single 1Hz frequency to build up one common dataset. On the combined dataset we have computed derived features which raises the feature count to 9. The class labels of the dataset are given as 0 for absence of the stress level and 1 for presence of the stress level. The combined dataset has data imbalanced class labels. ADASYN oversampling technique is used to resample and add synthetic data which produced balanced class labels and Overview of Supervised Binary Stress Level Classification Model is explained in Figure 52

Figure 52

Overview of Supervised Binary Stress Level Classification Model


Labeled Stress Train data
Stress Labels
Supervised Classification Algorithm
Prediction
Predicted Stress Label Classification Output
Validation Stress Data without Label

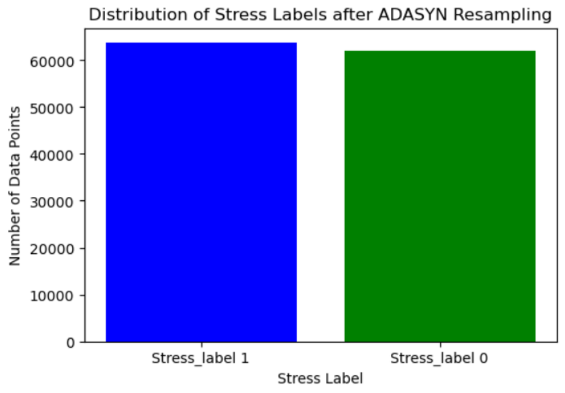
K-Nearest Neighbors (KNN)

K-NN is a supervised machine learning algorithm which can be used for regression and classification both. In this project, K-NN is utilized for classification of stress labels as 0 for absent and 1 for present. Supervised algorithms are machine learning algorithms which are trained using labeled data. K-NN is called lazy learning algorithm as it stores the training data and does nothing until test or unseen data arrives. The model calculates the similarity between test data points and training instances and selects k- nearest neighbors. K- nearest neighbors are classified according to distance metrics. We have used distance metrics such as Euclidean, Manhattan, Chebyshev and minkowski for calculating k- nearest neighbors.

The K-NN algorithm considers the assumption that samples are evenly distributed. In practical scenarios which is not always the case (Sun et al., 2010). The dataset used in our project was unbalanced which means many instances belonged to majority class and few numbers of instances belonged to minority class. Adaptive sampling technique (ADASYN) was utilized to equally distribute instances across both the classes as shown in Figure 53.

Figure 53

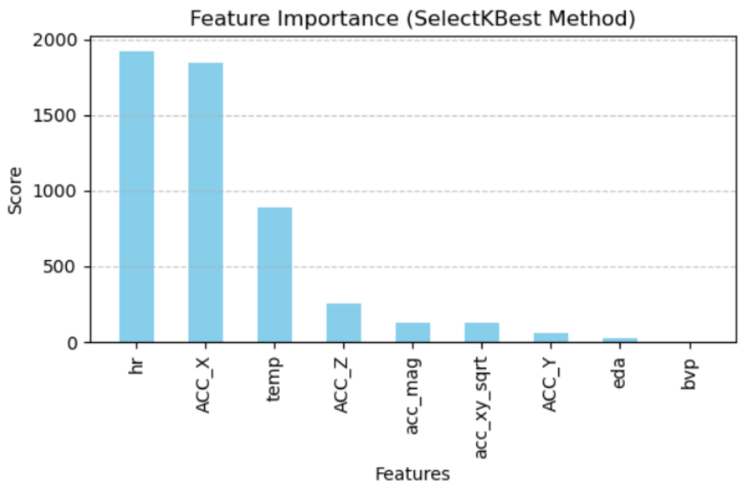
Represents distribution of Stress Labels after ADASYN Resampling



Relationship strength between features and target variable can be known through feature importance. SelectKBest method was used with ANOVA f-statistical value to find the important features for k-nearest neighbor algorithms. ANOVA f-value is determined by calculating variance between groups and within each group. A score is provided to features based on the ANOVA f-value. The higher ANOVA f-value, higher will be the score or importance attached to the feature. The highest score was attached to the heart rate (HR) followed by ACC_X. The feature importance through SelectKBest method was plotted as shown in Figure 54.

Figure 54

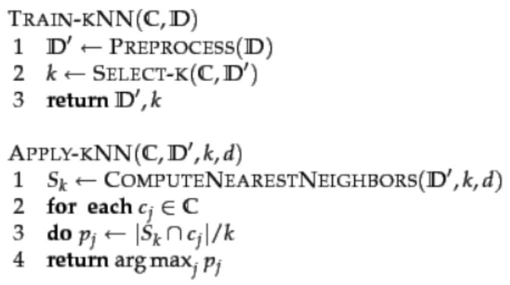
Represents feature importance through SelectKBest method



The k- nearest neighbor algorithm can be explained through a series of steps. The first step comprises of deciding on the number of nearest neighbors. The second step will be the calculation of distance between new data points and training data samples. The third step will be calculation of data points belonging to each class based on the selected nearest neighbors. Next step is to assign classes to unseen data points according to majority voting principle (Gupta et al., 2023). The methodology of the K-NN algorithm is shown in Figure 55.

Figure 55

Represents K-NN training and testing process in model development.

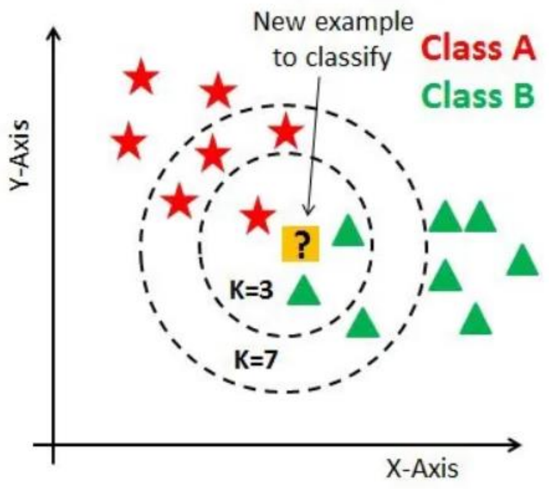


```
TRAIN-KNN($\mathbb{C}, \mathbb{D}$)
1  $\mathbb{D}' \leftarrow$ PREPROCESS($\mathbb{D}$)
2  $k \leftarrow$ SELECT-K($\mathbb{C}, \mathbb{D}'$)
3  return $\mathbb{D}', k$

APPLY-KNN($\mathbb{C}, \mathbb{D}', k, d$)
1  $S_k \leftarrow$ COMPUTENEARESTNEIGHBORS($\mathbb{D}', k, d$)
2  for each $c_j \in \mathbb{C}$
3  do $p_j \leftarrow |S_k \cap c_j|/k$
4  return $\arg\max_j p_j$
```

K-NN decides decision boundary locally and allocates document to majority class of its k-nearest neighbors as shown in Figure 56.

Figure 56

Represents K-NN classification of test sample.



We have a testing document d along with same label as training samples. We usually use odd value of k to avoid ties. $S_k(D)$ is the d number of k nearest neighbors and $I_c(d')$ is equivalent to 1 when d' belongs to class and 0 if not. The document is allocated to the class with highest weightage which is calculated by similarity. In Equation 2 similarity of the k nearest neighbors is determined by their cosine similarity.

$$score(c, d) = \Sigma_{d' \epsilon S_k(d)} I_C\,(d') \cos\,(\vec{v}(d'), \vec{v}(d)) \tag{2}$$

Model Optimization

k-nearest neighbor algorithm is widely used for classification problems due to its ease of use and understanding. This algorithm is simple to implement. To utilize this algorithm for real time data which is scalable in nature we need to consider hyperparameter tuning for optimizing the performance. The hyperparameter tuning refers to finding out the parameters which will result in improved performance of algorithm by considering time and resource optimization. We have used GridSearch technique for finding the best k- nearest neighbors. Cross Validation techniques have been implemented through five folds. GridSearch tunes the parameters through a cross validated model for different sets of parameters. Parameters used were n_neighbors (3, 5, 7, 9, 11), weights (uniform and distance), and metric (Euclidean, Minkowski, Manhattan, Chebyshev and Cosine similarity). We have utilized k-fold cross validation technique along with GridSearch. K-fold cross validation technique divides the training data into two parts where one part is used for testing and the remaining k-1 is used for training the model in first iteration. This process will go through series of iterations, and we will get the performance of model through averaging the performance in each iteration.

Model Supports

Environment, ML platform and Tools

All models in this project are built in Jupyter notebook installed on Anaconda, which is a free open-source distribution of python and contains large amounts of packages for preprocessing and machine learning models. The data preparation steps mainly rely on NumPy and pandas libraries. For the modeling part, the traditional Machine learning models, such as KNN, were trained with Windows 10 operating system and CPU environment using sklearn packages. The table 5 below showcases the detailed information and usages of the package used in this paper.

Table 5

Packages for Model Development

Library	**Method**	**Usage**
Os	os.path.join os.path.isdir os.path.basename	import operating system and read all raw data in specific folder
Glob	glob.glob	Combined with os to find the target file path
Numpy&Pandas	pd.read_csv pd.concat groupby dropna mean shape describe	Import dataframe to read the csv files. Approximate statistics and dataset structure review. Perform various preprocessing on dataframe,

		including calculation, removing or add columns, transformation and combine the dataset
Matplotlit&Seaborn	sns.histplot sns.countplot sns.barplot sns.heatmap plt.figure plt.subplot plt.tight_layout plt.legend plt.show	Visualize the distribution of features. Scale the figure into a proper size. Add legend on the axis and display the graph
Imblearn	imblearn.over_sampling	handle the imbalanced data
Sklearn	sklearn.preprocessing sklearn.model_selection sklearn.linear_model sklearn.neighbors sklearn.ensemble	Standardization and then split the dataset into train, validation and test. Building traditional models and evaluating model

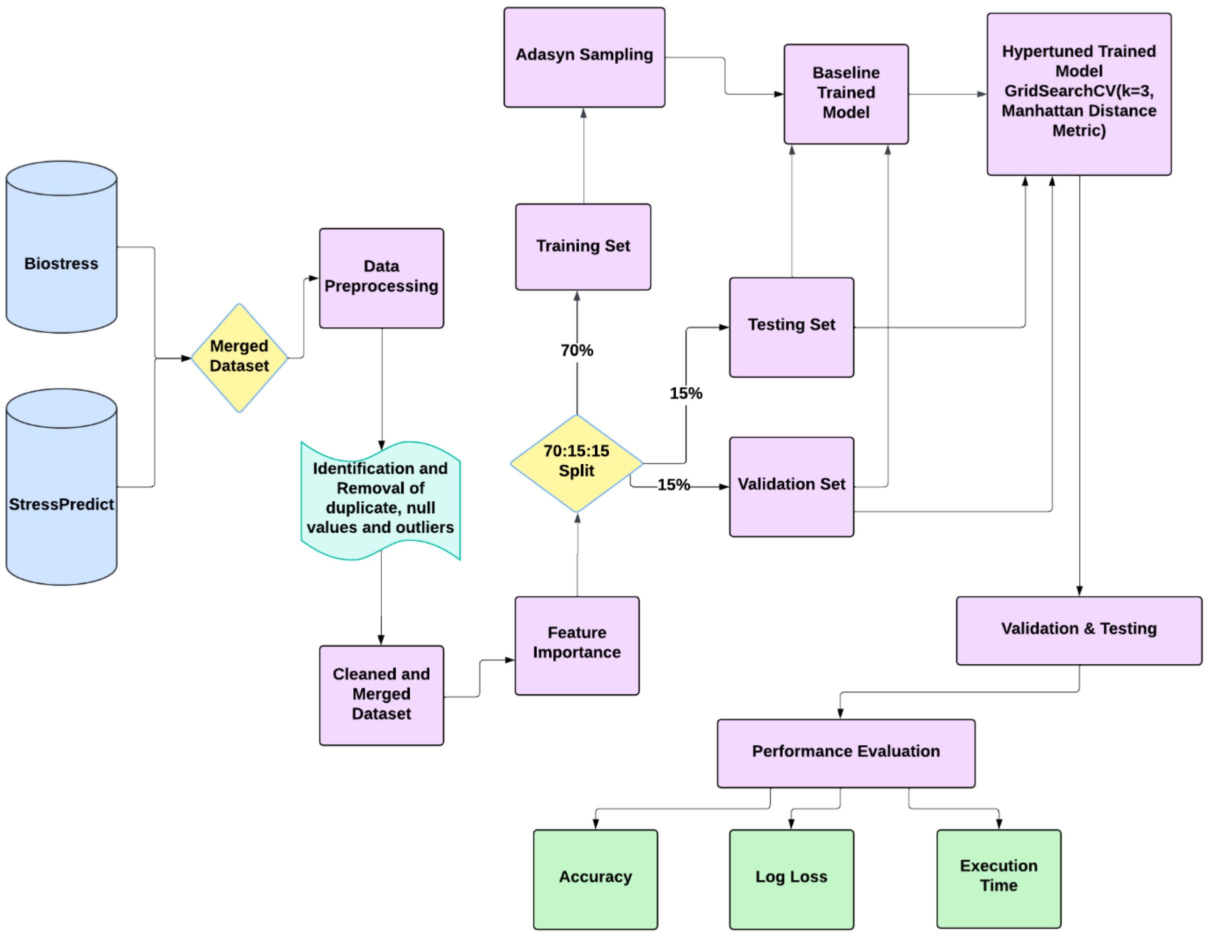
	sklearn.pipeline sklearn.metrics	performance using several classification metrics and confusion matrix.
Time	time.time	Track the model training time

Model Architecture and Dataflow

System architecture for K-NN model consists of data collection, data preprocessing, feature importance, model development as shown in Figure 57. Data is collected from two sources bio stress and stress prediction and merged into one. Data processing and feature importance is carried out. The data is split into 70:15:15 as training, validation and testing set for the model development. Baseline model for K-NN was developed with default parameters. Hyperparameter tuning was applied to optimize the performance of the model. GridSearchCV along with k-fold cross validation technique was used across distance metrics such as manhattan, minkowski, Chebyshev and cosine similarity. The K-NN model was evaluated through various performance metrics.

Figure 57

KNN model flow and System architecture



Model Comparisons and Justification

The KNN model was implemented with default parameters as baseline model and with tuned parameters as hypertuned model. The hypertuned model with k value as 3 and distance metric as manhattan outperformed all other KNN models. For current binary classification problems under the scope of project, Logistic regression, KNN, random forest, XGBoost algorithms are proposed as these algorithms are proven to achieve better accuracy in similar studies with different datasets. Logistic regression model gave lowest accuracy. Highest accuracy was achieved in random forest, but training time was highest for random forest among all the models. DNN took the second highest training time among all the models implemented. In terms of accuracy and training time KNN proved to be best model among all followed by XGBoost. Advantages and disadvantages of all the models proposed for this project are discussed in Table 6.

Table 6

Model Comparison

Model Name	**Advantage**	**Disadvantage**
Logistic Regression	Less prone to Overfitting Computationally less Intensive Faster training speeds Can easily understand influence of each Feature.	Due to its linear nature, logistic regression can underperform when relationships in data are nonlinear Logistic regression assumes linearity between the independent variables and the log odds. Can not capture complex relationships
KNN	Easy and simple to understand and implement K-NN works well with non linear decision boundaries This does not make assumption regarding shape of decision boundary in comparison to other algorithms Various distance metrics are supported to address different	K-NN may not work well with large size data This is affected by curse of dimensionality Large memory requirement All of the training data is saved in memory"

	domains"	
XGBoost	Takes less time to execute Feature importance score will be generated Algorithm does not require scaled data. Also it handles null values and outliers well. It can handle large dataset It reduces the overfitting"	Overfitting can happen when the parameters are not tuned properly This is a blackbox model , which impacts model interpretability
Random Forest	They are a two-purpose: classifier and regressor. They do not require much parameter tuning, and they are insensitive to which set of hyperparameters you choose. They measure what features make the biggest difference in performance (on average). That information can be helpful for	It's time-consuming to make random forests, since if they have many trees and even more data columns the calculations really mount up. This is probably not a good performance method if the source data is completely unbalanced or there are many noise-intensive characteristics in

	dropping less important ones.	database. Big trees are hard to read; when you have a series of them, it becomes difficult to understand what a single tree's decisions mean.
Deep Neural Network	DNN learns hierarchical representation of data and it can capture the non-linear and complex relationships, scale well with large and diverse dataset	More hyperparameters to tune. Need more computation resources. It's also challenging to interpret the internal working.

Justification for the KNN model

K-nearest neighbors is used for the binary classification of stress label as 0 for absent and 1 for present. KNN is easy to implement and understand. KNN is a non-parametric algorithm where we don’t have to make assumptions about linearity in comparison to algorithms like logistic regression. It can be implemented on linear and non-linear data both. KNN offers a variety of distance metrics to choose from based on the requirements of the model. It has one parameter which is k that needs to be hyper tuned for optimizing the performance of the model. The imbalance dataset was handled by ADASYN so that KNN works well on the dataset for the stress label binary classification. Underfitting occurs when the k value is too large and decision boundaries are not able to understand complex patterns. High bias leads to underfitting. The model will not be satisfactory on training as well as on test or unseen data. Overfitting occurs when the k value is too small, and variance is high. The model tends to learn noise from the data. Due to which model performs well on training data but does not generalize properly on test or unseen data. The optimal value of k can solve the problem of underfitting and overfitting. Cross validation and GridSearch can be used to test models with different sets of data and arrive at a optimal k value. This is done to arrive at a balance between bias and variance which will lead to optimized performance of the model.

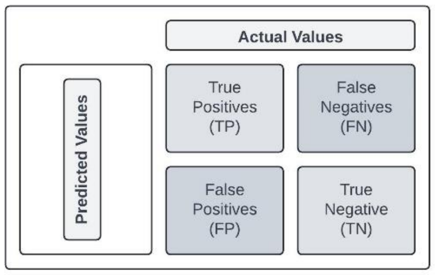
Model Evaluation Method

Confusion Matrix

A confusion matrix described in Figure 58 is a tool which is used for accessing the performance of classification models by displaying the counts of true positives, false positives, true negatives and false negatives.

Figure 58

Confusion matrix



True Positives (TP)

The case where the actual class is positive, and the model also predicts it as positive. It is crucial for understanding how well a model is identifying the positive outcomes.

True Negatives (TN)

The case where the model correctly predicts the negative class. That is the actual class is negative and the predicted class is also negative.

False Positives (FP)

The case where the model incorrectly predicts the positive class. That is the actual class is negative, but the model wrongly predicts it as positive.

False Negatives (FN)

The case where the model incorrectly predicts the negative class. That is the actual class is positive, but the model fails to recognize it as negative.

Accuracy

Accuracy measures the correctness of predictions made by a classification model. It is represented as the ratio of true positive in addition to true negative to all the predictions made by the machine learning model as shown in Equation 3

$$\text{Accuracy} = \frac{TP+TN}{TP+TN++FP+FN} \tag{3}$$

Recall

Recall, a performance indicator generally used with binary classification tasks, assesses how many of the actual positive cases (True Positives) are identified by a model among all actual positive cases. This is also known as accuracy rate, or sensitivity if you will. It is calculated by using the number of true positives divided by the total of false negatives and true positives.
A higher value of the recall means that the model is finding most positives at the same time reducing number of false negatives. t is especially crucial in situations when there could be serious repercussions from overlooking positive cases (false negatives) as shown in Equation 4.

$$\text{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \tag{4}$$

F1-Score

F1 score is the harmonic means of precision and recall. It is used to compare model's performance and is more reliable than accuracy. Equation 5 shows the F1 score formula.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Specificity (True Negative Rate)

Specificity is the proportion of negative values to the total number of values. Real negative rate is also known as specificity. Moreover, there are several real-life settings of the need for specificity, such as in medical diagnostics, in which case, finding negative values is extremely important since mistakes here are unacceptable as shown in Equation 6.

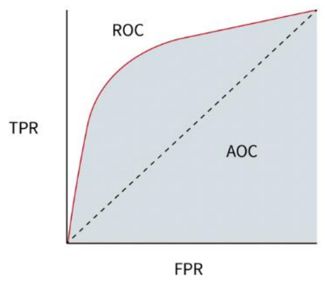
$$\text{Specificity} = \frac{TN}{TN + FP} \tag{6}$$

Receiver Operating Curve (ROC) and Area Under the Curve (AUC)

AUC and ROC are another evaluation metrices for Classification algorithm. Receiver Operating Curve or ROC gives the probability measure by plotting True Positive Rate (TPR) against False Positive Rate (FPR). Area under this Curve or AUC represents the separability between the predicted classes. Higher AUC indicates that the model classified the binary stress Labels well. Figure 59 shows the generic structure of the ROC and AUC.

Figure 59

Generic ROC and AUC plot



Log Loss

Log loss is also known as cross entropy loss. This is defined as the measure of difference between predicted results and actual expected results as shown in Equation 2. This is often used for imbalanced classification. The output is the probability of predicted label which will be between 0 and 1. The log loss increases as the predicted probability moves away from the actual label (Maros et al., 2021). Entropy refers to the measure of uncertainty. It means that a decrease in cross entropy loss will cause an increase in prediction accuracy of the model. Log loss close to 0 indicates the most accurate predictions made by the model in Equation 7.

$$H_P(q) = -\frac{1}{N} {}_{i=1}^{N}\Sigma\, y_i.\log\big(p(y_i)\big) + (1 - y_i).\log\,(1 - p(y_i)) \tag{7}$$

.

In this, y is the label (0 for stress label means absent and 1 means stress present) and p(y) is the predicted probability for the stress label.

Brier Score Loss

Brier Score loss calculates the mean squared difference between predicted outcome and the actual label. Brier loss is widely used for evaluating the performance of binary classification model. This is suitable for imbalanced classification of label. It assesses the probability of true cases as shown in Equation 8.

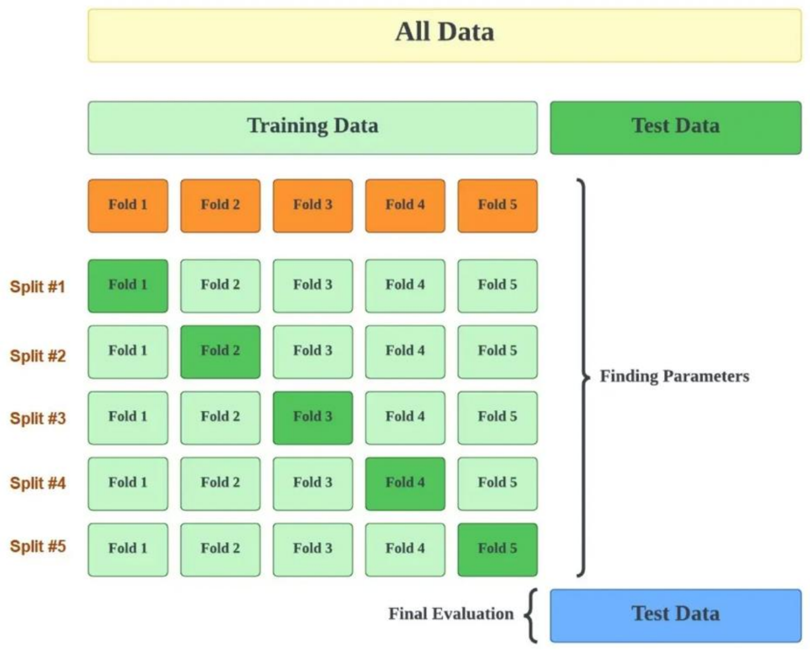
$$\boldsymbol{BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2} \tag{8}$$

Cross Validation and GridSearchCV

K- fold cross validation technique is an approach where dataset is randomly shuffled and split into k groups. Based on the book by James Gareth et al. (2013) , among the k folds with equal split size, the first fold is used as validation set and rest of the k-1 folds are used to train the model. In GridSearchCV cross validation is applied with grid of parameters and estimators. Figure 60 demonstrated the cross-validation technique.

Figure 60

k-Fold Cross Validation



Model Validation and Evaluation

All the implemented KNN models were evaluated on validation and testing dataset. We have two categories of KNN models where one is baseline model and other is hypertuned model for performance optimization.

K-Nearest Neighbors (K-NN)

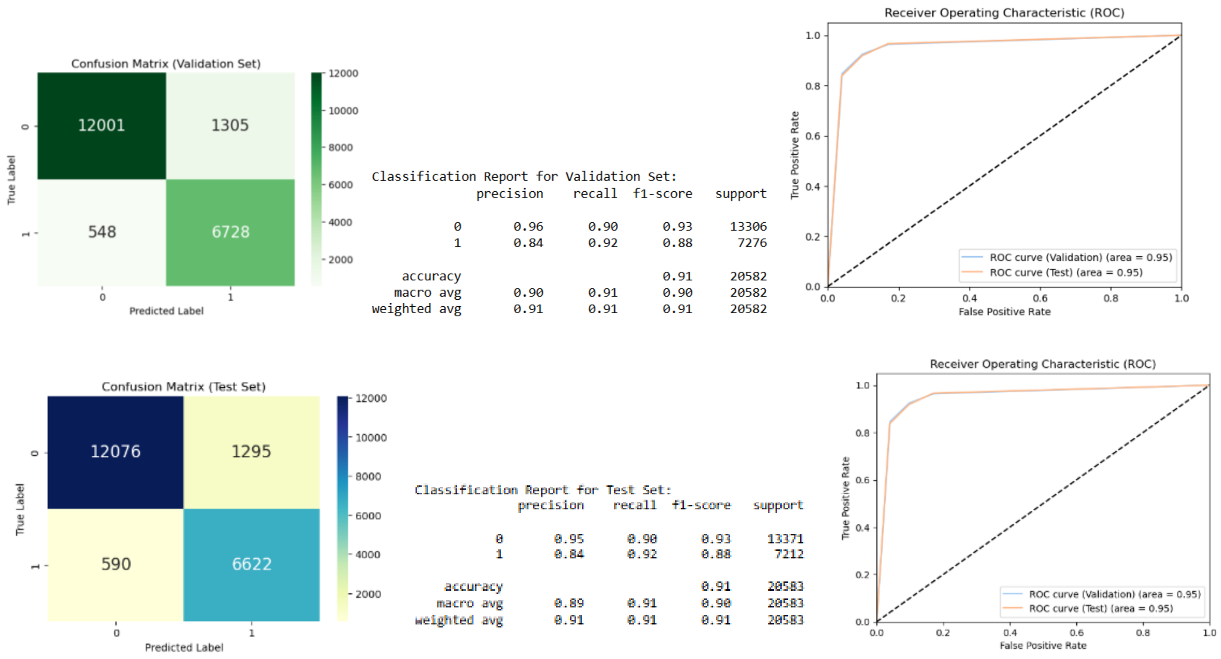
Baseline Model

The Baseline model for K-nearest neighbor implemented through KNeighborsClassifier from scikit-learn library in python. The model was created by default parameters such Euclidean distance metric and number of nearest neighbors as 3. The data was standardized through standard scalar. The K-

NN model was trained by including all the 9 features. The log loss was recorded as 3.24 with accuracy of 0.90 and AUC of 0.95 for validation set. The training time for model was 0.19 seconds. The performance for the testing set was evaluated as accuracy of 0.90, log loss of 3.3 and AUC of 0.95 as shown in Figure 61.

Figure 61

Represents the performance evaluation of K-NN baseline model for validation and testing set



```
Classification Report for Validation Set:
              precision    recall  f1-score   support

           0       0.96      0.90      0.93     13306
           1       0.84      0.92      0.88      7276

    accuracy                           0.91     20582
   macro avg       0.90      0.91      0.90     20582
weighted avg       0.91      0.91      0.91     20582
```

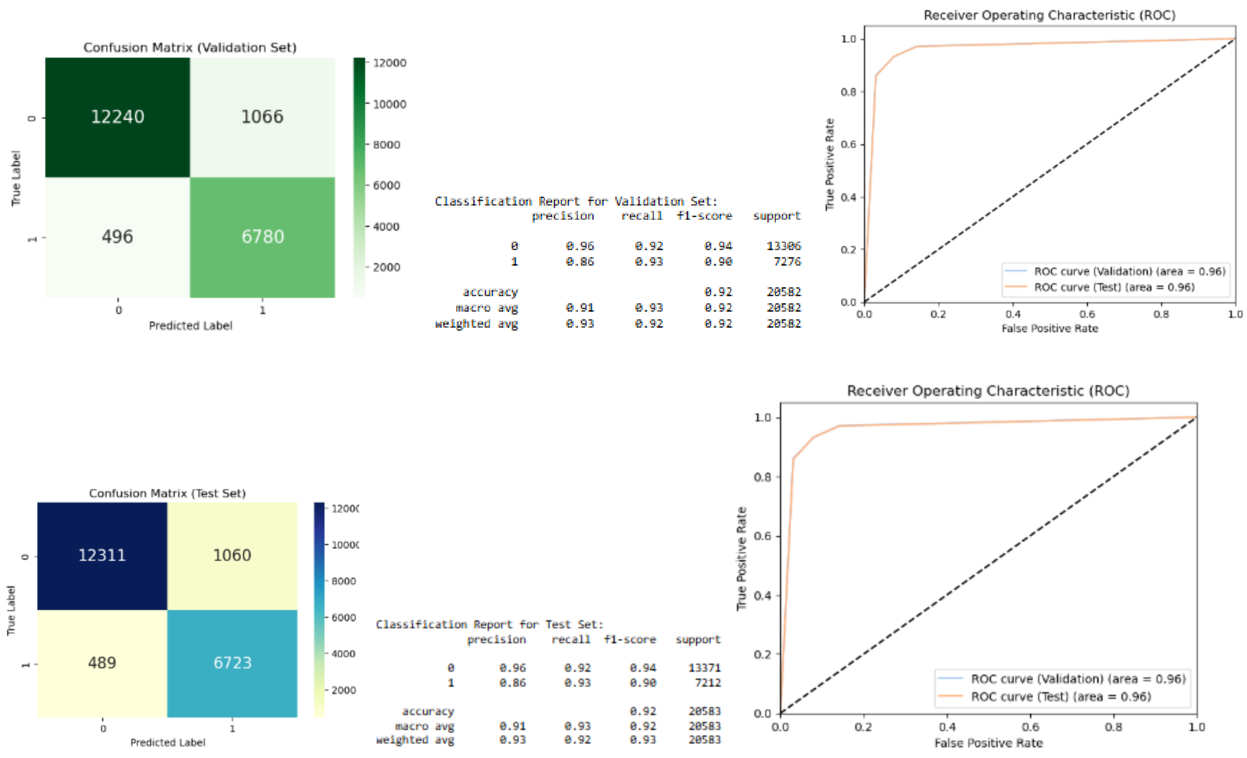
```
Classification Report for Test Set:
              precision    recall  f1-score   support

           0       0.95      0.90      0.93     13371
           1       0.84      0.92      0.88      7212

    accuracy                           0.91     20583
   macro avg       0.89      0.91      0.90     20583
weighted avg       0.91      0.91      0.91     20583
```

Hyperparameter tuned Model

Hyperparameter tuning was used to optimize the performance of the model by finding the best parameters. GridSearchCV from scikit-learn library was employed for tuning the model. K- fold cross validation equal to 5 was used. The model was trained on a parameter grid which has values of k as 3, 5, 7, 9 and 11. Manhattan distance metric gave the best performance among all other distance metrics. The training time taken by the model was 0.18 seconds. The tuned K-NN model achieved an accuracy of 0.92, log loss of 2.73 and AUC of 0.96 for validation set. The training or unseen set returned a log loss of 2.71, an accuracy of 0.92 and AUC of 0.96 as shown in Figure 62. It can be clearly seen that with hyperparameter tuning we have achieved a 2% increase in accuracy along with a decrease in log loss.

Figure 62

Represents the performance evaluation of K-NN hyperparameter tuned model for validation and testing set.



```
Classification Report for Validation Set:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94     13306
           1       0.86      0.93      0.90      7276

    accuracy                           0.92     20582
   macro avg       0.91      0.93      0.92     20582
weighted avg       0.93      0.92      0.92     20582
```

```
Classification Report for Test Set:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94     13371
           1       0.86      0.93      0.90      7212

    accuracy                           0.92     20583
   macro avg       0.91      0.93      0.92     20583
weighted avg       0.93      0.92      0.93     20583
```

Comparison of k-nearest neighbor (KNN) models' performance

The hyperparameters, accuracy, log loss and training time were compared as shown in Table 7. The number of best k neighbors across models were 3 as provided through GridSearchCV. The KNN model with Manhattan distance metric took training time as 0.18 seconds and outperformed others with 0.92 accuracy. The lowest performing model was implemented with Chebyshev distance metric took the highest training time as 0.26 and got an accuracy of 0.88. The cosine similarity took the least training time as 0.03 seconds, but performance was close to lowest of all the models with accuracy of 0.89. The baseline model was implemented with the default Euclidean distance metric took 0.19 seconds for training the model and accuracy was 0.90.

Table 7

Represents comparison of KNN models

Classifier	Training Time (in Sec)	Validation Log Loss	Validation Accuracy	Testing Log Loss	Testing Accuracy
K-Nearest Neighbor (K-NN) - Base Model	0.1943	3.245	0.9099	3.3008	0.90841
K-Nearest Neighbor (K-NN) – Manhattan metric Hypertuned Model	0.1809	2.7354	0.9241	2.7125	0.9247
K-Nearest Neighbor (K-NN) – Minkowski metric Hypertuned Model	0.2186	3.537	0.9018	3.516	0.9024
K-Nearest Neighbor (K-NN) – Chebyshev metric Hypertuned Model	0.2646	4.306	0.8805	4.272	0.8814
K-Nearest Neighbor (K-NN) – Cosine Similarity metric Hypertuned Model	0.0304	3.544	0.9016	3.612	0.8997

Comparison of various models' performance

All the implemented models are compared based on the execution time to build the models and the accuracy score. Table shows the comparison results for all implemented models for this project. Although Random Forest model achieved highest accuracy of 96% for both validation and test data, execution time to train the model is longer than any other models. Howver, kNN model execution time is the shortest with good accuracy for both validation and test data around 92%. Also, XGBoost base model with log-loss metric performed better with around 91% accuracy. So, both kNN and XGBoost models performed well in predicting Stress labels for the current project. Table 8 represents the model comparisons.

Table 8

Represents model comparison.

Classifier	**Training Time (in Sec)**	**Validation Log Loss**	**Validation Accuracy**	**Testing Log Loss**	**Testing Accuracy**
Logistic Regression Base Model	0.7686	0.62	0.6609	0.62	0.6617
Logistic Regression-GridSearchCv	295.9740	0.6287	0.6611	0.6278	0.6613
Logistic Regresson (Feature importance)	0.1811	0.6288	0.65732	0.6281	0.6587
K-Nearest Neighbor (K-NN) - Base Model	0.1943	3.245	0.9099	3.3008	0.90841
K-Nearest Neighbor (K-	0.1809	2.7354	0.9241	2.7125	0.9247

NN) - Hypertuned Model					
Random forest -Base Model	101.593	0.154	0.9547	0.156	0.9532
Random forest – After ADASYN	162.163	0.178	0.9433	0.179	0.9448
Random forest -Hyper tuned	7108.51058	0.12205292	0.96380332	0.12397	0.96302
XGB Base Model[metric-auc]	14.1581847667	3.83693	0.89354776	3.77895	0.895156
XGB Base Model [metric-log-loss]	11.899408340	3.12243	0.91337090	3.03647	0.915755
XGB - GridSearchCV [score- auc-roc]	71.85246968	3.19746	0.79943640	2.67958	0.831920
XGB - GridSearchCV [score - log-loss]	65.03023624	3.92479	0.753814012	3.70517	0.767589
DNN	385		0.9203		0.92

Limitations and Future Scope

K-nearest neighbor algorithm requires large memory as complete training data is stored in the memory. KNN works well on a smaller number of features. This is affected by curse of dimensionality which means with the increase in higher number of dimensions model learns noise and irrelevant patterns which can lead to overfitting. The distance metric such as Euclidean used are dependent on magnitude hence features with higher magnitude will tend to weigh more than features with lower magnitude. The KNN model has limitations on time and memory requirements.

Along with DNN, other deep learning models can be implemented to extract hidden and meaningful information about the features which will result in improved model's performance. Federated learning can be utilized to address user's privacy and security concerns. The data used in this project is taken from wearables from the users which might have some sensitive and private information related to the user. Such information should be dealt with carefully.

References

Amin, M. Rafiul, Wickramasuriya, D. S., & Faghih, R. T. (2022). A Wearable Exam Stress Dataset for Predicting Grades using Physiological Signals. IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT), Houston, TX, USA, 2022, pp. 30-36, doi: 10.1109/HI-POCT54491.2022.9744065. https://ieeexplore.ieee.org/abstract/document/9744065

Bentéjac, C., Csörgő, A. & Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms.

Artif Intell Rev 54, 1937–1967 (2021).

https://doi.org/10.1007/s10462-020-09896-5

Bobade, P., & Vani, M. (2020). Stress Detection with Machine Learning and Deep Learning using Multimodal Physiological Data. Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, pp. 51-57, doi: 10.1109/ICIRCA48905.2020.9183244. https://ieeexplore.ieee.org/abstract/document/9183244

Campanella, S., Altaleb, A., Belli, A., Pierleoni, P., & Palma, L. (2023). A Method for Stress Detection Using Empatica E4 Bracelet and Machine-Learning Techniques. *Sensors*, *23*, 3565. https://doi.org/10.3390/s23073565.

Çöpürkaya, Ç., Meriç, E., Erik, E. B., Kocaçınar, B., Akbulut, F. P., & Catal, C. (2023). Investigating the effects of stress on achievement: BIOSTRESS dataset. Data in brief, 49, 109297. https://doi.org/10.1016/j.dib.2023.109297

Fauzi, M. A., Yang, B., & Blobel, B. (2022). Comparative analysis between individual, centralized, and federated learning for smartwatch-based stress detection. Journal of Personalized Medicine.12(10):1584. doi: 10.3390/jpm12101584.

https://pubmed.ncbi.nlm.nih.gov/36294724/

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29(5), 1189–1232. http://www.jstor.org/stable/2699986

Iqbal, T., Simpkin, A.J., Roshan, D., Glynn, N., Killilea, J., Walsh, J., Molloy, G., Ganly, S., Ryman, H., Coen, E., Elahi, A., Wijns, W., & Shahzad, A. (2022). Stress Monitoring

Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328, doi: 10.1109/IJCNN.2008.4633969. https://ieeexplore.ieee.org/document/4633969

Rogers Melissa J. B., Hrovat Kenneth, McPherson Kevin, Moskowitz Milton E.,Reckart Timothy.(1997).Accelerometer Data Analysis and Presentation Techniques. https://ntrs.nasa.gov/api/citations/19970034695/downloads/19970034695.pdf

Koelstra, S., Muhl, C., Soleymani, M., Lee, J. S., Yazdani, A., Ebrahimi, T., Pun, T., Nijholt, A., & Patras, I. (2012). DEAP: A Database for Emotion Analysis; Using Physiological Signals, in IEEE Transactions on Affective Computing, vol. 3, no. 1, pp. 18-31, Jan.-March 2012, doi: 10.1109/T-AFFC.2011.15. https://ieeexplore.ieee.org/document/5871728.

Priya, A., Garg, S., & Tigga, N. P. (2020). Predicting Anxiety, Depression and Stress in Modern Life using Machine Learning Algorithms, Procedia Computer Science, Volume 167, Pages 1258-1267, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2020.03.442.

Maros, M.E., Cho, C.G., Junge, A.G. (2021). Comparative analysis of machine learning algorithms computer-assisted reporting based on fully automated cross-lingual RadLex mappings. Sci Rep 11, 5529 (2021). https://doi.org/10.1038/s41598-021-85016-9 https://www.nature.com/articles/s41598-021-85016-9

Sun, S., & Huang R. (2010). An adaptive k-nearest neighbor algorithm, 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, Yantai, China, 2010, pp. 91-94, doi: 10.1109/FSKD.2010.5569740 https://ieeexplore-ieee-org.libaccess.sjlibrary.org/document/5569740

Gupta A., Chakroborty S., Ghosh S., Ganguly S. (2023). A machine learning model for multi-class classification of quenched and partitioned steel microstructure type by the k-nearest neighbor algorithm, Computational Materials Science, Volume 228, 2023, 112321, ISSN 0927-0256, https://doi.org/10.1016/j.commatsci.2023.112321 https://www-sciencedirectcom.libaccess.sjlibrary.org/science/article/pii/S0927025623003154

Appendix

GitHub is used for handling source code developed as part of this project implementation and to maintain the code. Python Jupyter notebook is uploaded. GitHub Link-

https://github.com/shikha284/Cognitive-Stress-Level-Prediction-Model

Dataset is uploaded on google drive at https://drive.google.com/drive/folders/107rTSTAnnIWXBHxWSvtyiLd7V-g9Pfs3?usp=sharing

Pickle file is uploaded on google drive at

https://drive.google.com/drive/folders/107rTSTAnnIWXBHxWSvtyiLd7V-g9Pfs3?usp=sharing