



**VIT<sup>®</sup>**

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

ECE3502: IOT DOMAIN ANALYST

# Stress Level Detection using IoT and machine learning

SUBMITTED TO

Prof. Biswajit Dwivedy

**TEAM MEMBERS:**

1. TANYA GUPTA - 20BEC0740
2. DEEPIKA PAVUNDOSS - 20BEC0285
3. ISHITA NARANG - 20BEC0180

## **Index**

<i>Contents</i>	<i>Page No.</i>
Abstract	3
Introduction	4
Objective	5
Literature Survey	5-6
System Model	7
Circuit diagram	7
Methodology	8-17
Result and Discussion	17
Acknowledgement	20
References	21

## Abstract

According to a cardiac surgeon, it is difficult to predict age from heart rate as it is nonlinear, but we can use a person's heartbeat to predict whether that person is fit, unfit and overtrained or not, provided we have that person's age. Based on heart beat we can predict whether a person is in Stress or not. Stress is one of the main factors that are affecting millions of lives. Thus, it is important to inform the person about his unhealthy lifestyle and even alarm him/her before any acute condition occurs.

To detect the stress beforehand we have used heartbeat rate, spO2, body temperature, limb movement and sleeping hours as the parameters.

Based on these attribute values we have trained and tested our ML model using various algorithms like Random Forest, Decision Tree, Logistic Regression and Support Vector Machine to get the best accuracy for our dataset.

We obtained and created a test dataset from the sensor values and tested our model for real time data. We have used appropriate algorithms to get increased accuracy and predict the stress level.

Internet of Things (IoT) along with Machine Learning (ML) is used to alarm the situation when the person is in real risk. ML is used to predict the condition of the patient and IoT is used to communicate the patient about his/her acute stress condition.

## Introduction

The stress is one of the main symptoms in our human life. Stress conducts a major role in life of people. Stress may lead many diseases like cardio-vascular disease, lungs problems,

breathing problem, cancer, and other diseases. Due to the population in the global the stress level among the people increased. Nowadays stress is a common symptom in all human being.

The stress conducts major life-or-death problems in a people around the world. When a people are suffering from stress their normal behavior will be changed and which leads to some life

routine problems. For this we have to continuous monitoring their behavior through using machine learning algorithms which are implemented in the system from that we will get the

real time data for analyzing their behavior accurately of the system. In India 85% of people are suffering from stress. Nowadays due to the stress the person will undergo depression.

The stress detector system will differentiate the difference between normal behavior person and stressed person. The physiological devices have been used for the continuous

monitoring of the stressed person and it will help the parametric values for the doctors to predict whether the person is normal or stressed. The parallel the physiological devices monitor the

behavior of the person and stores in a database and these data will be maintained in a database for the further analysis by the doctors. Based on those values the doctors easily predict the

stress level in the person. So, these analyses get whether the person in stress or not stress. The proposed algorithm and devices are used for predicting depression level in India.

Through mapping of stress and heartbeat, we can identify a lot of things, for example whether the person is nervous or not, whether the person is in apprehension or fear, whether the

person is working out, whether the person is over trained, remote monitoring of a patient with heart disease etc. Additionally, when we are training, our heartbeat rate should be elevated, especially if a person is doing any cardio-exercise and is dependent upon a person's.

age. Since exercising has become a part of most people's daily regime, it is crucial to identify whether he/she is exercising correctly or not.

## Objective

The medical world has seen strong correlations between stress and heart disease, cancer, and such terminal illnesses. Further stress has been shown to weaken immune systems, as well as drop performance in all metrics of success. Stress cannot be quantified and is exceedingly difficult to detect. The aim of this project is to predict stress levels of a person using a physical device to reduce and lower the depression rate in India

## Literature Survey

The idea of stress detection system will help the doctors to predict the stress level using machine learning algorithms. The researchers are proposing their work related to person stress detection.

### **1.Detection and Analysis of Stress using Machine Learning Techniques**

Reshma Radheshamjee et. al., proposed a system as in this project the datasets are collected from social-media. Nowadays when person is in stress or depression, they will post the quotes or any other images to face book, twitter, or any other social media. In twitter so many post the quotes or any other wordings in discussion form or as a post. Based on the twitter dataset they will collect the data, they have proposed the system used the support vector machine and Naïve Bayes algorithms. Based on the algorithm that predict the detection the person is in stress or depression. In twitter dataset they have consider stress, depression etc. They have used the sentimental analysis to classify the depression and stress. When they use more techniques or algorithms, we can get the best results and also it shows the precision and recall values. They have used the confusion matrix to predict the stress and depression.

Limitations: In this proposed system due to the twitter dataset, they will not get accurate accuracy.

### **2.Design and Development of an IOT based Wearable device for the safety and security of women and girl children**

Alisha RM et. al., proposed a system as safety measures have taken to the girls. Nowadays the women and girls not having safety. In a world nowadays we are seeing the girl or women get raped in some of the places. Based on these problems so many girls are suffering. Based on this proposed system we are use the devices based on these values we can get the real time values. They have used accelerometer, body temperature and Galvanic skin response. This is a wearable device the girl should wear this device whenever the girl goes outside. The device will store all the data in database. Based on these values we have

some idea about the values when girl is in risk, when she will run her body temperature and she will sweat more. Based on these values we can predict the girl is in danger when the values above threshold value or less it will give alert message with respective person. Based on this message the police or other person can save her life.

Limitations: In this proposed system they have used the some of the devices to get appropriate values. Due to the devices, they will not get the accurate values.

### **3.Wearable Glove-Type Driver Stress Detection Using a Motion Sensor**

Wan-young Chung et. al., proposed a system as stress detection by drivers.

Nowadays we hear so many news as the road accidents. The driver will die due to the accidents. The driver will die in accidents we do not know what the problem in that is. When the driver will sleep, or he is having some problem due to that he will get accident. The proposed system as he has used the physiological devices were motion sensor, Galvanic. skin response and body temperature based on these devices it will collect the data and that data should be stored in database.

When the driver will get the sleeping, or he is not well based on these values we can predict the driver is having some problem we can alert the driver. Based on the data we can alert the driver as ringing the sound.

Limitations: In these proposed system they have used the physiological devices where they get some problem in calculating the values.

### **4.Machine Learning and IoT for Prediction and Detection of Stress**

Purnendu Shekar Pandey et. al., proposed a system as stress detection is based on the machine learning. When we predict the stress based on the heart rate and electrocardiogram values.

Limitations: In this proposed system they will predict the stress based on the heart rate and ECG.

### **5.A smart sensor in the IoMT for stress level detection**

Madhavi Ganapathiraju et. al., proposed a system in which monitoring of the stress by physiological devices. When we use physiological devices, we can get the accurate values.

When we see any person from the outside, we don't know the person is in healthy or not. Due to this problem, we can easily predict the person is having any disease or stress. In these physiological devices are Galvanic skin response, body temperature, pulse rate and their motion sensor are used. When we use the physiological device, it will store the values based on these values when the values will change, we easily get the person is having some problem.

Limitations: In these proposed system the data are collected from the physiological devices and calculate the values.

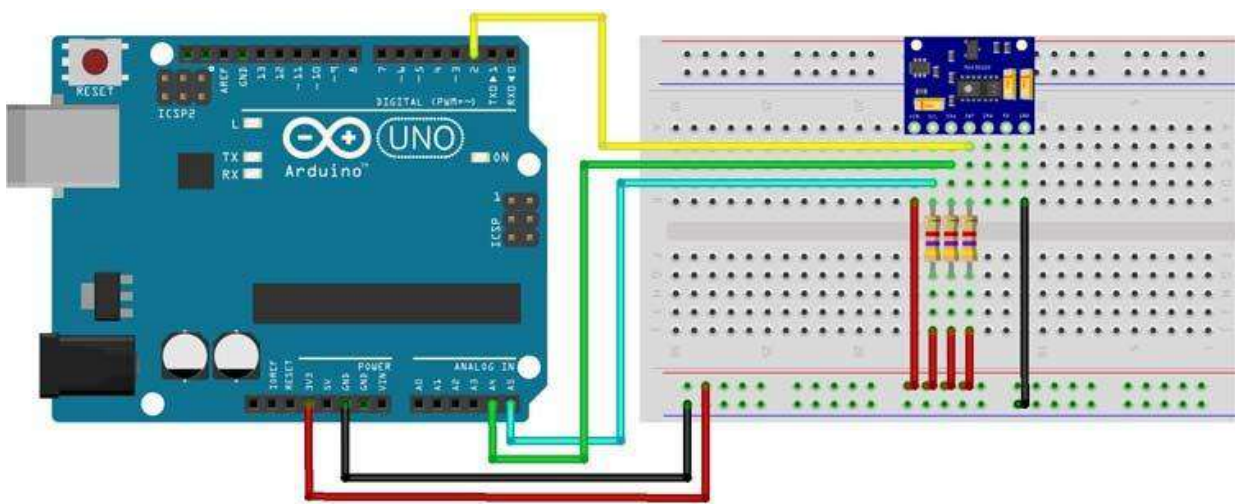
## System Model

The components required for this project model are:

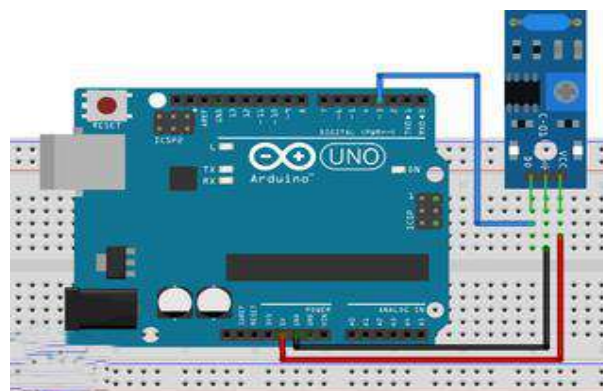
1. Arduino Uno board
2. MAX3010 Pulse and SPo2 sensor
3. SW-420 Vibration sensor module
4. puTTY
5. ThingSpeak
6. Jupyter Notebook
7. Streamlit

## Circuit Diagram

### Connections for SPo2 sensor



### Connections for Vibration sensor SW-420 Module



## Methodology

### A. Hardware Setup

1. We will first setup the hardware as per the connections in the circuit diagram to take in sensor data.
2. MAX3010 sensor is connected to the Arduino to record the Heart rate (in bpm) and the spO2 levels.
3. Vibration sensor is used to record the limb movement, the process we employed to calculate the limb movement is that we recorded the number of vibrations in a certain interval of time ( for our case 10s ).
4. We used ESP8266 to send limb movement data to ThingSpeak cloud.
5. Code was written on Arduino IDE

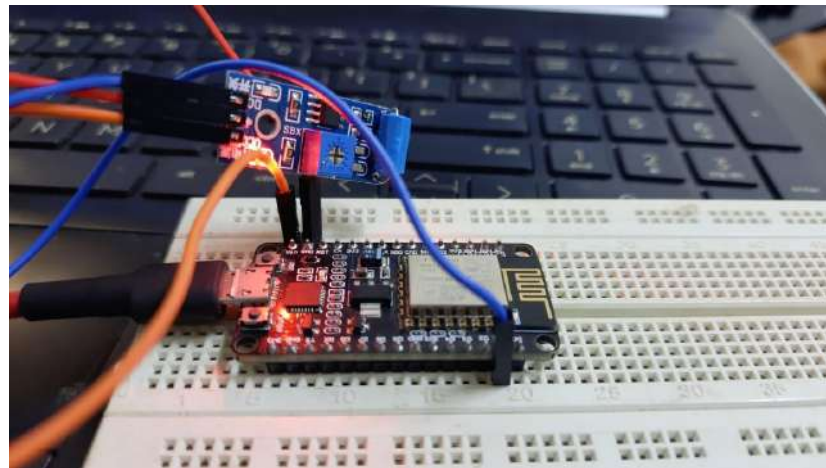


Figure 1: Connections for the Vibration sensor module

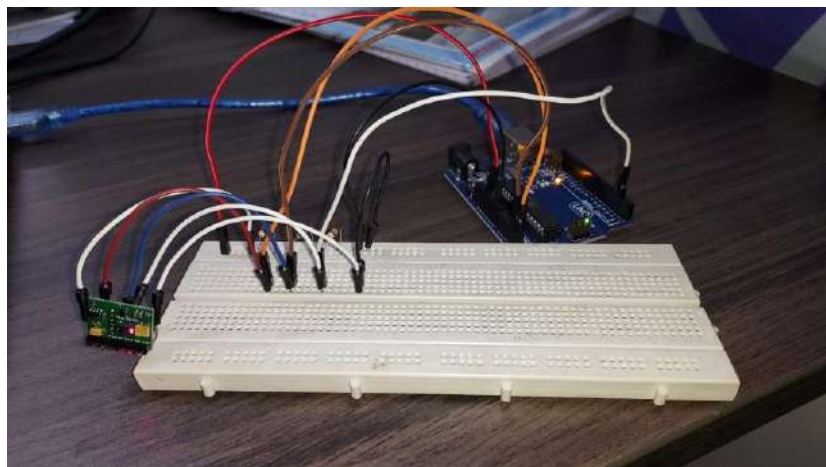


Figure 2: Connections for the MAX3010 sensor





## CODE for Vibration Sensor for Obtaining Limb Movement

```
#include<ESP8266WiFi.h>
#include<DHT.h>
#include<WiFiClient.h>
#include<ThingSpeak.h>

const char* ssid="deepika";//Your network SSID
const char* password="Deepi910";//Your Network password
WiFiClient client;
unsigned long myChannelNumber=2086175; // Channel code
const char* myWriteAPIKey="G4U62NUEIVLKRU17";//Your API key

int vib_pin=D1; // D1
int timer = 0;
int lm = 0;
void setup() {
  pinMode(vib_pin,INPUT);
  Serial.begin(9600);
  //Connect to WIFI Network
  WiFi.begin(ssid,password);
  ThingSpeak.begin(client);
}

void loop() {
  if(timer>200){
    Serial.println();
    if(lm>20){
      lm = 20;
    }
    Serial.println(lm);
    ThingSpeak.writeField(myChannelNumber, 1,lm,myWriteAPIKey);
    timer = 0;
    lm = 0;
  }
  else{
    timer++;
    delay(50);
    int val=digitalRead(vib_pin);
    if(val==0){
      lm++;
    }
    Serial.print(val);
  }
}
```



Figure 5: ThingSpeak graph plot for Limb movement

```

heart_rate
#include <LiquidCrystal.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS 1000

PulseOximeter pox;
uint32_t tsLastReport = 0;

void onBeatDetected()
{
    Serial.println("Beat!");
}

void setup()
{
    Serial.begin(115200);
    Serial.print("Initializing pulse oximeter..");

    Serial.print("Initializing...");
    delay(3000);

    // Initialize the PulseOximeter instance
    // Failures are generally due to an improper I2C wiring, missing power supply
}

Done uploading.
Compiling core...
Using precompiled core: C:\Users\ishit\AppData\Local\Temp\arduino_cache_985278\core\core_arduino_avr_uno_0c812875ac70eb4a5b385d8fb077f54c.a
Linking everything together...
"C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avr-gcc" -W -Os -g -fno-fuse-linker-plugin -Wl,--gc-sections -mmcu=atmega328p -o "C:\Users\ishit\AppData\Local\Temp\arduino_cache_985278\sketch_115200_0\sketch_115200_0.ino"

```

Figure 6: MAX3010 Arduino Code

```

COM6
Send

Heart rate:81.37
bpm / SpO2:97%

Beat!
Heart rate:37.60
bpm / SpO2:97%

Beat!
Heart rate:61.36
bpm / SpO2:96%

Heart rate:61.36
bpm / SpO2:96%

Beat!
Heart rate:38.02

Autoscroll Show timestamp Newline 115200 baud Clear output

```

Figure 7: Serial Monitor output for MAX3010

## CODE for MAX3010 Sensor

```
#include <LiquidCrystal.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS 1000

PulseOximeter pox;
uint32_t tsLastReport = 0;

void onBeatDetected()
{
    Serial.println("Beat!");
}

void setup()
{
    Serial.begin(115200);
    Serial.print("Initializing pulse oximeter..");

    Serial.print("Initializing...");
    delay(3000);

    // Initialize the PulseOximeter instance
    // Failures are generally due to an improper I2C wiring, missing power supply
    // or wrong target chip
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }
    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

    // Register a callback for the beat detection
    pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop()
{
    // Make sure to call update as fast as possible
    pox.update();
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart rate:");
        Serial.print(pox.getHeartRate());
        Serial.println();
        Serial.print("bpm / SpO2:");
        Serial.print(pox.getSpO2());
        Serial.println("%");
        Serial.println();
        tsLastReport = millis();
    }
}
```

### c. Preparing the Test Dataset by loading sensor values

1. We employed two different methods for loading sensor values in a csv.
2. In case of vibration sensor module, we uploaded the data to ThingSpeak.
3. and imported the csv from there and hence we recorded the limb movement data.
4. We used putty to load MAX3010 sensor values using serial communication.
5. PuTTY is a tool for logging sensor values from a specified serial port at a particular baud rate and stores it in a printable output say csv file.

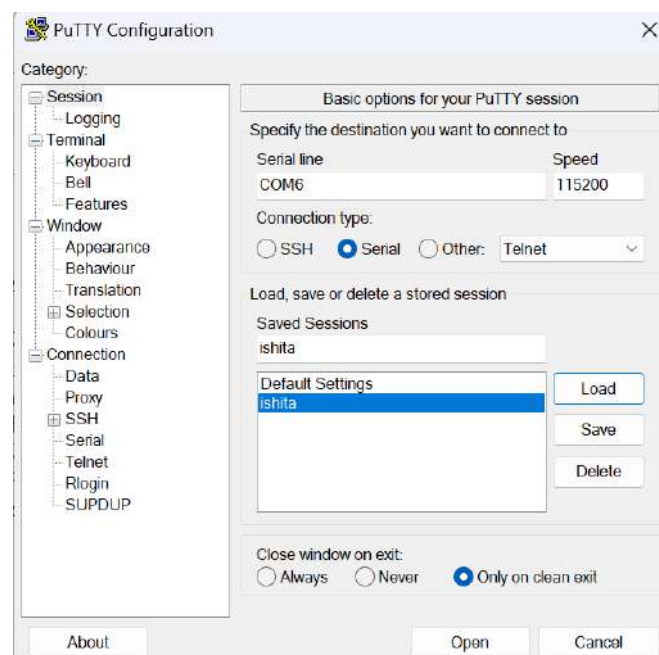


Figure 8: puTTY configuration settings



Figure 9: puTTY session output

A11	
	A
1	== PuTTY log 2023.04.04 18:30:59 ==
2	Initializing pulse ox rate:0.00bpm / SpO2:0%
3	Heart rate:0.00bpm / SpO2:0%
4	Initializing pulse oximeter..Initializing...SUCCESS
5	Heart rate:88.42bpm / SpO2:96%
6	Heart rate:66.63bpm / SpO2:95%
7	Heart rate:64.88bpm / SpO2:98%

Figure 10: CSV file log of MAX3010 sensor data

## D. Training the Machine Learning Model

1. Missing Values: No missing value was detected.
2. Duplicates: No duplicate was detected
3. Feature Extraction: Correlation among different features were studied for the same. The following have been concluded on visualizing the data and finding the correlation matrix.
4. Stress Levels and Sleeping Hours are strongly negatively correlated. The more people sleep, the less they are likely to be stressed. negative correlation between Stress (categorical variable) and Number of Hours Slept (continuous variable).
  - i. People who sleep 8 hours on average in a single day report no or very low stress levels. It is detected that people with high limb movement have more stress.
  - ii. Body temperature and stress level are highly negatively correlated.
  - iii. Stressed people have higher heart rate, people with low or no stress have heart rate between 50-60
5. Outliers detection:
  - i. The seaborn plot for each of the feature has been visualized to detect any outlier present in each of the feature – heart rate and limb movement.
  - ii. IQR method was used for limb movement feature
  - iii. Outlier detection using box method for heart rate, body temperature, blood oxygen and sleeping hours.
  - iv. No outlier was actually detected
6. Data standardization: Standard Scaler is applied to the independent features of the dataset the bring normalize the feature values
7. Training the model: Out of the different Supervised Classification ML models used such as SVM, Decision tree regression, Random Forest Regression and Logistic Regression, it was found that Random Forest Regression gives the best results.

#### E. Imputation of test Data and testing it's accuracy

1. A copy of training dataset is created for imputation purposes.
2. Rearranging the columns in dataset to ease Imputation: The Features to be Imputed are shifted to the end as they will be the dependent variables which are imputed using the other features.

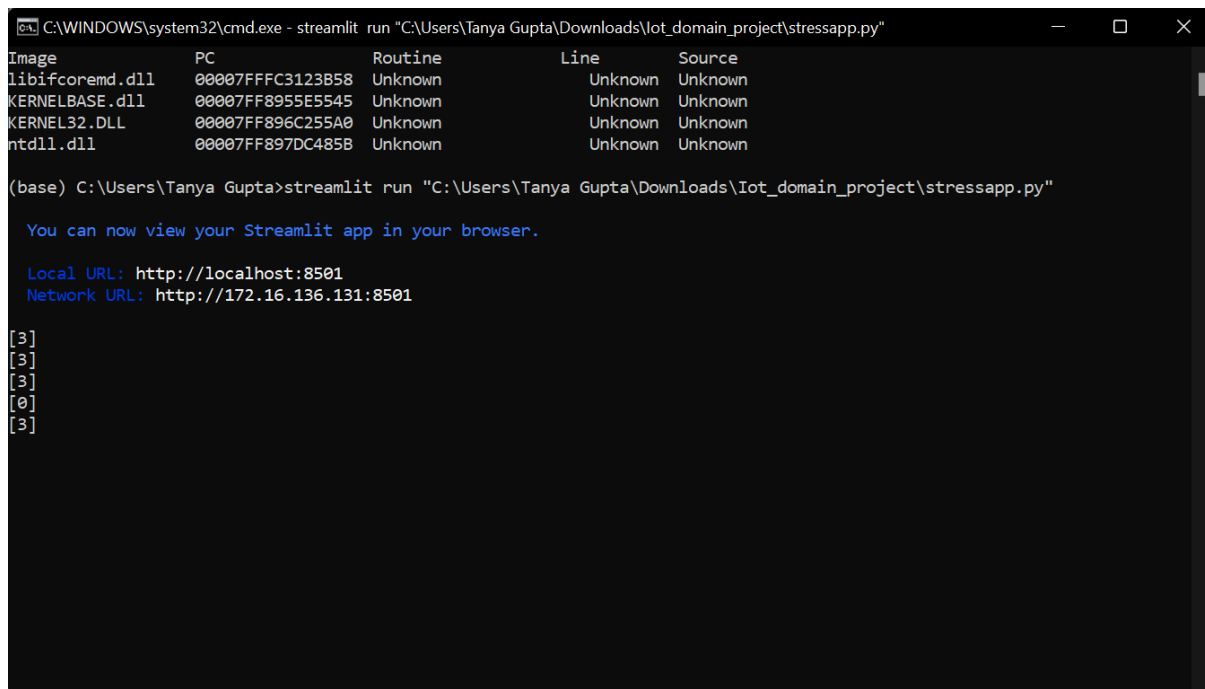
Algorithms tried for Imputation:

- Mice Algorithm
- Mean/Median
- Random Forest Regression
- Decision Tree Regression
- Polynomial Regression
- Support Vector Regression

The best final accuracy of Stress levels was provided by Decision Tree Regression Algorithm. Other methods gave accuracy of only about 24%. Mean/Median methods are only suitable when there are only a few missing values in a particular feature and not when all the values of the feature are supposed to be imputed.

3. We tried inputting other combinations of features. We also did try including the collected sensor values in testing data and imputing other features like Sleeping hours and preparing the dataset, but since features like sleeping hours wasn't that correlated with the other features, the imputed values weren't good enough, so the final testing dataset had a very low accuracy of about 0.32 or 32%
4. The imputed features (Body temperature and blood Oxygen) are combined with the testing data.
5. Standard scaler: It was applied on training dataset hence is applied on the testing data.
6. Prediction: The stress level values for the test set were predicted and the accuracy score on testing dataset was obtained to be 64%

## F. Web Interfacing and GUI



```
C:\WINDOWS\system32\cmd.exe - streamlit run "C:\Users\Tanya Gupta\Downloads\lot_domain_project\stressapp.py"

Image      PC          Routine      Line      Source
libifcoremd.dll  00007FFFC3123B58 Unknown      Unknown    Unknown
KERNELBASE.dll  00007FF8955E5545 Unknown      Unknown    Unknown
KERNEL32.DLL    00007FF896C255A0 Unknown      Unknown    Unknown
ntdll.dll       00007FF897DC485B Unknown      Unknown    Unknown

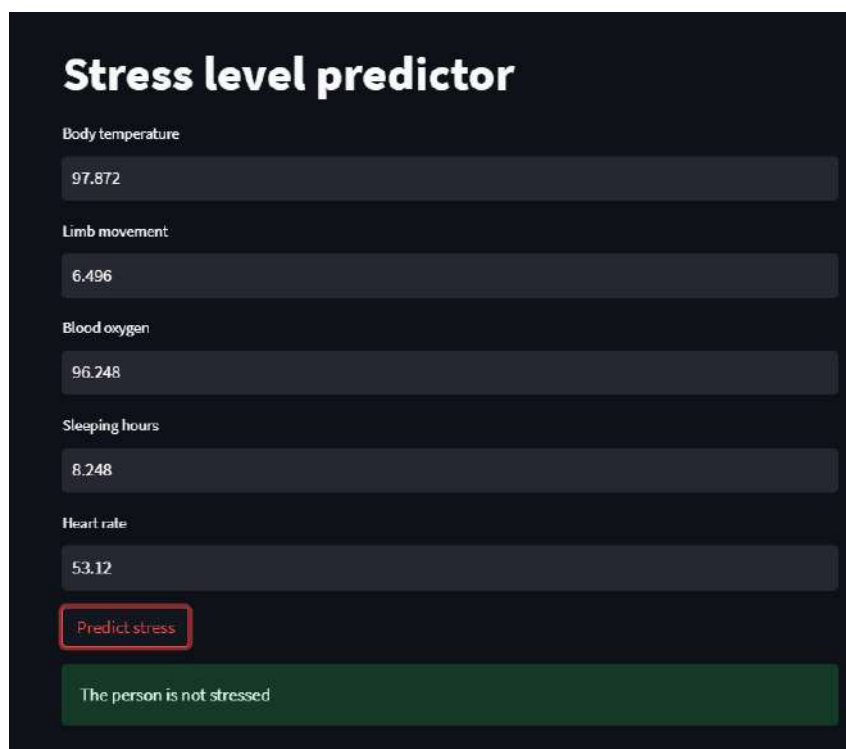
(base) C:\Users\Tanya Gupta>streamlit run "C:\Users\Tanya Gupta\Downloads\Iot_domain_project\stressapp.py"

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.16.136.131:8501

[3]
[3]
[3]
[0]
[3]
```

Figure 11: Console window for the application



### Stress level predictor

Body temperature

97.872

Limb movement

6.496

Blood oxygen

96.248

Sleeping hours

8.248

Heart rate

53.12

Predict stress

The person is not stressed

Figure 12: When person is not stressed.



**Stress level predictor**

Body temperature  
91.840

Limb movement  
16.600

Blood oxygen  
89.840

Sleeping hours  
1.840

Heart rate  
74.20

Predict stress

The person is highly stressed

Figure 13: When person is stressed.

## Result And Discussion

The proposed system may help in predicting the stressed people, thereby helpful for the society in solving the serious existing problem of stress by knowing the rate of the stressed level and taking necessary steps and preventive measures to further decrease the stressed level.

The proposed systems have taken statistical data and have taken some attributes are

Limb movement, body temperature, spO2, Sleeping hours and Heart Rate.

Based on the values have used some of the algorithms are Decision Tree, Random Forest, Logistic Regression to get accuracy because when use more algorithms can get best results will train that data and the data is also divided into training data and testing data and compare with the threshold values and that value declares whether the person is in stress or not stress.

We select Random Forest for prediction for training and test data and for imputation purposes we used Decision Tree.

Algorithm	Training data accuracy	Testing Data Accuracy
Logistic Regression	96%	64%
Random Forest	100%	92%

## Acknowledgement

We would like to express our gratitude to Professor Dr. Biswajit Dwivedy, who gave us this golden opportunity to do this wonderful project on the topic of Stress level Detection using Machine Learning and IOT, which also helped us to do a lot of research and we came to know about so many new things. We are thankful to you.

## References

- [1] Reshma Radheshamjee, and Supriya Kinariwala “Detection and Analysis of Stress using Machine Learning Techniques” International Journal of Engineering and Advanced Technology, pp. 2249-8958, 2019.
- [2] Anand Jatti, Madhvi Kannan, Alisha R. M, Vijayalakshmi P, Shrestha Sinha, “Design and Development of an IOT based Wearable device for the safety and security of women and girl children,” IEEE International conference on recent trends in electronics information communication technology, 2016.
- [3] Boon Giin Lee, Wan-Young Chung “Wearable Glove-Type Drive Stress Detection Using a Motion Sensor,” IEEE Transaction on Intelligent Transportation systems,” 2018.
- [4] Purnendu Shekhar Pandey, “Machine Learning and IoT for Prediction Detection of Stress,” 2017.
- [5] Lavanya Rachakonda, Prabha Sundaravadivel, Saraju P. Mohanty, Elias Kougianos, Madhavi Ganapathiraju, “A smart sensor in the IoMT for stress level detection.”
- [6]. L. Rachakonda, A. K. Bapatla, S. P. Mohanty, and E. Kougianos, “SaYoPillow: Blockchain-Integrated Privacy-Assured IoMT Framework for Stress Management Considering Sleeping Habits”, IEEE Transactions on Consumer Electronics (TCE), Vol. 67, No. 1, Feb 2021, pp. 20-29.
- [7]. L. Rachakonda, S. P. Mohanty, E. Kougianos, K. Karunakaran, and M. Ganapathiraju, “Smart-Pillow: An IoT based Device for Stress Detection Considering Sleeping Habits”, in Proceedings of the 4th IEEE International Symposium on Smart Electronic Systems (iSES), 2018, pp. 161--166.

```
1 import pandas as pd
2 import numpy as np
3 import datetime as dt
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import numpy as np
```

▼ Preprocessing Training Dataset

```
1 pillow = pd.read_csv("SaYoPillow.csv")
2 pillow.head()
```

	body_temperature	limb_movement	Blood_oxygen	Sleeping_hours	Heart_rate	Stress_level
0	91.840	16.600	89.840	1.840	74.20	3
1	91.552	15.880	89.552	1.552	72.76	3
2	96.000	10.000	95.000	7.000	60.00	1
3	90.768	13.920	88.768	0.768	68.84	3
4	97.872	6.496	96.248	8.248	53.12	0

```
1 pillow.shape

(630, 6)
```

```
1 pillow.isnull().values.sum()

0
```

```
1 pillow.duplicated().sum()

0
```

```
1 pillow.dtypes

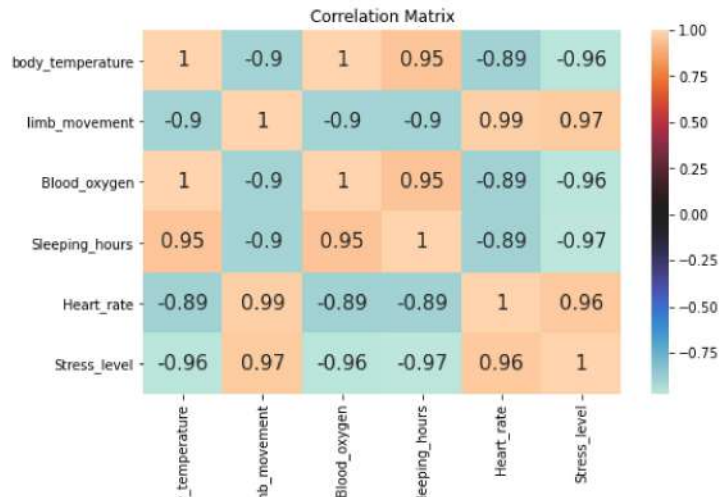
body_temperature    float64
limb_movement      float64
Blood_oxygen       float64
Sleeping_hours     float64
Heart_rate         float64
Stress_level       int64
dtype: object
```

```
1 pillow.describe()
```

	body_temperature	limb_movement	Blood_oxygen	Sleeping_hours	Heart_rate	Stress_level
count	630.00000	630.000000	630.000000	630.000000	630.000000	630.000000
mean	92.80000	11.700000	90.900000	3.700000	64.500000	2.000000
std	3.52969	4.299629	3.902483	3.054572	9.915277	1.415337
min	85.00000	4.000000	82.000000	0.000000	50.000000	0.000000
25%	90.50000	8.500000	88.500000	0.500000	56.250000	1.000000
50%	93.00000	11.000000	91.000000	3.500000	62.500000	2.000000
75%	95.50000	15.750000	94.250000	6.500000	72.500000	3.000000
max	99.00000	19.000000	97.000000	9.000000	85.000000	4.000000

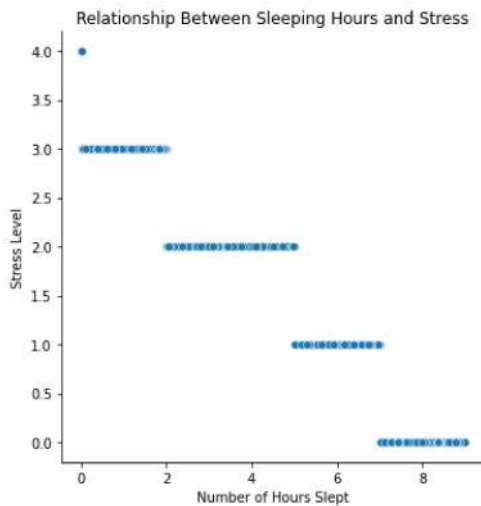
▼ Data Visualization

```
1 fig, ax = plt.subplots(figsize=(8,5))
2 cols=["body_temperature", "limb_movement","Blood_oxygen","Sleeping_hours", "Heart_rate", "Stress_level"]
3 sns.heatmap(pillow[cols].corr(), cmap="icefire", annot=True, annot_kws={'size': 15})
4 plt.title("Correlation Matrix")
5 plt.show()
```



- Stress Levels and Sleeping Hours are strongly negatively correlated. The more people sleep, the less they are likely to be stressed
- Body temperature and stress level are highly negatively correlated

```
1 sns.relplot(x="Sleeping_hours", y="Stress_level", data=pillow, kind="scatter")
2 plt.xlabel("Number of Hours Slept")
3 plt.ylabel("Stress Level")
4 plt.title("Relationship Between Sleeping Hours and Stress")
5 plt.show()
```



Here we can clearly spot the negative correlation between Stress (categorical variable) and Number of Hours Slept (continuous variable).

```
1 fig, ax = plt.subplots(figsize=(7,7))
2 pillow.groupby(pillow["Stress_level"])[["Sleeping_hours"]].mean().plot(kind='bar', rot=0, color='#9966cc')
3 plt.title("Stress Levels Measured by Sleeping Hours")
4 plt.xlabel("Stress Levels")
5 plt.ylabel("Number of Hours Slept")
6 plt.show()
```



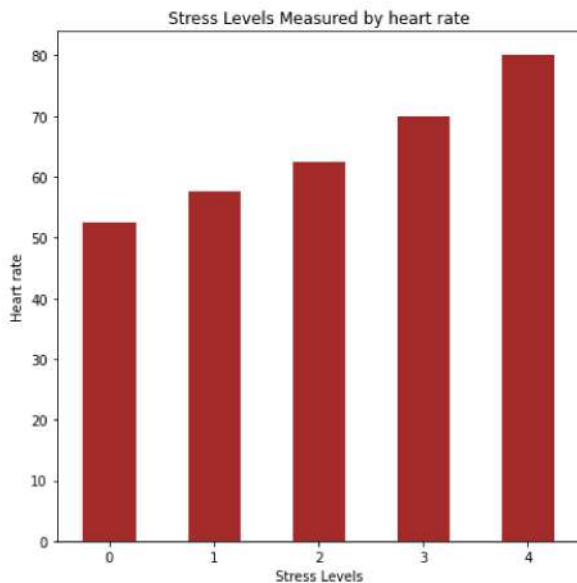
People who sleep 8 hours on average in a single day report no or very low stress levels

```
1 fig, ax = plt.subplots(figsize=(7,7))
2 pillow.groupby(pillow["Stress_level"])[["limb_movement"]].mean().plot(kind='bar', rot=0, color='#4b5320')
3 plt.title("Stress Levels Measured by Limb movement")
4 plt.xlabel("Stress Levels")
5 plt.ylabel("Limb movement")
6 plt.show()
```



It is detected that people with high limb movement have more stress

```
1 fig, ax = plt.subplots(figsize=(7,7))
2 pillow.groupby(pillow["Stress_level"])[["Heart_rate"]].mean().plot(kind='bar', rot=0, color='#a52a2a')
3 plt.title("Stress Levels Measured by heart rate")
4 plt.xlabel("Stress Levels")
5 plt.ylabel("Heart rate")
6 plt.show()
```



Stressed people have higher heart rate, people with low or no stress have heart rate between 50-60

## ▼ Outliers reduction

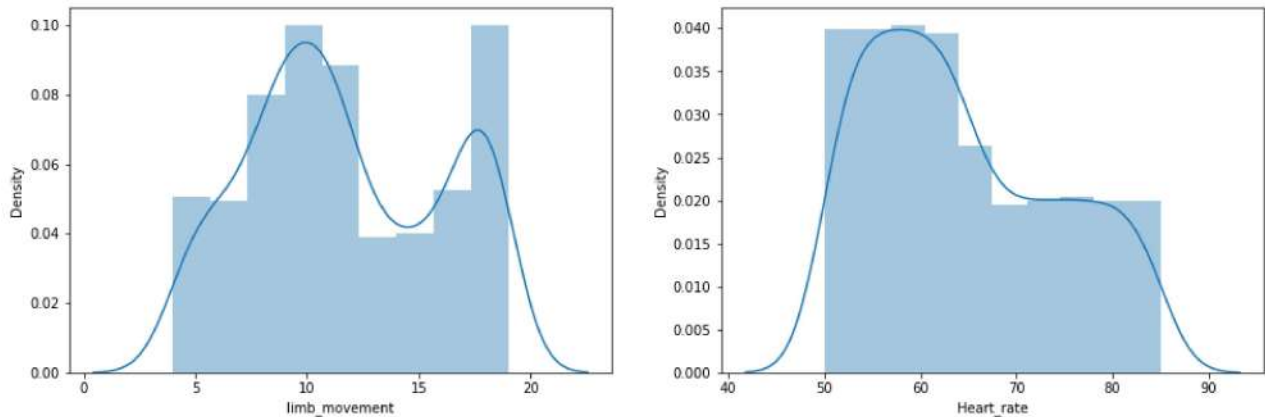
```

1 plt.figure(figsize=(16,5))
2 plt.subplot(1,2,1)
3 sns.distplot(pillow['limb_movement'])
4 plt.subplot(1,2,2)
5 sns.distplot(pillow['Heart_rate'])
6 plt.show()

```

C:\Users\ishit\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Use `histplot` instead.  
 warnings.warn(msg, FutureWarning)

C:\Users\ishit\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Use `histplot` instead.  
 warnings.warn(msg, FutureWarning)



### ▼ Applying IQR in limb movement

```

1 print("Highest allowed",pillow['limb_movement'].mean() + 3*pillow['limb_movement'].std())
2 print("Lowest allowed",pillow['limb_movement'].mean() - 3*pillow['limb_movement'].std())

```

Highest allowed 24.59888744416733  
 Lowest allowed -1.1988874441673119

```
1 pillow[(pillow['limb_movement'] > 24.59888744416733) | (pillow['limb_movement'] < -1.1988874441673119)]
```

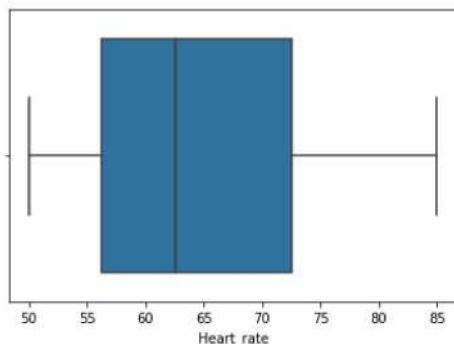
body\_temperature limb\_movement Blood\_oxygen Sleeping\_hours Heart\_rate Stress\_level

No outlier detected in limb movement

### ▼ Detecting outlier using box plot

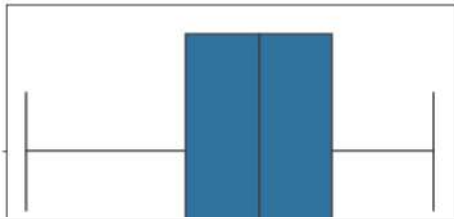
```
1 sns.boxplot(pillow['Heart_rate'])
```

C:\Users\ishit\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: warnings.warn(  
 <AxesSubplot:xlabel='Heart\_rate'>



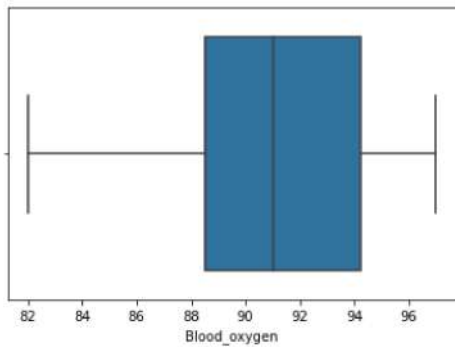
```
1 sns.boxplot(pillow['body_temperature'])
```

```
C:\Users\ishit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
warnings.warn(
<AxesSubplot:xlabel='body_temperature'>
```



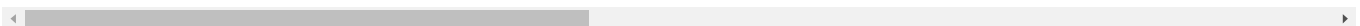
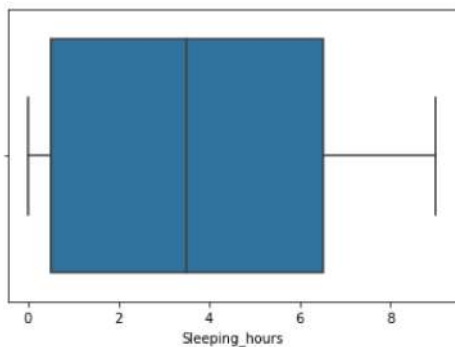
```
1 sns.boxplot(pillow['Blood_oxygen'])
```

```
C:\Users\ishit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
warnings.warn(
<AxesSubplot:xlabel='Blood_oxygen'>
```



```
1 sns.boxplot(pillow['Sleeping_hours'])
```

```
C:\Users\ishit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
warnings.warn(
<AxesSubplot:xlabel='Sleeping_hours'>
```



▼ No outlier detected in Heart rate, body temperature, Blood oxygen, Sleeping hours

```
1 df1 = pillow.copy()
```

▼ Data standardization

```
1 Y=pillow['Stress_level']
```

```
1 X=pillow.drop(['Stress_level'],axis=1)
```

```
1 X.head()
```

```
1 from sklearn.preprocessing import StandardScaler
2 scaler = StandardScaler()
3 scaler.fit(X)
```

```
StandardScaler()
```

```
1 X= scaler.transform(X)
```

```
7          0.7312          0.400          0.240          0.270          0.12
```

## ▼ Training the ML Model

We use decision tree for imputation, LR and Random forest for prediction

## ▼ Logistic Regression

```
1 from sklearn.linear_model import LogisticRegression
2 lr=LogisticRegression(solver='liblinear',multi_class='ovr')
3 lr.fit(X,Y)
4 from sklearn.metrics import accuracy_score
5 X_train_prediction = lr.predict(X)
6 training_data_accuracy_LR= accuracy_score(Y, X_train_prediction)
7 print('Accuracy score of training data : ', training_data_accuracy_LR)
```

```
Accuracy score of training data : 0.9698412698412698
```

## ▼ Random Forest

```
1 from sklearn.ensemble import RandomForestClassifier
2 classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
3 classifier.fit(X, Y)
4 X_train_prediction = classifier.predict(X)
5 training_data_accuracy_RF= accuracy_score(Y, X_train_prediction)
6 print('Accuracy score of training data : ', training_data_accuracy_RF)
```

```
Accuracy score of training data : 1.0
```

## ▼ Testing Data

```
1 p_test=pd.read_csv('test.csv')
```

## ▼ Dropping unnecessary columns

```
1 p_test.rename(columns = {'Sleeping_hours\t':'Sleeping_hours'}, inplace = True)
```

```
1 p_test = p_test.drop(['Snoring_rate','Respiration_rate','Eye_movement'],axis=1)
2 p_test.head()
```

	limb_movement	Sleeping_hours	Heart_rate	Stress_level
0	16	1.840	70	3
1	20	1.552	95	3
2	20	7.000	65	1
3	20	0.768	89	3
4	7	8.248	72	0

## ▼ Creating a copy of the testing dataset for imputation purposes

```
1 p1_test = p_test.copy()
2 p1_test.head()
```



	limb_movement	Sleeping_hours	Heart_rate	Stress_level
0	16	1.840	70	3
1	20	1.552	95	3
2	20	7.000	65	1
3	20	0.768	89	3
4	7	8.248	72	0

```
1 df1.head() #reqd
```

	body_temperature	limb_movement	Blood_oxygen	Sleeping_hours	Heart_rate	Stress_level
0	91.840	16.600	89.840	1.840	74.20	3
1	91.552	15.880	89.552	1.552	72.76	3
2	96.000	10.000	95.000	7.000	60.00	1
3	90.768	13.920	88.768	0.768	68.84	3
4	97.872	6.496	96.248	8.248	53.12	0

The Features to be Imputed are shifted to the end as they will be the dependent variables which are imputed using the other features.:

```
1 df2 = df1.copy()
```

```
1 df1 = df1[['limb_movement', 'Sleeping_hours', 'Heart_rate', 'Stress_level', 'body_temperature', 'Blood_oxygen']]
2 df1.head()
```

	limb_movement	Sleeping_hours	Heart_rate	Stress_level	body_temperature	Blood_oxygen
0	16.600	1.840	74.20	3	91.840	89.840
1	15.880	1.552	72.76	3	91.552	89.552
2	10.000	7.000	60.00	1	96.000	95.000
3	13.920	0.768	68.84	3	90.768	88.768
4	6.496	8.248	53.12	0	97.872	96.248

## ▼ body\_temperature imputation

Creating 2 variables of independent features and dependent feature (which is about to be used for imputation)

```
1 df1_x = df1.iloc[:, :-2].values
2 df1_y = df1.iloc[:, -2].values
3 df1_x
```

```
array([[16.6 , 1.84 , 74.2 , 3. ],
       [15.88 , 1.552, 72.76 , 3. ],
       [10.   , 7.   , 60.   , 1. ],
       ...,
       [17.752, 0.   , 78.76 , 4. ],
       [ 9.728, 6.728, 59.32 , 1. ],
       [11.392, 4.088, 63.48 , 2. ]])
```

```
1 p1_test1 = np.array(p1_test)
2 #p1_test1
```

Algorithms tried for Imputation:

- Mice Algorithm
- Mean/Median
- Random Forest Regression
- Decision Tree Regression
- Polynomial Regression
- Support Vector Regression

The best final accuracy of Stress levels was provided by Decision Tree Regression Algorithm.

Other methods gave accuracy of only about 24%.

Mean/Median methods are only suitable when there are only a few missing values in a particular feature and not when all the values of the feature are supposed to be imputed.

## ▼ SVR

```
1 # from sklearn.preprocessing import StandardScaler
2 # sc_x = StandardScaler()
3 # X_body_temp = sc_x.fit_transform(df1_x)
4 # sc_y = StandardScaler()
5 # y_body_temp = sc_y.fit_transform(df1_y.reshape(len(df1_y),1))
6
7 # from sklearn.svm import SVR
8 # regressor = SVR(kernel = 'rbf') # or leave blank
9 # regressor.fit(X_body_temp,y_body_temp)
10
11 # sc_y.inverse_transform(regressor.predict(sc_x.transform(np.array(p1_test1[0]).reshape(1, -1))).reshape(1,-1))[0][0]
```

```
1 # # Prediction (SVR)
2 # body_temp_pred = []
3 # body_temp_pred_df = pd.DataFrame(columns = ['body_temperature'])
4 # j = 0
5 # for i in range(len(p1_test1)):
6 #     body_temp_pred_df = pd.DataFrame(sc_y.inverse_transform(regressor.predict(sc_x.transform(np.array(p1_test1[i]).reshape(1, -1))).reshape(1,-1))).re
7 #     #body_temp_pred_df.insert(j,'body_temperature',sc_y.inverse_transform(np.array(regressor.predict(sc_x.transform(np.array(i).reshap
```

## ▼ Decision Tree and Random Forest Regression

```
1 # from sklearn.ensemble import RandomForestRegressor
2 # regressor = RandomForestRegressor(n_estimators=10,random_state=0)
3 from sklearn.tree import DecisionTreeRegressor
4 regressor = DecisionTreeRegressor(random_state = 0)
5 regressor.fit(df1_x,df1_y)
6 p1_body_temp = regressor.predict(p1_test1)
7 p1_body_temp
```

```
array([[91.392, 91.664, 96.    , 91.2  , 98.376, 95.664, 97.272, 98.88 ,
        96.408, 94.88 , 93.376, 95.856, 98.376, 89.72 , 91.2  , 94.784,
        95.28 , 91.2  , 91.184, 92.384, 93.728, 93.728, 92.784, 90.368,
        91.648, 94.784, 93.728, 94.784, 91.984, 92.208, 98.4  , 90.    ,
        98.376, 85.08 , 92.    , 90.608, 98.376, 86.48 , 96.6  , 86.48 ,
        92.144, 92.592, 93.408, 98.4  , 93.376, 89.72 , 92.272, 89.72 ,
        94.88 , 86.48 ])
```

Creating a dataframe of the Imputed data which is to be later concatenated with the testing dataset

```
1 p1_body_temp = pd.DataFrame(p1_body_temp, columns = ['body_temperature'])
2 p1_body_temp.head()
```

	body_temperature
0	91.392
1	91.664
2	96.000
3	91.200
4	98.376

## ▼ Blood\_oxygen Imputation

Creating 2 variables of independent features and dependent feature (which is about to be used for imputation)

```
1 df1_x2 = df1.iloc[:, :-2].values
2 df1_y2 = df1.iloc[:, -1].values
3 df1_x2
```

```
array([[16.6 ,  1.84 , 74.2  ,  3.   ],
       [15.88 ,  1.552, 72.76 ,  3.   ],
       [10.   ,  7.    , 60.   ,  1.   ],
       ...,
       [17.752,  0.    , 78.76 ,  4.   ],
```

```
[ 9.728,  6.728, 59.32 ,  1.  ],
[11.392,  4.088, 63.48 ,  2.  ]])
```

```
1 #df1_y2
```

```
1 # p1_test1
```

### ▼ Training the Decision Tree Regression model with the training dataset to impute the feature - blood oxygen

```
1 # from sklearn.ensemble import RandomForestRegressor
2 # regressor1 = RandomForestRegressor(n_estimators=10,random_state=0)
3 from sklearn.tree import DecisionTreeRegressor
4 regressor1 = DecisionTreeRegressor(random_state = 0)
5 regressor1.fit(df1_x2,df1_y2)
6 p1_blood_ox = regressor1.predict(p1_test1)
7 p1_blood_ox

array([ 89.84 ,  89.68 ,  95.   ,  88.976,  96.552,  94.568,  96.568,  97.   ,
        95.096,  93.32 ,  90.976,  94.784,  96.552,  85.168,  89.2   ,  92.936,
        92.936,  88.976,  88.736,  90.656,  91.616,  91.6   ,  90.752,  88.464,
        89.328,  93.248,  91.84 ,  92.936,  89.984,  90.112,  96.92 ,  88.   ,
        96.552,  83.056,  90.   ,  88.656,  96.552,  83.68 ,  95.384,  83.68 ,
        90.112,  90.112,  91.536,  96.92 ,  90.976,  85.168,  90.112,  85.168,
        94.328,  83.824])
```

```
1 p1_blood_ox = pd.DataFrame(p1_blood_ox, columns = ['Blood_oxygen'])
2 p1_blood_ox.head()
```

	Blood_oxygen
0	89.840
1	89.680
2	95.000
3	88.976
4	96.552

### ▼ The final testing dataset

```
1 test = pd.concat([p1_test,p1_body_temp,p1_blood_ox], axis=1, join='inner')
2 test.head()
```

	limb_movement	Sleeping_hours	Heart_rate	Stress_level	body_temperature	Blood_oxygen
0	16	1.840	70	3	91.392	89.840
1	20	1.552	95	3	91.664	89.680
2	20	7.000	65	1	96.000	95.000
3	20	0.768	89	3	91.200	88.976
4	7	8.248	72	0	98.376	96.552

```
1 test.tail()
```

	limb_movement	Sleeping_hours	Heart_rate	Stress_level	body_temperature	Blood_oxygen
45	15	0.000	85	4	89.720	85.168
46	8	2.432	67	2	92.272	90.112
47	15	0.000	89	4	89.720	85.168
48	4	6.344	70	1	94.880	94.328
49	19	0.000	78	4	86.480	83.824

### ▼ Rearranging the features according to the dataset with which the Classification model has been trained with

```
1 test = test[['body_temperature','limb_movement','Blood_oxygen','Sleeping_hours','Heart_rate','Stress_level']]
2 test.head()
```

	body_temperature	limb_movement	Blood_oxygen	Sleeping_hours	Heart_rate	Stress_level
0	91.392	16	89.840	1.840	70	3
1	91.664	20	89.680	1.552	95	3
2	96.000	20	95.000	7.000	65	1
3	91.200	20	88.976	0.768	89	3
4	98.376	7	96.552	8.248	72	0

▼ Prediction

```
1 Y_test = test['Stress_level']
```

```
1 X_test = test.drop(['Stress_level'],axis=1)
```

```
1 X
array([[ -0.2721947 ,  1.14053872, -0.27183777, -0.60940713,  0.97906569],
       [ -0.35385311,  0.97294936, -0.34569558, -0.70376694,  0.83371986],
       [  0.90731566, -0.39569711,  1.05144797,  1.08120619, -0.45420573],
       ...,
       [ -1.67853398,  1.4086817 , -1.70385862, -1.21226149,  1.43932751],
       [  0.83019383, -0.45900864,  0.94681608,  0.99208859, -0.52284127],
       [  0.1678534 , -0.07169101,  0.12617376,  0.12712364, -0.1029533 ]])
```

```
1 X_test.head()
```

	body_temperature	limb_movement	Blood_oxygen	Sleeping_hours	Heart_rate
0	91.392	16	89.840	1.840	70
1	91.664	20	89.680	1.552	95
2	96.000	20	95.000	7.000	65
3	91.200	20	88.976	0.768	89
4	98.376	7	96.552	8.248	72

▼ Normalizing the testing dataset

```
1 X_test = scaler.transform(X_test)
```

▼ Predicting Test results

```
1 X_test_prediction = lr.predict(X_test)
2 test_data_accuracy_LR= accuracy_score(Y_test, X_test_prediction)
3 print('Accuracy score of training data : ', test_data_accuracy_LR)
```

Accuracy score of training data : 0.64

▼ Predicting test data using random Forest

```
1 from sklearn.ensemble import RandomForestClassifier
2 classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
3 classifier.fit(X, Y)
```

RandomForestClassifier(criterion='entropy', n\_estimators=10, random\_state=0)

```
1 X_test_prediction = classifier.predict(X_test)
2 test_data_accuracy_RF= accuracy_score(Y_test, X_test_prediction)
3 print('Accuracy score of training data : ', test_data_accuracy_RF)
```

Accuracy score of training data : 0.92

▼ Creating the GUI Interface

```
1 from tkinter import *
```

```
1 import pickle
2 filename = 'trained_model.sav'
3 pickle.dump(lr,open(filename,'wb'))
```

[+ Code](#)[+ Text](#)