

ISHWARYA R [2127200501059]

JESSICA JANET P A [2127200501063]

PREETI PURNIMAA K [2127200501111]

CS18811 – PROJECT WORK

Batch No : 20
Name of the Guide : Dr. P. JANARTHANAN
Date of Review : 27.05.2024
Domain : DATA ANALYTICS / MACHINE LEARNING

ABSTRACT

- Stress level analysis using supervised learning is a proposed system that aims to predict and analyse an individual's stress levels based on various input features.
- Supervised learning refers to a machine learning approach where a model is trained on labeled data to make predictions or classifications.
- Stress Analysis and Insights: The predicted stress levels can be further analyzed to gain insights into patterns, triggers, or factors contributing to stress.
- Data visualization techniques and statistical analyses can help identify trends or correlations between features and stress levels.
- Data is collected and analyzed, and the system can provide personalized feedback and recommendations to individuals based on their stress levels. This feedback can include stress management strategies, relaxation techniques, or lifestyle adjustments to help reduce stress and improve well-being.



PROBLEM STATEMENT

Stress is any situation that evokes negative thoughts and feelings in a person. In today's modern environment, individuals are increasingly concerned about their mental health and lifestyle. If the stress level is tracked, it can help to prevent depression and other serious mental health issues to maintain a healthy lifestyle, and improve general health.



INTRODUCTION TO BASIC CONCEPTS

- Stress is a prevalent issue affecting individuals across various domains of life, including work, education, and personal relationships. Identifying and understanding stress levels is crucial for effective intervention and support.
- This abstract presents an overview of a study that focuses on stress level analysis using supervised learning approaches.
- The objective of this research is to develop a reliable and accurate system that can predict stress levels based on various input features.
- The study employs supervised learning techniques, which utilize labeled training data to
- build a predictive model.
- The model is trained to classify stress levels based on the input features and corresponding stress labels.



INTRODUCTION TO BASIC CONCEPTS

The scope of human stress analysis encompasses a wide range of factors related to the physiological, psychological, and environmental aspects of stress experienced by individuals.

- **Physiological Responses:** This includes studying changes in heart rate, blood pressure, and other physiological markers.
- **Psychological Responses:** Stress can have significant effects on mental health and well-being. Research in this area explores cognitive responses to stress, such as the impact of stress on mood, emotions, and mental disorders like anxiety and depression.
- **Risk Factors and Protective Factors:** Research in this area aims to identify factors that increase vulnerability to stress.
- **Health Consequences:** Chronic stress is associated with a range of adverse health outcomes, including cardiovascular diseases, immune dysfunction, gastrointestinal issues, and mental health disorders.
- **Assessment Tools and Measurement:** Developing reliable and valid measures for assessing stress is essential for research and clinical practice.



LITERATURE REVIEW

S.No	Authors	Research Article Title	Review of the Paper
1.	Park, J., Ahn, H., Youn, K., Lee, M., & Hong, S. (2023).	Ensemble Learning to Identify Depression Indicators for Korean Farmers.	The study conducted on understanding the factors contributing to depression among farmers is of paramount importance for ensuring their well-being and productivity. By employing advanced tree-based machine learning algorithms, specifically focusing on the Category Boosting (CatB) algorithm, the research has provided valuable insights into this critical issue. This research serves as a valuable resource for addressing mental health challenges in the farming community and promoting a healthier and more resilient agricultural workforce.
2.	Dennis R. da C. Silva , Zelun Wang , and Ricardo Gutierrez-Osuna (2021).	Towards Participant-Independent Stress Detection Using Instrumented Peripherals.	This paper proposes a minimalist design that can be easily replicated by other researchers using off-the-shelf and low-cost hardware. It validates the design in a laboratory experiment that simulates office tasks and mild stressors while avoiding methodological limitations of previous studies. They compare stress-detection performance when using conventional features reported in the literature.



LITERATURE REVIEW

S.No	Authors	Research Article Title	Review of the Paper
3.	Karl Magtibay, and Karthikeyan Umapathy, Senior Member, IEEE, (2023).	A Review of Tools and Methods for Detection, Analysis, and Prediction of Allostatic Load due to Workplace Stress.	This study emphasizes the need for the allostatic load model in affective computing to create disease and illness prediction models. First, it briefly introduces the physiological and psychological basis of allostatic load. Next, it reviews stress studies within affective computing that may benefit from an allostatic load model of stress.
4.	Hena Yasmin, Swaziland, Salman Khalil, Amity University, India, Ramsha Mazhar, Allan Grey, South Africa (2020).	Covid 19: Stress Management among Students and its Impact on Their Effective Learning.	Covid-19 has caused significant distress around the globe. It is not limited to adults only, but stress is increasingly affecting children of all age groups. An attempt is made through this paper to know the impact of stress among students and the necessity of managing it in order to make the learning effective.



LITERATURE REVIEW

S.No	Authors	Research Article Title	Review of the Paper
5.	Mr. Devraj Singh Chouhan(2016).	Stress and Its Major Effects on Human Health.	In the recent time medical sector have better advancement, but few people are having an effort toward Mental health, according to survey in recent year psychological problems are very common that's lead affecting the quality of life and physical health. Stress is a change in body due to the reason of stimuli this change is having a harmful effect on body mechanism & disturbs the nutritional pattern of the body. This paper is helpful to improve the knowledge regarding mental health & how to cope with stress.
6.	Amir Mohammad Shamsavarani, Esfandiar Azad Marz Abadi , Maryam Hakimi Kalkhoran (2015)	Facts and Theories through Literature Review	The design of the present study was systematic review. Inclusion criteria were subjective relevance to study keywords (include stress, stress control, stress reduction, social stress, community stress, group stress).



ISSUES AND CHALLENGES

1. **Data quality and quantity:** Obtaining high-quality and sufficient data for training ML models can be a challenge, as stress levels are subjective and can vary greatly among individuals.
2. **Feature selection:** Identifying relevant features that accurately represent stress levels can be complex, as stress is influenced by various factors such as physical, emotional, and environmental conditions.
3. **Overfitting and generalization:** Ensuring that ML models generalize well to unseen data and do not overfit to the training data is crucial for accurate stress level analysis.
4. **Class imbalance:** Imbalance in the distribution of stress levels in the dataset can lead to biased models that perform poorly on underrepresented classes..

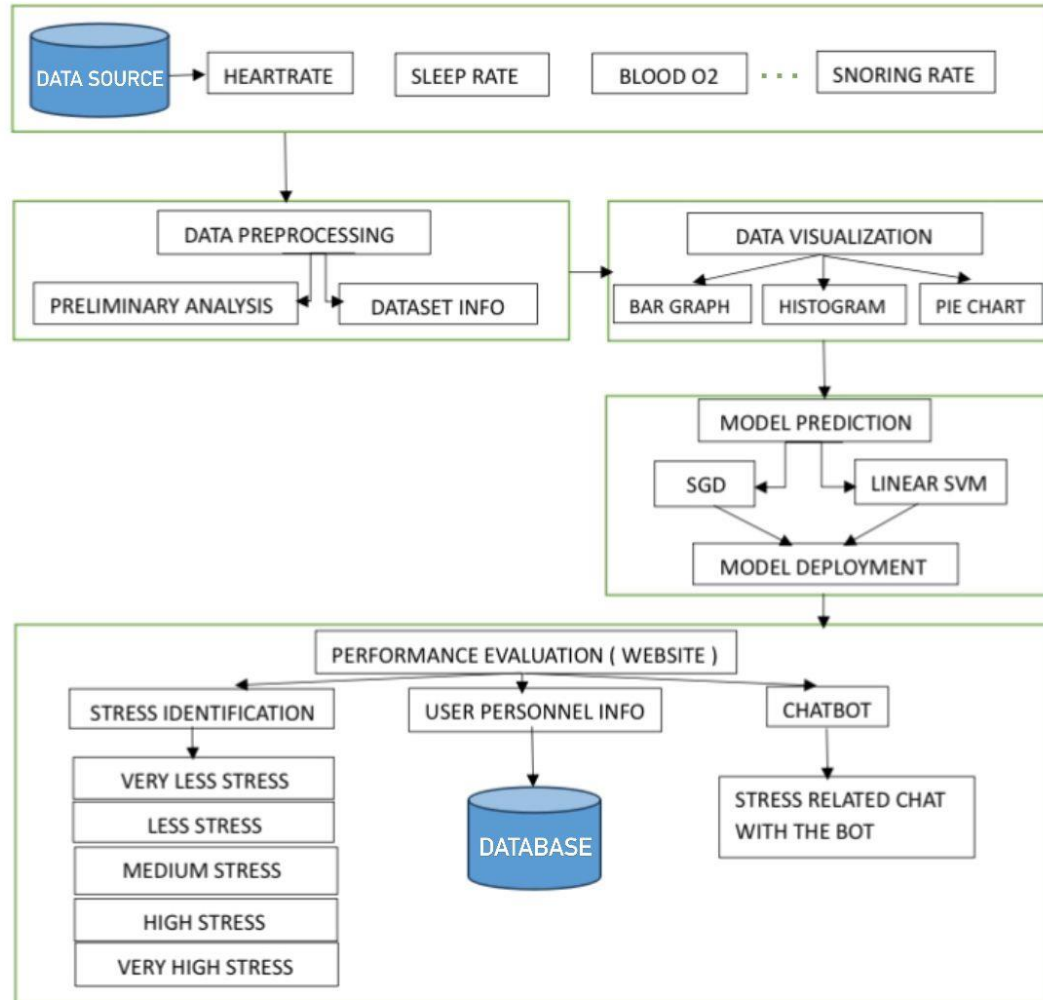


PROPOSED WORK

1. **Data Collection:** Gather a diverse dataset containing information on various factors that may influence stress levels, such as demographic data, lifestyle habits, work-related stressors, and mental health indicators.
2. **Data Preprocessing:** Clean and preprocess the collected data to handle missing values, normalize features, and ensure data quality for training machine learning models.
3. **Algorithm Selection:** Explore and evaluate different machine learning algorithms, such as regression models, decision trees, and neural networks, to determine the most suitable approach for predicting stress levels accurately.
4. **Algorithm Training:** Train the selected machine learning model on the preprocessed data to learn patterns and relationships between input features and stress levels.
5. **Evaluation:** Assess the performance of the trained model using metrics such as accuracy, precision, recall, and F1 score to measure its effectiveness in predicting stress levels.
6. **Model Deployment:** Implement the trained model into a user-friendly interface or application that can provide real-time predictions of stress levels based on user input and build a interactive web application for the users.



PROPOSED WORK-DETAILED ARCHITECTURE DIAGRAM



SYSTEM REQUIREMENTS AND TOOLS

- **SOFTWARE REQUIREMENTS :**

- **Operating System :**

- Windows 10 or later
- Programming language- Python 3.1

- **IDE :**

- Jupyter Notebook

- **HARDWARE REQUIREMENTS :**

- **Processor :**

- Intel i3 or better

- **Memory :**

- 8GB RAM and 32bit processor



SYSTEM REQUIREMENTS AND TOOLS

- **Tools and Libraries:**
- Jupyter Notebook
- Visualization Library- Seaborn and Matplotlib
- Data storage and manipulation- csv and pandas libraries
- Deployment- Django



MODULES AND ITS DESCRIPTION

MODULE 1: DATA PREPROCESSING

Input: Raw data from Kaggle

Output: Processed and cleaned data

Description:

- This module focuses on cleaning the raw data ie., To find the missing values, duplicate values, and description of the data type whether it is a float variable or integer. This process is essential because the quality of the data used in machine learning significantly impacts the performance of the models.
- Once the consistent data is obtained after the cleaning process, preprocessing of the data is done. Preprocessing involves transforming and preparing the data for analysis.



MODULES AND ITS DESCRIPTION

MODULE 2 : DATA VISUALIZATION AND ANALYSIS

Input: Preprocessed data

Output: Data visualization

Description:

- This module focuses on visualizing the processed data for deeper analysis and easy interpretation of the values. Data visualization is the representation of data through the use of common graphics, such as charts, plots, infographics, etc.
- We use various visualization techniques such as count plot, histogram, box plot, violin plot, density graph, heat maps, pie chart, etc. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.



MODULES AND ITS DESCRIPTION

MODULE 3 : ALGORITHM IMPLEMENTATION

Algorithm 1: Stochastic Gradient Descent (SGD)

Input: Train and test data

Output: Accuracy, precision, f1-score, support

Description:

- Stochastic Gradient Descent (SGD) works by iteratively updating the model parameters in the direction that minimizes the loss function. Instead of computing the gradient of the entire dataset (as in batch gradient descent), SGD computes the gradient of the loss function with respect to a single data point or a small subset of data points (mini-batch) at each iteration.
- During each iteration, the model parameters are updated using the gradient of the loss function with respect to the current data point or mini-batch, scaled by a learning rate.

MODULES AND ITS DESCRIPTION

Algorithm 2: Linear SVM

Input: Train and test data

Output: Predictive model with high accuracy, precision, f1 score

Description:

- Linear Support Vector Machine (Linear SVM) is a supervised machine learning algorithm used for classification and regression tasks.
- It works by finding the optimal hyperplane that best separates the data points into different classes in a high-dimensional space.
- The hyperplane is chosen in such a way that it maximizes the margin, which is the distance between the hyperplane and the nearest data points from each class, known as support vectors.

MODULES AND ITS DESCRIPTION

MODULE 4 : DEPLOYMENT USING DJANGO

Input: Medical values

Output: Stress level result and Chat Bot

Description:

- This module focuses on the Application part of the proposed work. This process is carried out by the user entering the stress prediction values such as Snoring rate, Respiration rate, Body Temperature, Limb Movement, Blood oxygen levels etc.
- Once the values are entered the Stress Level (Very Low, Low, Medium, High and Very High) is predicted and is shown as the Result. Additionally the project has a chatbot which allows the users to ask questions to assess their stress levels. The chatbot offers coping strategies according to the users stress level.

IMPLEMENTATION SCREENSHOTS

DATA PREPROCESSING AND DATA CLEANING

```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('STRESS.csv')
df.head()
```

	snoring rate	respiration rate	body temperature	limb movement
0	93.80	25.680	91.840	16.600
1	91.64	25.104	91.552	15.880
2	60.00	20.000	96.000	10.000
3	85.76	23.536	90.768	13.920
4	48.12	17.248	97.872	6.496

	blood oxygen	retina eye movement	sleeping rate	heart rate
0	89.840	99.60	1.840	74.20
1	89.552	98.88	1.552	72.76
2	95.000	85.00	7.000	60.00
3	88.768	96.92	0.768	68.84
4	96.248	72.48	8.248	53.12

```
# sr-sleeping rate
# rr-heart rate
# t-temperature
# lm-lib movement
# bo-body mass
# rem-retina eye movement
# sr1-snoring rate
# hr- sleeping hours
# sl-stress level
```

```
df.tail()
```

	snoring rate	respiration rate	body temperature	limb movement
12592	98.624	28.624	88.280	18.312
12593	98.400	28.400	88.000	18.200
12594	91.880	25.168	91.584	15.960

12595	64.800	20.480	92.480	10.480
12596	48.440	17.376	98.064	6.752

```
df.describe()
```

	snoring rate	respiration rate	body temperature	limb movement
count	11783.000000	11783.000000	11783.000000	11783.000000
mean	71.432587	21.773457	92.833860	11.656748
std	19.530463	4.010241	3.577956	4.354447
min	45.000000	16.000000	85.000000	4.000000
25%	51.840000	18.368000	90.464000	8.368000
50%	69.600000	20.960000	93.040000	10.960000
75%	91.520000	25.072000	95.632000	15.840000
max	100.000000	30.000000	99.000000	19.000000

	blood oxygen	retina eye movement	sleeping rate	heart rate
count	11783.000000	11783.000000	11783.000000	11783.000000
mean	90.934125	88.297194	3.740022	64.433642
std	3.952450	12.097247	3.086975	10.025601
min	82.000000	60.000000	0.000000	50.000000

25%	88.464000	80.920000	0.464000	55.920000
50%	91.040000	89.800000	3.560000	62.400000
75%	94.448000	98.840000	6.632000	72.680000
max	97.000000	105.000000	9.000000	85.000000



IMPLEMENTATION SCREENSHOTS

```
df.shape
(12597, 9)

df.size
113373

df.columns
Index(['snoring rate', 'respiration rate', 'body temperature', 'limb movement',
      'blood oxygen', 'retina eye movement', 'sleeping rate', 'heart rate',
      'stress level'],
      dtype='object')

df.isnull().sum()
snoring rate      814
respiration rate  814
body temperature  814
limb movement     814
blood oxygen      814
retina eye movement 814
sleeping rate     814
heart rate        814
stress level      814
dtype: int64

df = df.dropna()
```

```
df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11783 entries, 0 to 12596
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   snoring rate          11783 non-null  float64
1   respiration rate      11783 non-null  float64
2   body temperature      11783 non-null  float64
3   limb movement         11783 non-null  float64
4   blood oxygen          11783 non-null  float64
5   retina eye movement   11783 non-null  float64
6   sleeping rate         11783 non-null  float64
7   heart rate            11783 non-null  float64
8   stress level          11783 non-null  float64
dtypes: float64(9)
memory usage: 920.5 KB
```

IMPLEMENTATION SCREENSHOTS

DATA VISUALIZATION AND DATA ANALYSIS

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('STRESS.csv')
df.head()
```

	snoring rate	respiration rate	body temperature	limb movement \
0	93.80	25.680	91.840	16.600
1	91.64	25.104	91.552	15.880
2	60.00	20.000	96.000	10.000
3	85.76	23.536	90.768	13.920
4	48.12	17.248	97.872	6.496

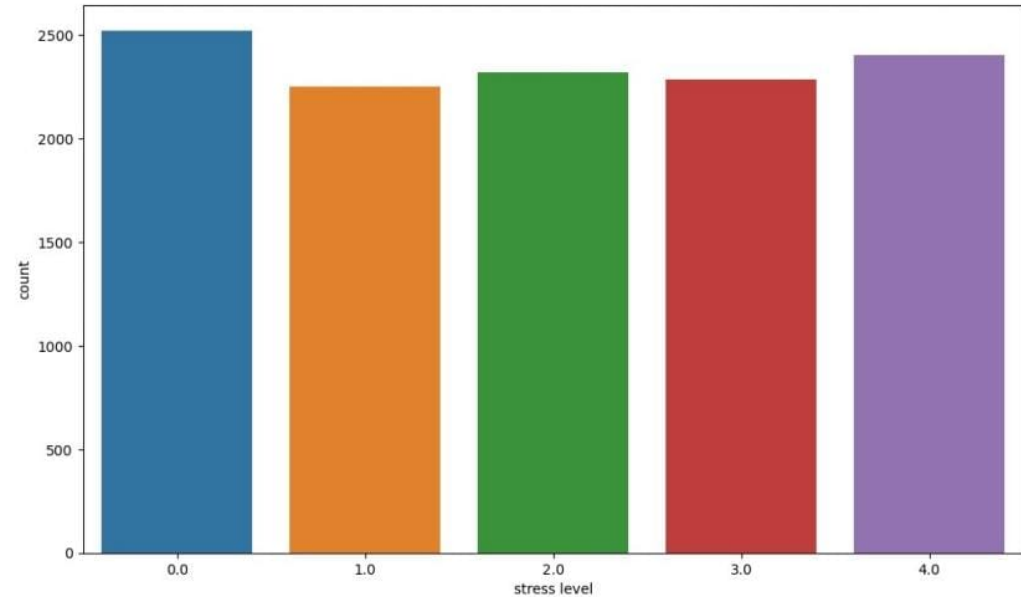
	blood oxygen	retina eye movement	sleeping rate	heart rate
stress level				
0	89.840	99.60	1.840	74.20
3.0	89.552	98.88	1.552	72.76
1				
3.0	95.000	85.00	7.000	60.00
2				
1.0				
3	88.768	96.92	0.768	68.84
3.0				
4	96.248	72.48	8.248	53.12
0.0				

```
df.columns

Index(['snoring rate', 'respiration rate', 'body temperature', 'limb
movement',
      'blood oxygen', 'retina eye movement', 'sleeping rate', 'heart
rate',
      'stress level'],
      dtype='object')

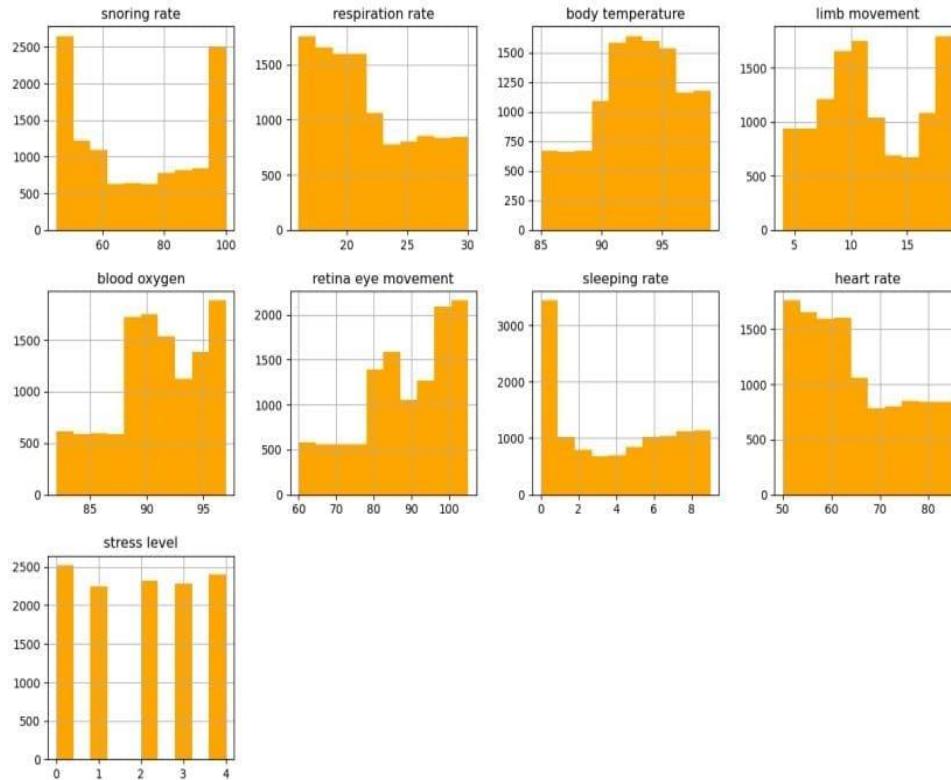
plt.figure(figsize=(12,7))
sns.countplot(x='stress level',data=df)

<Axes: xlabel='stress level', ylabel='count'>
```



IMPLEMENTATION SCREENSHOTS

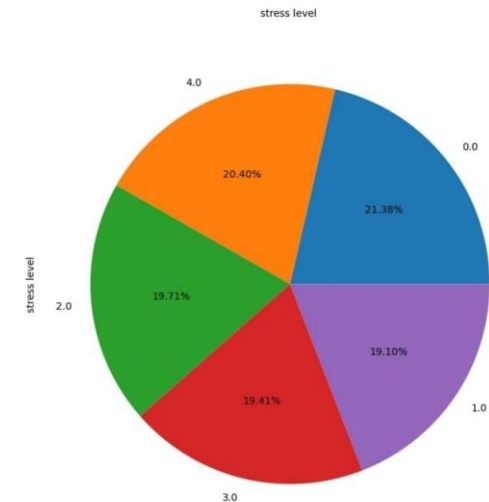
```
df.hist(figsize=(15,55),layout=(15,4), color='orange')  
plt.show()
```



```
def plot(df, variable):  
    dataframe_pie = df[variable].value_counts()  
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%',  
    fontsize = 10)  
    ax.set_title(variable + '\n', fontsize = 10)  
    return np.round(dataframe_pie/df.shape[0]*100,2)
```

```
plot(df, 'stress level')
```

```
0.0    20.00  
4.0    19.08  
2.0    18.43  
3.0    18.16  
1.0    17.87  
Name: stress level, dtype: float64
```



IMPLEMENTATION SCREENSHOTS

SGD CLASSIFIER ALGORITHM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('STRESS.csv')
df.head()
```

	snoring rate	respiration rate	body temperature	limb movement \
0	93.80	25.680	91.840	16.600
1	91.64	25.104	91.552	15.880
2	60.00	20.000	96.000	10.000
3	85.76	23.536	90.768	13.920
4	48.12	17.248	97.872	6.496

	blood oxygen	retina eye movement	sleeping rate	heart rate
stress level				
0	89.840	99.60	1.840	74.20
3.0				
1	89.552	98.88	1.552	72.76
3.0				
2	95.000	85.00	7.000	60.00
1.0				
3	88.768	96.92	0.768	68.84
3.0				
4	96.248	72.48	8.248	53.12
0.0				

```
df=df.dropna()
df.columns
Index(['snoring rate', 'respiration rate', 'body temperature', 'limb
movement',
      'blood oxygen', 'retina eye movement', 'sleeping rate', 'heart
rate',
      'stress level'],
      dtype='object')
df.head()
```

	snoring rate	respiration rate	body temperature	limb movement \
0	93.80	25.680	91.840	16.600
1	91.64	25.104	91.552	15.880

IMPLEMENTATION SCREENSHOTS

```

2      60.00      20.000      96.000      10.000
3      85.76      23.536      90.768      13.920
4      48.12      17.248      97.872      6.496

```

	blood oxygen stress level	retina eye movement	sleeping rate	heart rate
0	89.840	99.60	1.840	74.20
3.0	89.552	98.88	1.552	72.76
1	95.000	85.00	7.000	60.00
3.0	88.768	96.92	0.768	68.84
2	96.248	72.48	8.248	53.12
0.0				

```

x1 = df.drop(labels='stress level', axis=1)
y1 = df.loc[:, 'stress level']

```

```

import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

```

```

ros = RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      : ", Counter(y1))
print("OVER SAMPLING DATA COUNT : ", Counter(y))

OUR DATASET COUNT      : Counter({0.0: 2521, 4.0: 2403, 2.0: 2322,
3.0: 2287, 1.0: 2251})
OVER SAMPLING DATA COUNT : Counter({3.0: 2521, 1.0: 2521, 0.0: 2521,
2.0: 2521, 4.0: 2521})

```

```

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.20, random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET      : ", len(x_train))
print("NUMBER OF TEST DATASET       : ", len(x_test))
print("TOTAL NUMBER OF DATASET      : ", len(x_train)+len(x_test))

```

```

NUMBER OF TRAIN DATASET      : 10084
NUMBER OF TEST DATASET       : 2521
TOTAL NUMBER OF DATASET      : 12605

```

```

print("NUMBER OF TRAIN DATASET      : ", len(y_train))
print("NUMBER OF TEST DATASET       : ", len(y_test))
print("TOTAL NUMBER OF DATASET      : ", len(y_train)+len(y_test))

```

```

NUMBER OF TRAIN DATASET      : 10084
NUMBER OF TEST DATASET       : 2521
TOTAL NUMBER OF DATASET      : 12605

```

```

from sklearn.linear_model import SGDClassifier

```

```

SGT = SGDClassifier()
SGT.fit(x_train,y_train)

```

```

SGDClassifier()

```

```

predicted = SGT.predict(x_test)

```

```

from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF SGD CLASSIFIER:\n\n',cr)

```

THE CLASSIFICATION REPORT OF SGD CLASSIFIER:

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	504
1.0	0.88	1.00	0.94	504
2.0	1.00	0.86	0.93	504
3.0	1.00	1.00	1.00	505
4.0	1.00	1.00	1.00	504
accuracy			0.97	2521
macro avg	0.98	0.97	0.97	2521
weighted avg	0.98	0.97	0.97	2521

```

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF SGD CLASSIFIER:\n\n\n',cm)

```

THE CONFUSION MATRIX SCORE OF SGD CLASSIFIER:

```

[[504  0  0  0  0]
 [ 0 504  0  0  0]
 [ 0 69 435  0  0]
 [ 0  0  0 505  0]
 [ 0  0  0  0 504]]

```

```

from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(SGT, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',
accuracy*100)

```

THE CROSS VALIDATION TEST RESULT OF ACCURACY :

IMPLEMENTATION SCREENSHOTS

```
[96.3506545 96.5093217 99.04799683 99.8413328 96.98532328]

from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF SGD CLASSIFIER IS :",accuracy.mean()*100)

THE ACCURACY SCORE OF SGD CLASSIFIER IS : 97.74692582308609

from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF SGD CLASSIFIER IS :",hl*100)

THE HAMMING LOSS OF SGD CLASSIFIER IS : 2.7370091233637446

def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF SGD
CLASSIFIER\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF SGD CLASSIFIER:\n\n')
print(cm)
plot_confusion_matrix(cm)

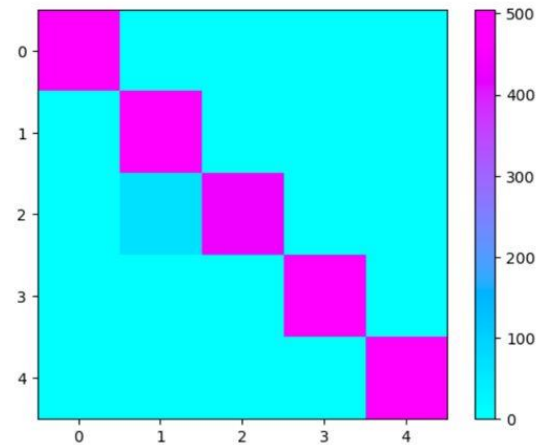
THE CONFUSION MATRIX SCORE OF SGD CLASSIFIER:
```

```
[[504  0  0  0  0]
 [  0 504  0  0  0]
 [  0 69 435  0  0]
 [  0  0  0 505  0]
 [  0  0  0  0 504]]
```

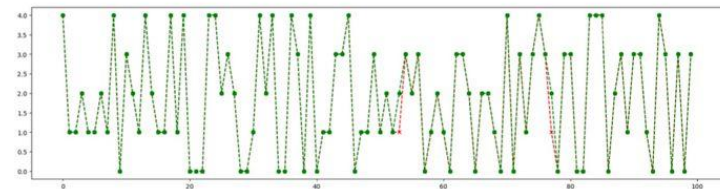
```
import joblib
joblib.dump(SGT, 'MODEL.pkl')

['MODEL.pkl']
```

THE CONFUSION MATRIX SCORE OF SGD CLASSIFIER



```
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed',
color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed',
color='green')
plt.show()
```



IMPLEMENTATION SCREENSHOTS

LINEAR SUPPORT VECTOR MACHINE ALGORITHM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('STRESS.csv')
df.head()
```

	snoring rate	respiration rate	body temperature	limb movement \
0	93.80	25.680	91.840	16.600
1	91.64	25.104	91.552	15.880
2	60.00	20.000	96.000	10.000
3	85.76	23.536	90.768	13.920
4	48.12	17.248	97.872	6.496

	blood oxygen	retina eye movement	sleeping rate	heart rate
stress level				
0	89.840	99.60	1.840	74.20
3.0				
1	89.552	98.88	1.552	72.76
3.0				
2	95.000	85.00	7.000	60.00
1.0				
3	88.768	96.92	0.768	68.84
3.0				
4	96.248	72.48	8.248	53.12
0.0				

```
df=df.dropna()

df.columns

Index(['snoring rate', 'respiration rate', 'body temperature', 'limb
movement',
      'blood oxygen', 'retina eye movement', 'sleeping rate', 'heart
rate',
      'stress level'],
      dtype='object')

df.head()
```


IMPLEMENTATION SCREENSHOTS

```

snoring rate  respiration rate  body temperature  limb movement  \
0      93.80      25.680      91.840      16.600
1      91.64      25.104      91.552      15.880
2      60.00      20.000      96.000      10.000
3      85.76      23.536      90.768      13.920
4      48.12      17.248      97.872      6.496

blood oxygen  retina eye movement  sleeping rate  heart rate
stress level
0      89.840      99.60      1.840      74.20
3.0      89.552      98.88      1.552      72.76
1      95.000      85.00      7.000      60.00
2      88.768      96.92      0.768      68.84
3.0      96.248      72.48      8.248      53.12
0.0

x1 = df.drop(labels='stress level', axis=1)
y1 = df.loc[:, 'stress level']

import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros = RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      : ", Counter(y1))
print("OVER SAMPLING DATA COUNT : ", Counter(y))

OUR DATASET COUNT      : Counter({0.0: 2521, 4.0: 2403, 2.0: 2322,
3.0: 2287, 1.0: 2251})
OVER SAMPLING DATA COUNT : Counter({3.0: 2521, 1.0: 2521, 0.0: 2521,
2.0: 2521, 4.0: 2521})

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.20, random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET      : ", len(x_train))
print("NUMBER OF TEST DATASET       : ", len(x_test))
print("TOTAL NUMBER OF DATASET      : ", len(x_train)+len(x_test))

NUMBER OF TRAIN DATASET      : 10084
NUMBER OF TEST DATASET       : 2521
TOTAL NUMBER OF DATASET      : 12605

print("NUMBER OF TRAIN DATASET      : ", len(y_train))
print("NUMBER OF TEST DATASET       : ", len(y_test))
print("TOTAL NUMBER OF DATASET      : ", len(y_train)+len(y_test))

```

```

NUMBER OF TRAIN DATASET      : 10084
NUMBER OF TEST DATASET       : 2521
TOTAL NUMBER OF DATASET      : 12605

from sklearn.svm import LinearSVC

lr = LinearSVC()
lr.fit(x_train,y_train)

LinearSVC()

predicted = lr.predict(x_test)

from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF LINEAR SUPPORT VECTOR MACHINE:\n\
n',cr)

THE CLASSIFICATION REPORT OF LINEAR SUPPORT VECTOR MACHINE:

              precision    recall  f1-score   support

0.0               1.00               1.00               1.00         504
1.0               1.00               1.00               1.00         504
2.0               0.89               1.00               0.94         504
3.0               1.00               0.88               0.93         505
4.0               1.00               1.00               1.00         504

accuracy               0.98
macro avg              0.98               0.98               0.98         2521
weighted avg           0.98               0.98               0.98         2521

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF LINEAR SUPPORT VECTOR MACHINE:\n\
n',cm)

THE CONFUSION MATRIX SCORE OF LINEAR SUPPORT VECTOR MACHINE:

[[504  0  0  0  0]
 [ 0 504  0  0  0]
 [ 0  0 504  0  0]
 [ 0  0 62 443  0]
 [ 0  0  0  0 504]]

from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(lr, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n',
accuracy*100)

```

IMPLEMENTATION SCREENSHOTS

THE CROSS VALIDATION TEST RESULT OF ACCURACY :

```
[94.60531535 98.73066244 91.90797303 97.46132487 92.9789766 ]
```

```
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF LINEAR SUPPORT VECTOR MACHINE
IS :",accuracy.mean()*100)
```

THE ACCURACY SCORE OF LINEAR SUPPORT VECTOR MACHINE IS :
95.13685045616818

```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF LINEAR SUPPORT VECTOR MACHINE IS :",hl*100)
```

THE HAMMING LOSS OF LINEAR SUPPORT VECTOR MACHINE IS :
2.459341531138437

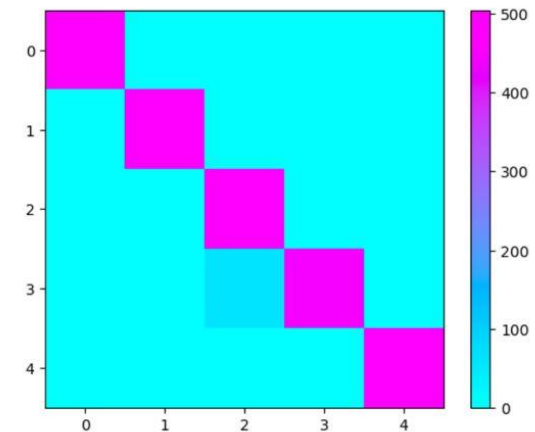
```
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
LINEAR SUPPORT VECTOR MACHINE\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
```

```
cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF LINEAR SUPPORT VECTOR MACHINE:\n\
n')
print(cm)
plot_confusion_matrix(cm)
```

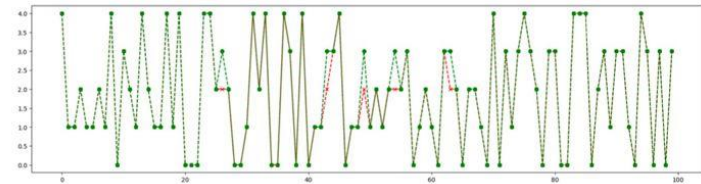
THE CONFUSION MATRIX SCORE OF LINEAR SUPPORT VECTOR MACHINE:

```
[[504  0  0  0  0]
 [ 0 504  0  0  0]
 [ 0  0 504  0  0]
 [ 0  0 62 443  0]
 [ 0  0  0  0 504]]
```

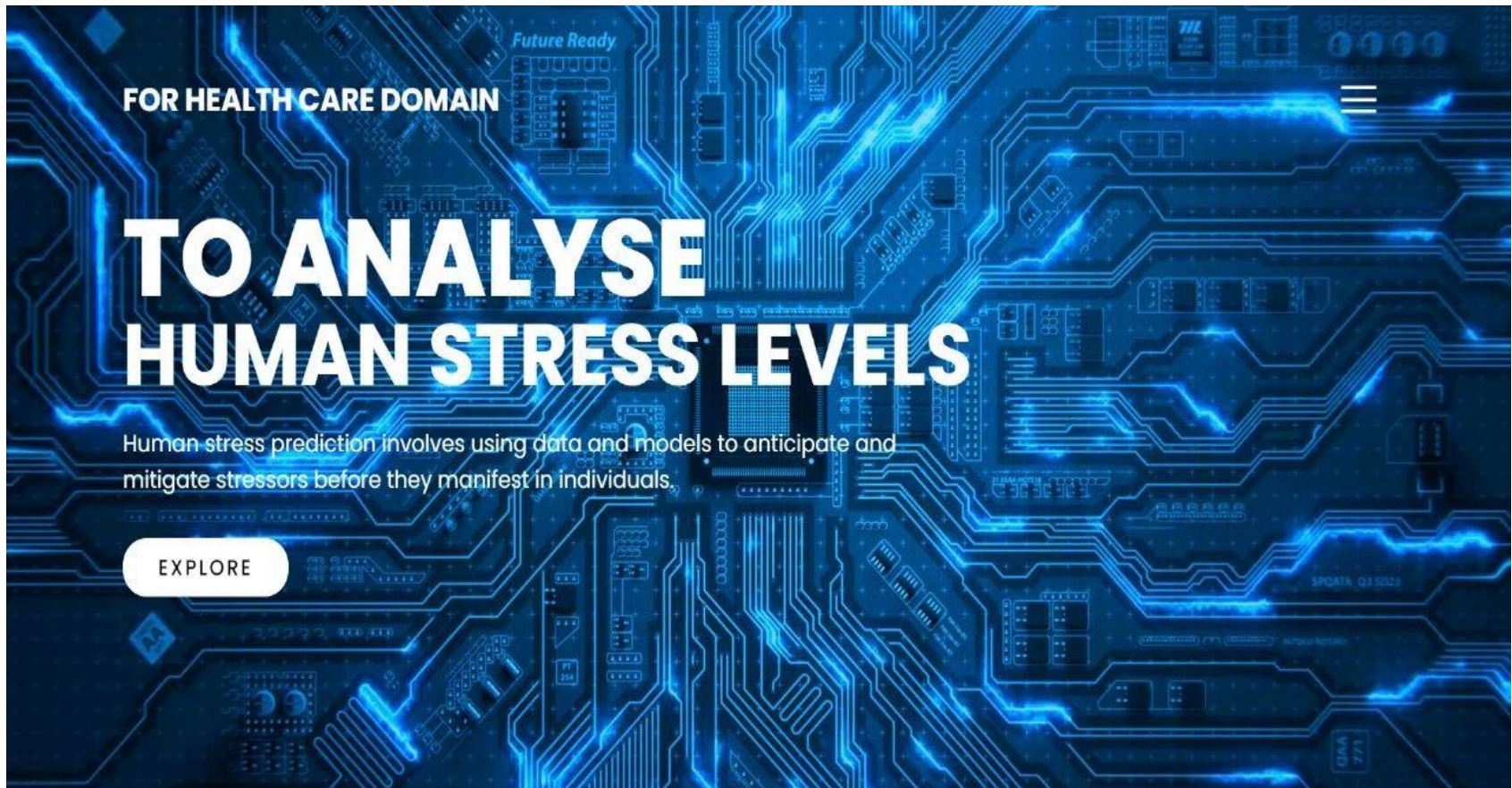
THE CONFUSION MATRIX SCORE OF LINEAR SUPPORT VECTOR MACHINE



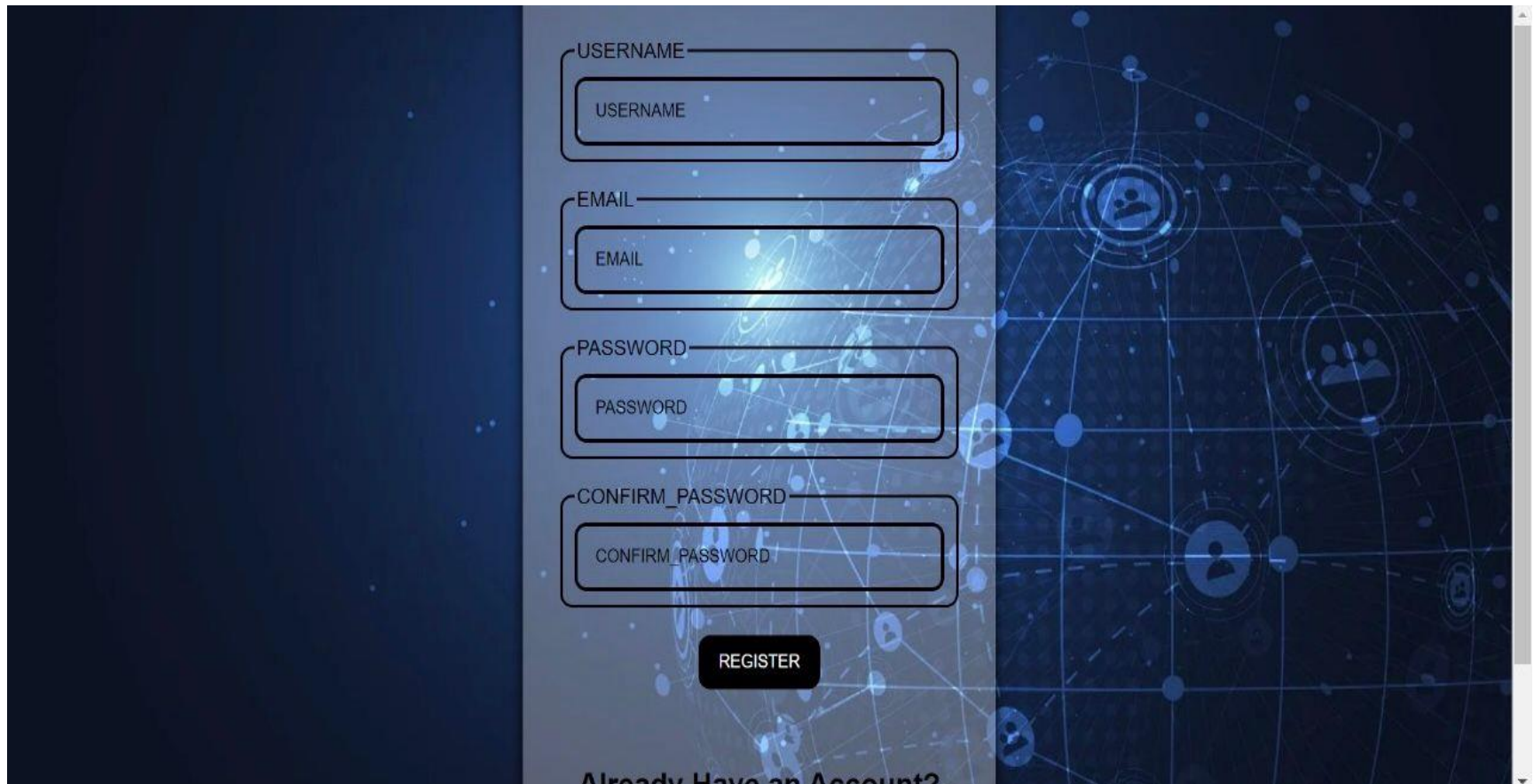
```
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed',
color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed',
color='green')
plt.show()
```



APPLICATION SCREENSHOTS



APPLICATION SCREENSHOTS



A screenshot of a web application registration form. The form is centered on a dark blue background with a network diagram. It contains four input fields: USERNAME, EMAIL, PASSWORD, and CONFIRM_PASSWORD, each with a label above it. Below the fields is a black REGISTER button. At the bottom, there is a link that says "Already Have an Account?".

USERNAME

EMAIL

PASSWORD

CONFIRM_PASSWORD

REGISTER

Already Have an Account?

APPLICATION SCREENSHOTS

Click to go back, hold to see history

LOGIN

USERNAME

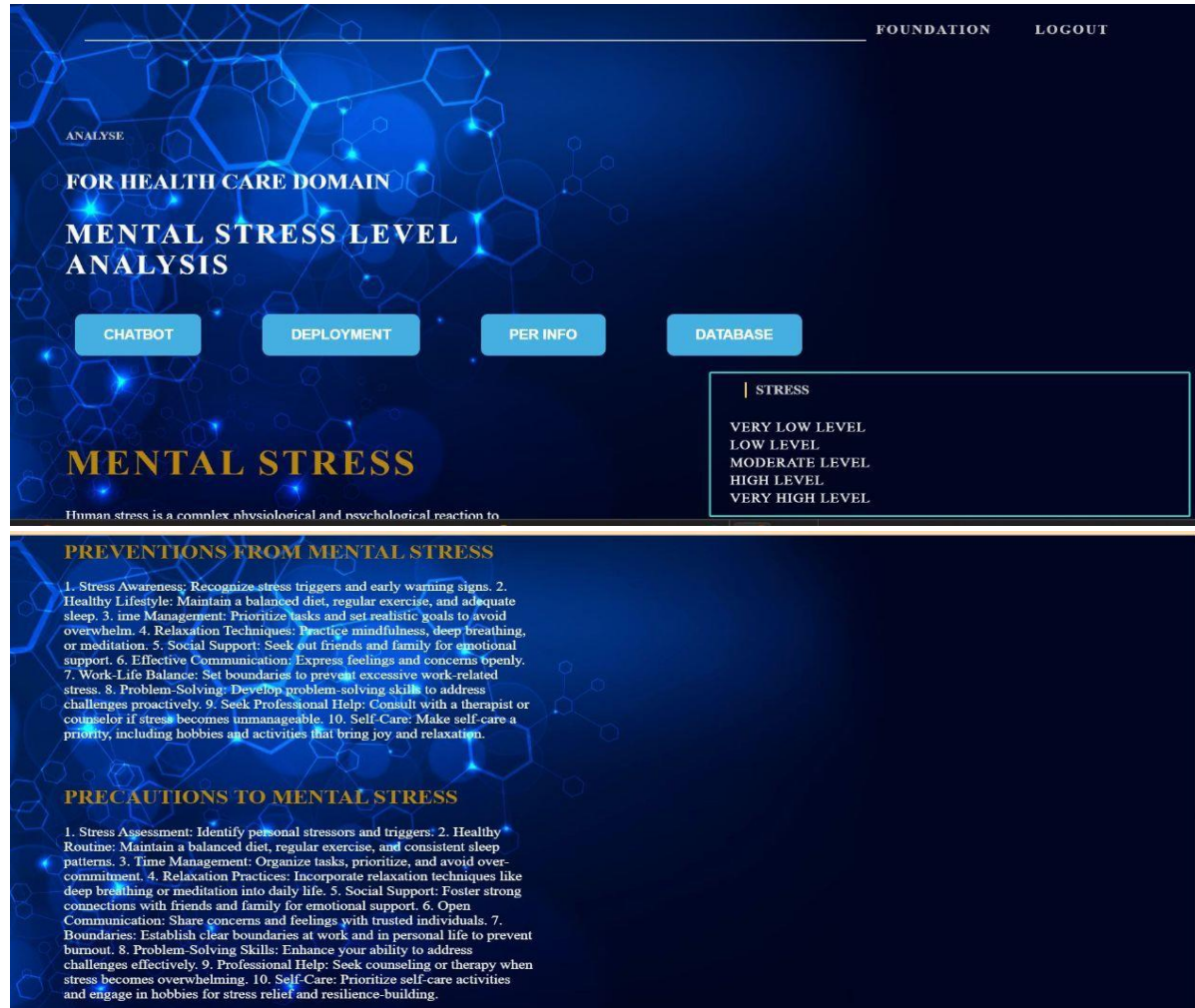
PASSWORD

LOGIN

Don't Have an Account?

[REGISTER](#)

APPLICATION SCREENSHOTS



APPLICATION SCREENSHOTS

PERSONAL REGISTRATION FORM

FIRST_NAME

LAST_NAME

AGE

CITY

STATE

NATIONALITY

APPLICATION SCREENSHOTS

PERSONAL INFO DATABASE

firstname	lastname	age	address	phone	city	state	country
Jessica	Janet	20	SVCE	7826928883	Chennai	Tamil nadu	Indian

APPLICATION SCREENSHOTS

AUTOMATED STRESS LEVEL ASSESSMENT BY EXPLORING PREDICTIVE TECHNIQUES

SNORING_RATE

RESPIRATION_RATE

BODY_TEMPERATURE

LIMB_MOVEMENT

BLOOD_OXYGEN

EYE_BALL_MOVEMENT

SLEEPING_RATE

HEART_RATE

PREDICT

RESULT

APPLICATION SCREENSHOTS

The screenshot shows a web application interface for stress level assessment. The title is "AUTOMATED STRESS LEVEL ASSESSMENT BY EXPLORING PREDICTIVE TECHNIQUES". The interface features a dark blue background with a circuit-like pattern. On the left, there are eight input fields with their respective values: SNORING_RATE (56), RESPIRATION_RATE (68), BODY_TEMPERATURE (97), LIMB_MOVEMENT (0.4), BLOOD_OXYGEN (98), EYE_BALL_MOVEMENT (75), SLEEPING_RATE (8.2), and HEART_RATE (63). A large white button labeled "RESULT" is on the right. At the bottom, there is a blue button labeled "PREDICT".

AUTOMATED STRESS LEVEL ASSESSMENT BY EXPLORING PREDICTIVE TECHNIQUES

Parameter	Value
SNORING_RATE	56
RESPIRATION_RATE	68
BODY_TEMPERATURE	97
LIMB_MOVEMENT	0.4
BLOOD_OXYGEN	98
EYE_BALL_MOVEMENT	75
SLEEPING_RATE	8.2
HEART_RATE	63

RESULT

PREDICT

APPLICATION SCREENSHOTS



APPLICATION SCREENSHOTS

AUTOMATED STRESS LEVEL ASSESSMENT BY EXPLORING PREDICTIVE TECHNIQUES

[HOME](#)

STRESS LEVEL:

THE LESS DEPRESSION MIGHT BE OCCUR IN THIS CONDITIONS, THIS IS STAGE 1 DEPRESSION LEVEL.

PREVENTIONS : Regular Exercise: Physical activity can reduce stress hormones and increase endorphins, improving mood and overall health. Healthy Diet: Consuming a balanced diet rich in fruits, vegetables, lean proteins, and whole grains can support stress reduction. Adequate Sleep: Ensure you get 7-9 hours of quality sleep each night to rejuvenate your body and mind. Mindfulness Meditation: Practicing mindfulness techniques can help manage stress by focusing on the present moment and reducing anxiety. Deep Breathing Exercises: Engage in deep breathing exercises to activate the body's relaxation response and calm the mind.

APPLICATION SCREENSHOTS

AUTOMATED STRESS LEVEL ASSESSMENT BY EXPLORING PREDICTIVE TECHNIQUES

[HOME](#)

STRESS LEVEL:

THE HIGH DEPRESSION MIGHT BE OCCUR IN THIS CONDITIONS. THIS IS STAGE 3 DEPRESSION LEVEL.

PREVENTIONS : Express Gratitude: Focus on what you're grateful for each day to shift your mindset towards positivity. Humor and Laughter: Find opportunities for humor and laughter, as they can lighten your mood and relieve tension. Hobbies and Leisure Activities: Engage in activities you enjoy to relax and recharge outside of work or responsibilities. Seek Professional Help: If stress becomes overwhelming, consider seeking support from a therapist or counselor. Practice Assertiveness: Communicate your needs and concerns assertively, rather than bottling up emotions.

APPLICATION SCREENSHOTS

AUTOMATED STRESS LEVEL ASSESSMENT BY EXPLORING PREDICTIVE TECHNIQUES

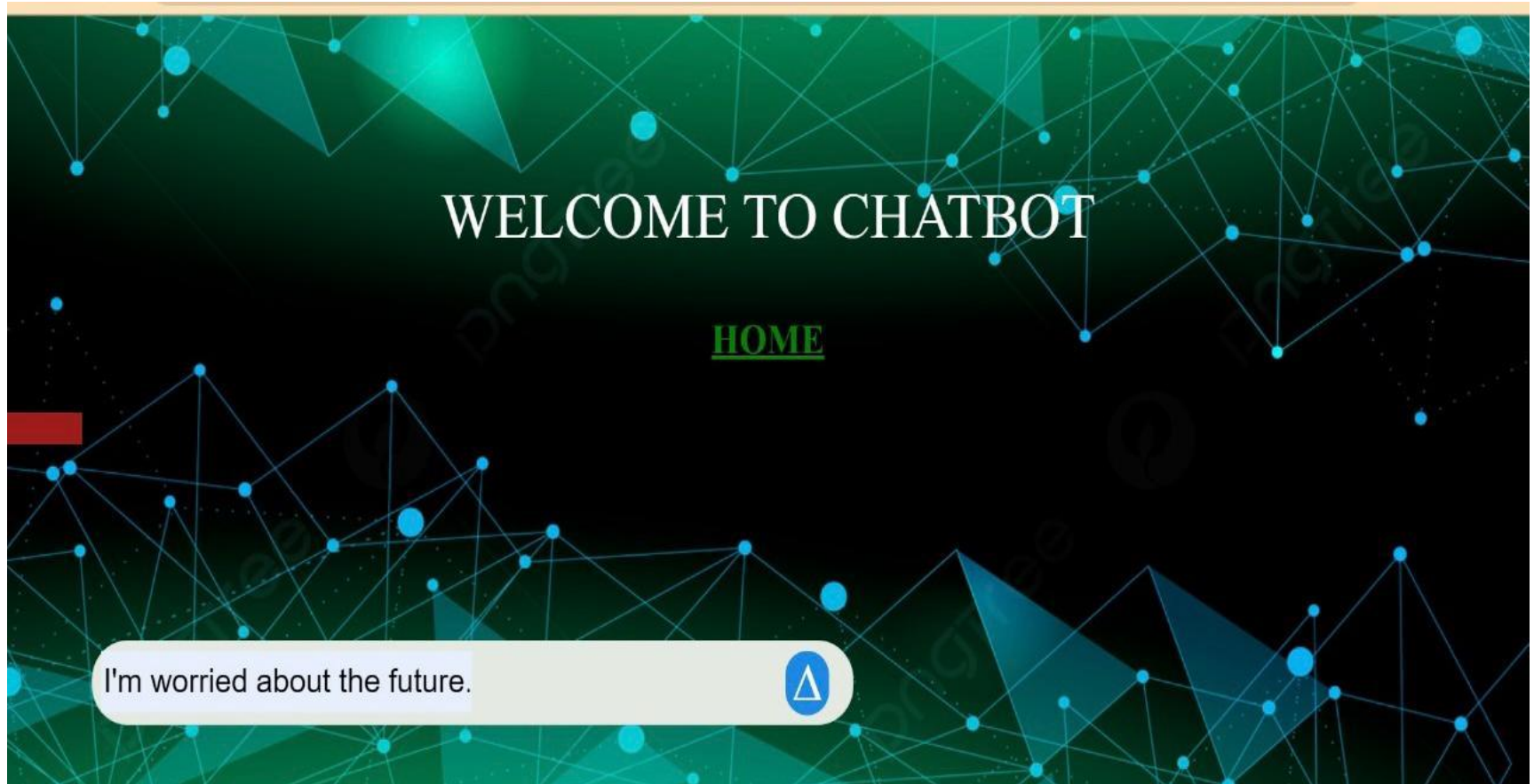
[HOME](#)

STRESS LEVEL:

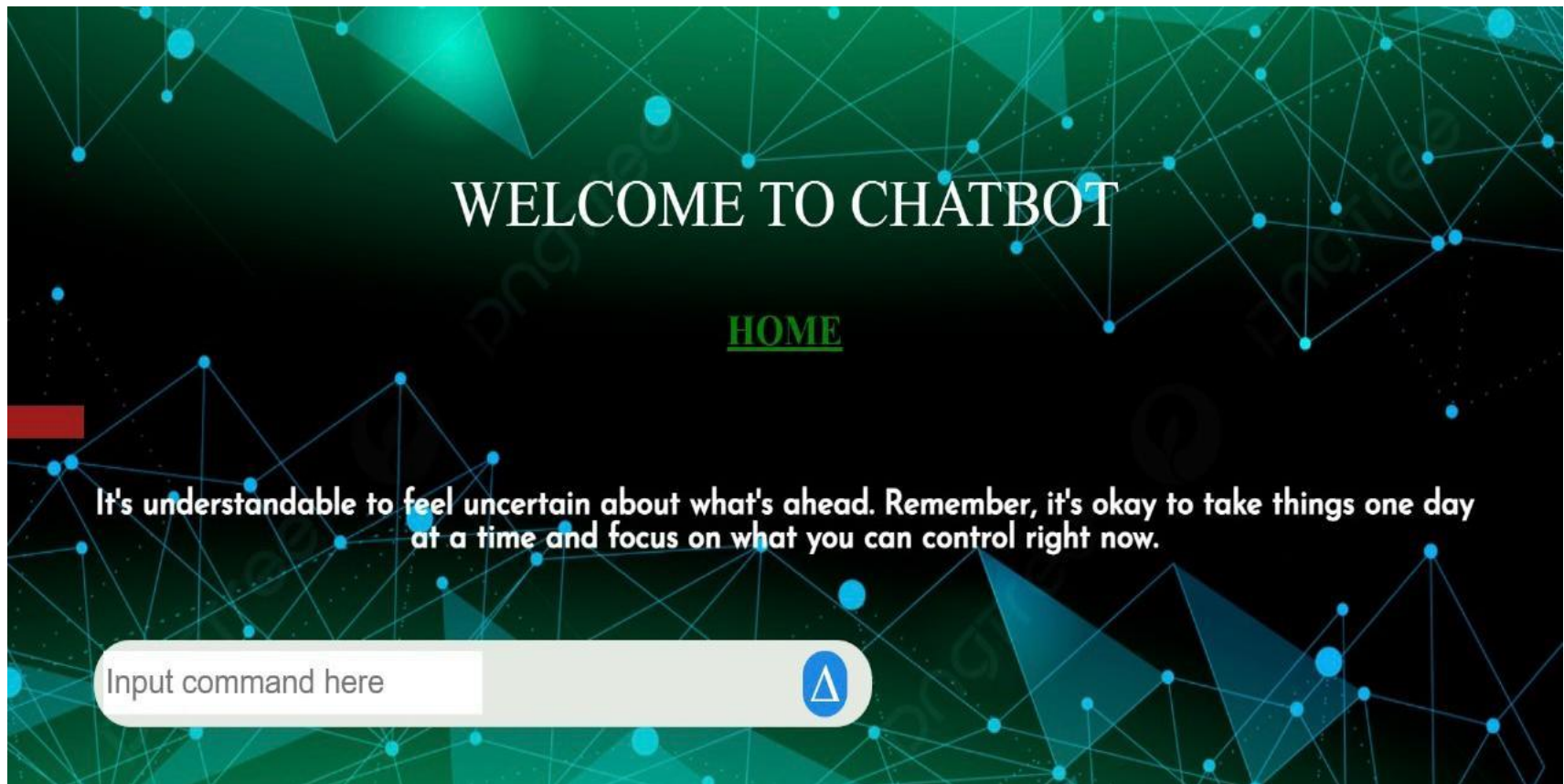
THE VERY HIGH DEPRESSION MIGHT BE OCCUR IN THIS CONDITIONS. THIS IS STAGE 4 DEPRESSION LEVEL.

PREVENTIONS : Avoid Procrastination: Break tasks into smaller, manageable steps and tackle them gradually to reduce stress from looming deadlines. Stay Organized: Maintain a clutter-free environment and keep things organized to minimize stress from disorganization. Nature Walks: Spending time outdoors, especially in nature, can promote relaxation and reduce stress levels. Limit Screen Time: Reduce exposure to screens, especially before bedtime, to improve sleep quality and reduce stress. Self-Care Rituals: Incorporate self-care practices into your daily routine, such as taking baths, reading, or practicing hobbies that bring you joy.

APPLICATION SCREENSHOTS



APPLICATION SCREENSHOTS



REFERENCES

BASE PAPER:

Park, J., Ahn, H., Youn, K., Lee, M., & Hong, S. (2023). Ensemble Learning to Identify Depression Indicators for Korean Farmers. IEEE Access, 11, 118787-118800.

REFERENCES:

1. Magtibay, Karl, and Karthikeyan Umapathy. “ (2023). A Review of Tools and Methods for Detection, Analysis, and Prediction of Allostatic Load due to Workplace Stress.” IEEE Transactions on Affective Computing, 1-22.
2. Dennis Rodrigo, Zelun Wang, and Ricardo Gutierrez-Osuna. (2021) "Towards participant-independent stress detection using instrumented peripherals." IEEE Transactions on Affective Computing.
3. Yasmin, Hena, Salman Khalil, and Ramsha Mazhar. (2020) "COVID 19: Stress management among students and its impact on their effective learning." International technology and education journal 4.2, 65-74.
4. Chouhan, Devraj Singh. (2016) "Stress and Its Major Effects on Human Health." International Journal of Multidisciplinary Allied Research Review and Practices.
5. Shahsavarani, et al. (2015) "Stress: Facts and theories through literature review." International Journal of Medical Reviews 2.2, 230-241.

