

Human Stress Detection using a User Dependent Model by Deploying Supervised Learning Techniques

R. Jai Sakthi¹, Bhavana M², Rajeev Kumar Dubey³, Dr. P. Durgadevi⁴

Students, SRM Institute of Science and Technology, Vadapalani Campus^{1,2,3}

Assistant Professor, SRM Institute of Science and Technology, Vadapalani Campus⁴

Abstract: *Stress is a natural response to challenges and hazards, but chronic stress can lead to various health problems. Managing emotions and time can help mitigate stress, and if it persists, coping mechanisms can be developed. Stress measurement has become increasingly important, and we propose a supervised learning-based model that gathers physiological data from users in various stress scenarios. Our machine learning model, which accurately categorizes a user's stress levels, is specific to each user. We evaluate our model's effectiveness using several supervised learning algorithms, and our research shows that our user-specific approach outperforms traditional user-independent models, with accuracy rates of up to 95% in stress classification. Personalized variables can significantly improve stress detection models, which can have important implications for developing personalized stress management solutions*

Keywords: Stress Detection, Deployment, Supervised Learning Techniques, Support Vector Machine Classifier, Decision Tree Classifier, Adaboost Classifier, MLP Classifier.

I. INTRODUCTION

Stress is a widespread problem that can significantly impact an individual's health. Machine learning algorithms can help in identifying stress through physiological signals, leading to timely management of stress and improved well-being. This study aims to develop a personalized stress detection model using supervised machine learning techniques, including Support Vector Machines, Decision Tree Classifier, AdaBoost, and MLP Classifier. The effectiveness of these algorithms will be evaluated, and potential applications in managing stress will be explored. Furthermore, the study will investigate how this model can be used to identify stress in children with mental health conditions to prevent complications. The ultimate goal is to develop effective stress management tools that can enhance overall well-being.

II. LITERATURE SURVEY

N. Madjar et al[1] and F.Lindblom et al[2] investigated the stress detection and measure selection in a medical technology company using unsupervised learning techniques and algorithms. The study shows that machine learning algorithms can identify behavioral patterns and correlations in data points to classify stress, leading to improved stress detection procedures and methodologies. The results can enhance the stress-bot's capabilities to accurately and efficiently analyze client data. Further research with larger datasets is recommended to achieve optimal results. This article highlights the potential application of machine learning principles in the medical technology industry to improve stress detection and measure selection.

E. Padma et al[3], Talapaneni Paveen et al[4] and Shaik Karimulla et al[5] did the study that uses machine learning and image processing to detect stress and proposed a novel standard for stress detection by including additional physiological parameters. The authors collected wearable sensing data in an office environment to evaluate the accuracy of their system and reference related studies. The research provides insights into state-of-the-art techniques for stress detection and suggests incorporating image processing and machine learning approaches for stress management in different industries. Overall, this research provides valuable insights into advanced approaches for stress detection and management in various industries, highlighting the potential benefits of incorporating image processing and machine learning techniques.

Z.Zainudin et al[6], S.Hasan et al[7] and S.M. Shamsuddin et al[8] investigated the use of machine learning and deep learning techniques for detecting stress, a prevalent issue in society. The research explores various physiological and behavioral changes associated with stress and examines different types of data that can be utilized for stress identification. The authors also discuss the challenges associated with stress detection and potential applications of stress detection in different industries. This paper offers valuable insights into the potential benefits of machine learning and deep learning for stress detection and their potential role in improving human well-being.

III. OBJECTIVE

The primary aim of this study was to develop a machine learning model for stress detection to replace outdated supervised classification models. Four popular algorithms - SVM, Decision Tree Classifier, AdaBoost, and MLP Classifier - were evaluated based on metrics like accuracy, precision, recall, and F1 score, using cross-validation techniques to avoid overfitting. The best-performing algorithm was selected for further analysis and real-world deployment, potentially improving stress detection and management strategies.

After thorough evaluation, the algorithm that offered the best accuracy in stress detection prediction was selected for further analysis and deployment in real-world scenarios. This study's findings could help enhance the accuracy of stress detection techniques, leading to more effective stress management strategies in various fields.

IV. PROPOSED SYSTEM

The dataset is then cleaned and examined for accuracy before being used for analysis. The collected data is split into a training set and a test set for forecasting purposes, usually in a 7:3 ratio. Machine learning techniques are used to create a data model, which is then applied to the training set. The test set is predicted based on the accuracy of the test results. The ML prediction model is effective at pre-processing irrelevant data, outliers, and different types of variables.

Advantages:

- Accuracy may be improvised.
- Performance metrics will be compared.
- The project will be deployed.

V. SYSTEM ARCHITECTURE

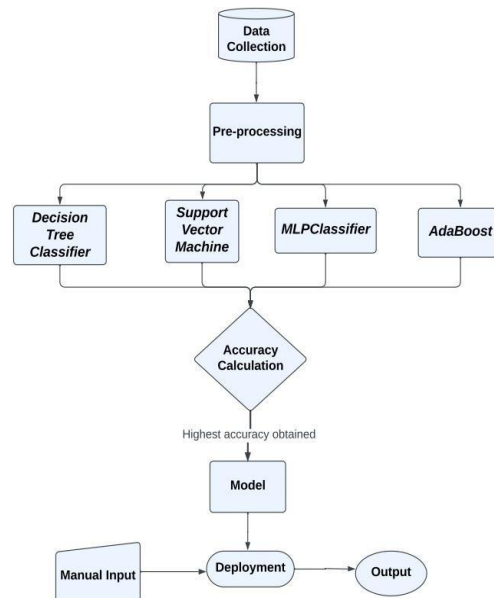


Fig 1: Diagramatic representation of System Architecture

In Fig 1, the process of the entire process is given as an architecture diagram. The processes are explained in the sections given below.

Data Collection

The dataset was taken from Kaggle, which was created by certified doctors. The dataset consists of 630 entries for patients with the nine parameters given in the table below:

1.	SR	SLEEPING RATE
2.	RR	RESPIRATION RATE
3.	T	BODY TEMPERATURE
4.	LM	LIMB MOVEMENT
5.	BO	BODY OXYGEN
6.	REM	EYE MOVEMENT
7.	SH	SLEEPING HOURS
8.	HR	HEART RATE
9.	SL	STRESS LEVEL

Table 1: Parameter’s Abbreviation

The data set is separated into a training set and a test set for prediction. Typically, the training set to test set ratio is 7:3. The decision tree method, MLP Classifier algorithm, AdaBoost technique, and Support Vector Machine algorithm were all used in the testing phase in order to predict the test set.

Data Pre-processing

Missing values in the collected data are a possibility, which could lead to inconsistencies in the results. To improve the algorithm's effectiveness and deliver better results, prior processing of the information is required.

Building the classification model

In our study, we have employed several supervised learning techniques, including Decision Tree Classifier, Support vector machine, Adaboost, and MLPClassifier to construct a high-accuracy stress prediction model.

Additionally, the model is proficient in dealing with inconsistent data pre-processing, limited variables, and a combination of continuous, categorical, and discrete data. While there may be out-of-bag estimation errors, they do not significantly affect the model's performance and can be easily corrected based on results from various tests.

Accuracy Calculation

Through continuous refinement, a fine-tuned model achieves higher levels of precision, and a thorough comparison of all four algorithms determines the most optimal algorithm as the ultimate choice.

For doing it, we are using the formula:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Model

Following the identification of the algorithm with the best accuracy, we proceeded to save the trained model as a .pkl file, which is a type of Python Pickle File. This conversion enabled us to move forward to the next phase of the process, which involves deploying the model.

Deploying The Model

To enhance the user interface and preempt human fatigue, the module converts the trained machine learning model into a data format file called a pickle (.pkl) file.



Fig 2: Deployment Workflow

In Fig 2, the deployment workflow initially imports packages and later reads the PKL files and input data. Finally the predicted value prints as the output.

In order to deploy the model, we are using the following techniques:

- Django (Web Framework)
- HTML
- CSS

VI. MODULES AND IMPLEMENTATION

The Modules that we are going to explain in this section are:

- Data Pre-processing
- Algorithm1-**DecisionTree Classifier**
- Algorithm 2-**SVM**
- Algorithm 3-**AdaBoost**
- Algorithm 4-**MLPClassifier**

Deployment-**Django**(Web FrameWork)

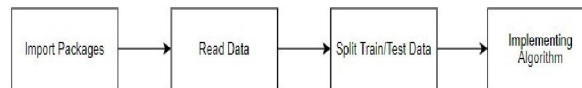


Fig 3: Algorithm Workflow

In Fig 3, the algorithm workflow initially imports packages and later reads the data and splits the train and test data in 8:2 ratio. Finally the implementation of the algorithm takes place.

Data Pre-processing

Data preprocessing is a crucial step in machine learning that involves cleansing and converting raw data into a more manageable format that can be easily analyzed by computers. This step is an essential component of a complete machine learning pipeline.

The accuracy and ability of machine learning models to be applied to new datasets largely depend on the quality of the data used during training. To guarantee that the data is in the appropriate format, pre-processing is a critical step.

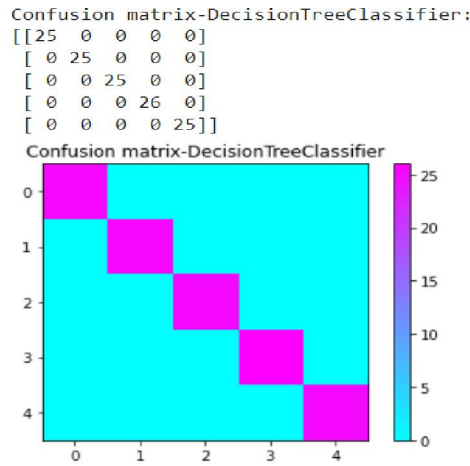
Decision Tree Classifier

The data was divided into two sets, with 80% of the data allocated for training and the remaining 20% reserved for testing. This division was stratified based solely on the target variable y to ensure that the class proportions were maintained.

The decision tree classifier model was applied to the test set using the `classification_report()` function.

To evaluate the effectiveness of the `DecisionTreeClassifier` model, the `cross_val_score()` function was used, which computes an array of accuracy scores. These scores were then used to determine the model's ability to generalize

In Fig 4, the resulting confusion matrix aids in evaluating the model's classification performance by displaying the number of correct and incorrect classifications for each class. The resultant accuracy comes to 85.55%.



Accuracy Result of DecisionTreeClassifier is: 85.55

Fig 4: Confusion Matrix of Decision Tree Classifier

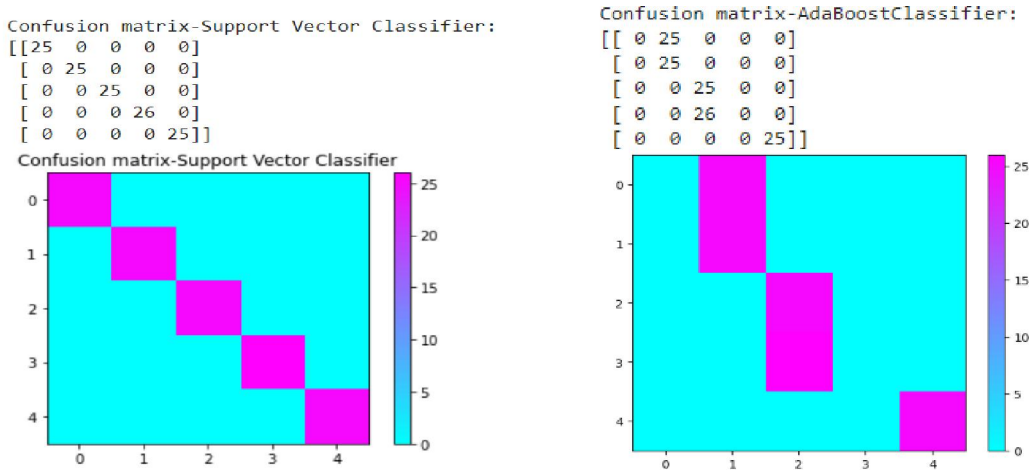
Support Vector Machine- SVM Classifier

To train and test an SVM classifier, all Sklearn components and classes were imported. The classifier's performance was evaluated using the accuracy score, confusion matrix, and classification report functions.

The last step of the analysis involved a function called plot_confusion_matrix, which utilizes the plt module to generate a visual representation of the confusion matrix.

The function accepts a confusion matrix (CM) as well as several optional parameters to create the matrix cm1 using the confusion_matrix() function, with the predicted labels and true labels y_test as inputs. This allows for the calculation of both the confusion matrix and accuracy.

In Fig 5, the resulting confusion matrix aids in evaluating the model's classification performance by displaying the number of correct and incorrect classifications for each class. The resultant accuracy comes to 94.09%.



Accuracy Result of Support Vector Classifier is: 94.09

Accuracy Result of AdaBoostClassifier is: 51.58

Fig 5: Confusion Matrix of Support Vector Machine

AdaBoost- Adaptive Boosting Classifier

The study utilized the AdaBoost algorithm in the Scikit-Learn library to construct and evaluate the classifier. To evaluate the performance of the classifier, various functions such as the confusion matrix, classification report, accuracy score, and cross-val score were employed. The AdaBoostClassifier class was utilized to implement the AdaBoost algorithm, which is a classification ensemble method.

AdaBoost classifier's effectiveness on the test data. The report displays accuracy, recall, F1-score, and support metrics for each class in the target variable. Using the plot function for the confusion matrix, the cm1 matrix is displayed in a visual format through the use of the imshow function from the matplotlib library. In Fig 6, the resulting confusion matrix aids in evaluating the model's classification performance by displaying the number of correct and incorrect classifications for each class. The resultant accuracy comes to 51.58%.

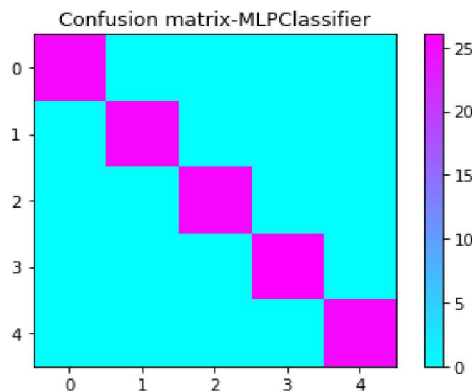
MLP Classifier

To create an instance of the MLP classifier with default parameters, the MLPClassifier function is used. The accuracy for each fold was computed and averaged to achieve a more precise evaluation. The plot_confusion_matrix function visualized the confusion matrix by displaying the mean accuracy as a percentage.

Finally, the analysis produced the accuracy and confusion matrix, and it was found that the MLPClassifier algorithm had the highest accuracy among the four algorithms

Consequently, the trained model was saved as a .pkl file used in the next stage which is the deployment of the model.

```
Confusion matrix-MLPClassifier:
[[25  0  0  0  0]
 [ 0 25  0  0  0]
 [ 0  0 25  0  0]
 [ 0  0  0 26  0]
 [ 0  0  0  0 25]]
```



Accuracy Result of MLPClassifier is: 100.0

Fig 7: Confusion Matrix of MLPClassifier

In Fig 7, the resulting confusion matrix aids in evaluating the model's classification performance by displaying the number of correct and incorrect classifications for each class. The resultant accuracy comes to 100%.

Graphical Representation- Accuracy Comparison

In Fig 8, the accuracy comparison is shown as a graphical representation.

The algorithm with the least accuracy was found to be Adaboost, with 51.58% accuracy

The decision tree algorithm had an accuracy of 85.55% .

Next was the Vector Machine with 94.09% accuracy.

And finally, the algorithm with the highest accuracy was MLP with 100% accuracy.

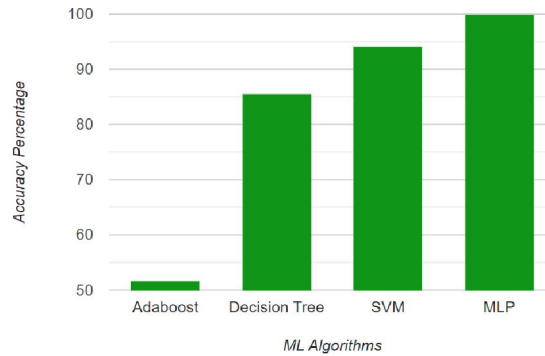


Fig 8: Graphical representation of the accuracies obtained from the four machine learning algorithms. For the deployment stage, the trained model of the MLP algorithm, the algorithm with the highest accuracy, was converted to a.pkl file for building the website.

Deployment- Django (Web FrameWork)

The core component of Django is the WSGI application, which serves as a central hub for the application's modules and functionality. The package or module name is used to locate resources within the package or directory. The Django instance is created in the main module or in the init.py file of the package. This instance serves as a central registry for various components, including view functions, URL rules, and template configurations. Overall, the Django object streamlines the development process and simplifies the management of various components.

USER LOGIN

Username:

Password:

[Login](#)

Dont have an account? [signup](#)

PSYCHIATRIST LOGIN

Username:

Password:

[Login](#)

Dont have an account? [signup](#)

Fig 9: User Login/Signup

Fig 10: Psychiatrist Login/Signup

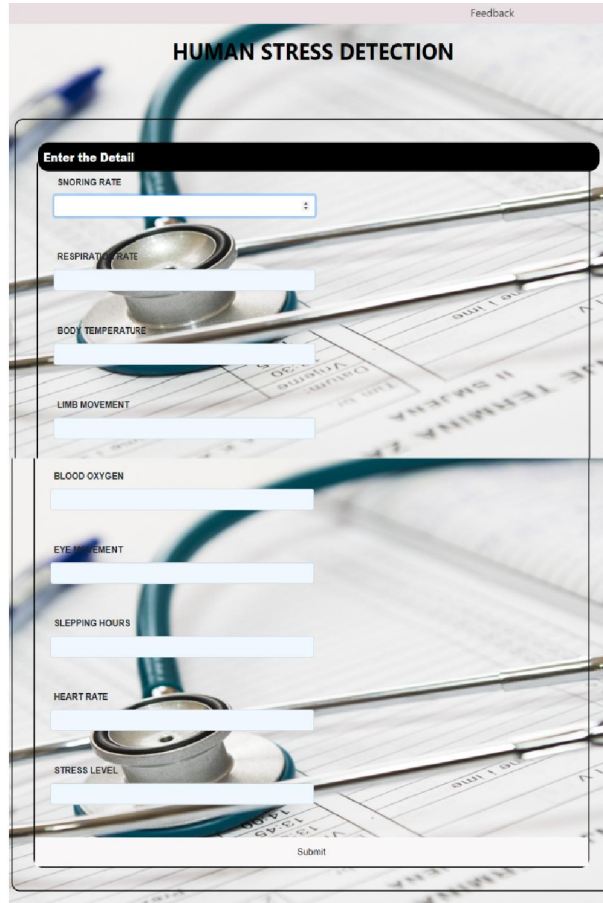


Fig 11: Details that are entered by Psychiatrist after they login.

In Fig 9, 10 & 11; the deployment is shown using the Django framework.

VII. CONCLUSION AND FUTURE WORK

After the data was cleaned and processed, it was stored in a .pkl file and deployed on a website for psychiatrists to access. Through this web platform, psychiatrists can input nine parameters to determine the stress level of their patients. The website uses the pre-saved .pkl file to produce a precise assessment of the patient's stress level. This website is a convenient and efficient way for psychiatrists to evaluate their patient's stress levels based on specific parameters.

In addition, the utilization of a two-login website, one for patients and one for psychiatrists, can establish a secure and convenient communication channel between the patient and their healthcare provider, which may enhance patient outcomes and facilitate the development of more efficient and effective treatment plans.

In brief, employing advanced algorithms like MLPClassifier and utilizing web-based platforms can transform the approach to assessing and handling stress levels in patients, leading to better health outcomes and an enhanced quality of life for individuals experiencing stress-related conditions.

A. Future Work:

- Deploying the project in the cloud.
- To implement in the IOT system

REFERENCES

- [1] Madjar, N., & Lindblom, F. (2020). Machine Learning implementation for Stress-Detection (Dissertation). Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-280897>
- [2] F. Lindblom, 'Machine Learning implementation for Stress-Detection', Dissertation, 2020.
- [3] E. PADMA, <https://jespublication.com/upload/2022-V13I3053.pdf>, Vol 13, Issue 03, MARCH/2022, ISSN NO:0377-9254
- [4] TALAPANENI PRAVEEN, <https://jespublication.com/upload/2022-V13I3053.pdf>, Vol 13, Issue 03, MARCH/2022, ISSN NO:0377-9254
- [5] SHAIK KARIMULLA, <https://jespublication.com/upload/2022-V13I3053.pdf>, Vol 13, Issue 03, MARCH/2022, ISSN NO:0377-9254
- [6] Z. Zainudin et al 2021 J. Phys.: Conf. Ser. 1997 012019