Principal Component Analysis and Human Stress Detection using Machine Learning

Sumaiya Iqbal, ID: 40221319
GitHub Link: https://github.com/sumaiyalamia/INSE6220_40221319.git

***Abstract*—Large datasets are highly common and often challenging to interpret. And Principal Component Analysis (PCA) is a technique that can be used to reduce the dimensionality of such datasets, improve interpretability, and minimize information loss. In this report, PCA is applied on human stress detection through sleep dataset to understand the relationship between stress and sleep. Three different classification algorithms i.e.; Logistic Regression (LR), K-Nearest Neighbor (K-NN), and Naïve Bayes (NB) are applied on original dataset and transformed dataset (after applying PCA)to identify the stress levels by monitoring the relationships among different sleeping parameters such as snoring range, respiration rate, body temperature, limb movement rate, blood oxygen levels, eye movement, number of hours of sleep, and heart rate. Next each model is tuned with ideal hyperparameters to obtain better performance metrics and the performance of each algorithm is measured using F1 score, confusion matrix and Receiver Operating Characteristic (ROC) curves. Logistic Regression shows the superior performance over all other available machine learning models in PyCaret Library. However, other two algorithms perform as good as LR. Finally, an experiment is carried out for the interpretation of the model using the explainable AI (artificial intelligence) Shapley values. Extra trees (ET) classifier modelis used for this purpose. Overall, the algorithms successfully determine the eight classes of human stress detection dataset and F1-score 1 is achieved.**

***Keywords*—Machine Learning Algorithms, Linear Regression, Principal Component Analysis (PCA), Binary Classification, Logistic Regression, Extra Tree Classifier, Naïve Bayes, K-Nearest Neighbor.**

I. Introduction

Stress is the number one health concern around the world. It can be defined as a specific strain on the human body that is caused by different stressors. About 73% of people have stress due to lack of sleep [1]. Also, a high level of stress can lead to trouble sleeping. Scientific research has shown that long-term stress can result in chronic stress [2], which can then lead to serious health issues such as digestive, cancer, diabetes, or mental health problems. Stress can also have significant negative effects on work and oneself by causing emotional breakdowns. Therefore, the early detection of stress and an understanding of how to deal with it and its symptoms are of great interest. In this regard, machine learning (ML) and deep learning techniques can help in the rapid and accurate detection of human stress.

Stress detection is one of the important topics in machine learning. In recent years, machine learning algorithms have helped doctors detect human stress by developing stress detection systems. As a result, doctors can easily predict whether a person is stressed or not by monitoring sleep through stress detection devices. This can also help them prescribe the best treatment plan for their patients. In this report, at first, Principal Component Analysis (PCA) is applied to the human stress dataset with the aim of dimensionality reduction. After that, classification algorithms such as Logistic Regression (LR), K-nearest neighbors (K-NN), and Naïve Bayes (NB) are applied to the original dataset and PCA-transformed dataset. The purpose is to analyze and predict stress. Finally, Shapley values are used to make classification models explainable by AI (Artificial Intelligence). It should be noted that all of the presented results from the classification algorithms in this report represent the results obtained after applying PCA. More clearly, in this report, the results are used from the transformed dataset. The classification results of the original dataset can be found on the Google Colab notebook.

The later sections of the paper are organized as follows: Section II describes the PCA methodology. Section III elaborates on the various machine learning techniques that have been used to detect human stress through sleep. Section IV describes the dataset used for the research. Section V discusses the PCA results. An extensive analysis of the classification results has been shown in Section VI. Section VII provides a discussion on the explainable AI Shapley values. At last, Section VIII concludes the paper.

II. Principal Component Analysis

Principal component analysis (PCA) is a technique for reducing the dimensionality of large data sets by transforming a large set of variables into a smaller one that still contains most of the information of the original dataset. It has been applied and found useful in many disciplines.

PCA is helpful in emphasizing variation and bringing out strong patterns in a dataset based on the correlation between the features. Another very good use of PCA is to speed up the training process of the machine learning algorithms by reducing the number of features. And this helps with enabling computation without losing the data's resemblance to reality [3]. On the other hand, PCA is highly sensitive to data scaling. As a result, prior to PCA, it is important to standardize the features and assign equal importance to all of them.

A. PCA algorithm

The PCA algorithm is based on mathematical ideas namely:

- Variance and Covariance
- Eigen Vectors and Eigen Values.

Steps involved in Principal Component Analysis are explained below [4]:

1) **Standardize the Dataset:** The first step is to standardize the data before performing PCA. This will ensure an equal contribution of the data to the analysis. For that, the mean vector $\bar{x}$ of each column of the dataset needs to be computed first. The following expression can be used to compute the mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (1)$$

The next step is to compute the centered data matrix Y by subtracting off column means. The final centered data matrix

(Y) can be expressed as follows:

$$Y = HX \tag{2}$$

Where, H represents the centering matrix.

2) **Covariance Matrix Computation:** The main goal of this step is to see how the variables in the input data set are varying with the mean value calculated. In other words, this step helps to see if there are any relationsips between the variables. Sometimes variables are so closely related that they contain redundant information. In order to identify highly interrelated variables, covariance matrix is computed. The $p \times p$ covariance matrix is computed as follows:

$$S = \frac{1}{n-1} Y^T Y \tag{3}$$

3) **Eigen Decomposition:** In this step at first we need to compute the eigenvectors and eigenvalues of the covarience matrix to determine the principal components of the variables. Here, eigenvectors represent the direction of each principal component whereas eigenvalues represent the variance captured by each principal component. Eigen decomposition can be computed using the following equation:

$$S = A \Lambda A^T \tag{4}$$

Where A is the p × p square matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

4) **Principal Compoents:** This step aims to compute the transformed matrix Z which is size of $n \times p$. The rows of Z represent the observations while columns of Z represent the PCs. The number of PCs is the same as the original data matrix's dimension. The equation of Z can be given by:

$$Z = YA \tag{5}$$

III. Machine Learning-Based Classification Algorithms

A. Logistic Regression (LR)

Logistic Regression is an extension of the linear regression model, which is used for classification [5]. The logistic regression model uses the logistic function and labels the output of a linear equation between 0 and 1. For example, LR classifies the output as zero if the value is 0.49 or lower. On the other hand, the LR considered the output as 1 if the value is 0.5 or higher. And this decision is determined by the logistic function. The following equation is used to define the logistic function:

$$logistic(n) = \frac{1}{1 + \exp(-n)} \tag{6}$$

Here, n represents the input and the output of $logistic(n)$ ranges from 0 to 1. The main advantage of linear regression is that it is one of the simplest machine learning algorithms, which is easy to implement and provides great training efficiency in some cases [6]. Also, it proves to be very efficient in classifying binary problems, such as if the stress level is high, low, or medium. On the other hand, logistic regression cannot handle missing values, and as a result, extra work needs to be done on the data regarding processing missing values. Another disadvantage of LR is that it is prone to overfitting. However, this problem can be overcome by introducing larger training data and regularization.

B. K- nearest neighbor (K-NN)

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier that uses proximity to make classifications or predictions about the grouping of an individual data point [7]. KNN is a kind of lazy learning algorithm, where the key computations are done during classification rather than training. As a result, the training time can be quite fast for KNN algorithms. However, the testing time is really slow. 'k' in KNN represents the maximum distance between neighbors that should be considered. In this algorithm, neighbors who are close contribute more to the average than those who are farther away.

Euclidean distance is the most commonly used distance measure for K-NN. In order to classify a sample, first K-NN selects the number of neighbors, calculates the Euclidean distance of k number of neighbors, takes the K nearest neighbors as per the calculated Euclidean distance. For example, the weight of each neighbor is calculated by 1/d, where d denotes distance.

C. Naïve Bayes (NB)

A Naïve Bayes classifier is a probabilistic machine learning model for classification. It is based on the Bayes theorem. Its fundamental assumptions are that each predictor makes an independent and equal contribution to the outcome [8]. Its simplicity makes it is useful for the classification of large data sets. Equation 7 finds the probability of an event given the probability of another event that has already occurred.

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \tag{7}$$

Where A is the event, we are finding the probability of, and B is the event that has already occurred [9]. Naïve Bayes is a very popular classifier that works really well even with a small dataset. Many real-world problems, including spam filtering, are solved by this simple machine-learning algorithm. It is an extremely fast learner compared to more complex machine learning classifiers.

IV. Data Set Description

The human stress detection through sleep dataset that is used for this project is obtained from Kaggle [10]. The dataset has 630 rows of data with 9 columns. However, the first eight columns are sleeping features and consist of "snoring rate of the user - sr", "respiration rate - rr (measured in breaths per minute and normal respiration rate for an adult at rest is 12 to 20 breaths per minute)", "body temperature – t (measured in Fahrenheit)", "limb movement rate - lm (measured in seconds)", blood oxygen levels - bo (measured in percentages and 90-100% is the normal range)", "eye movement – rem", "number of hours of sleep – sr.1", "heart rate - hr (measured

in BPM). These sleeping features are used to detect and classify human stress levels. Here, the last column titled "sl" is the label for the class to identify the stress levels as 0-low/normal, 1 – medium low, 2- medium, 3-medium high, and 4 -high. Finally, there are no null or missing values in the dataset, so there is no need to do any data cleaning.

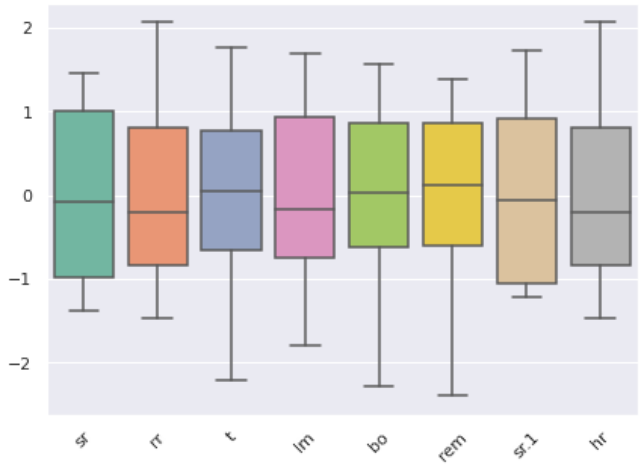

Fig. 1: Box plot

Utilizing the box and whisker plots and their eight-number summaries on the dataset, the distributions, central values, and variability of the features were measured. Fig. 1 illustrates the box plot of the features of the stress detection dataset. It can be observed from Fig. 1 that most of the features follow an approximately normal distribution. However, no outliers exist in any feature.

Fig. 2 shows the correlation matrix for the normalized features of the dataset. The features with large positive numbers are sr, rr, lm, rem, and hr. This evidently implies that these five features are highly correlated. Other three features, i.e., t, bo, and sr.1, show less correlation with the other features in the dataset. The pair plot in Fig. 3 supports this observation. The highly correlated features contain a higher number of values with a regularly increasing line. On the contrary, t, bo, and sr.1 display a less apparent correlation.

V. PCA Results

PCA is applied to the stress detection dataset. PCA can be implemented in two ways: (1) from scratch using standard Python libraries such as NumPy, or (2) by using a popular and well-documented PCA library. In the Google Colab notebook, implementation of both methods is provided. Even though the results obtained from both methods are almost similar, the usage of the PCA library brings more flexibility to the user, and a lot can be done by writing only a single line of code. In this report, the figures and plots are shown from the implementation using the PCA library.

The PCA steps can reduce the feature set of 8 to r numbers of features, where $r < 8$. The eigenvector matrix A is used to reduce the original $n \times p$ dataset. Each column of the eigenvector matrix A is represented by a PC. Each PC captures an amount of data that determines the dimension (r). The obtained eigenvector matrix (A) for the stress detection dataset is as follows (note: modified result of the matrix (A) has been shown here. The original result can be found on Google Colab Notebook):

$$A = \begin{bmatrix} -0.35 & -0.19 & -0.01 & -0.25 & 0.87 & -0.04 & 0.01 & 4.1 \times 10^{-16} \\ -0.35 & -0.29 & 0.35 & -0.12 & -0.25 & -0.28 & -0.02 & -0.07 \\ 0.34 & -0.50 & -0.18 & -0.24 & -003 & 0.06 & -0.72 & -3.8 \times 10^{-16} \\ -0.35 & -0.27 & 0.07 & 0.15 & -0.12 & 0.86 & 0.03 & -4.1 \times 10^{-16} \\ 0.34 & -0.49 & -0.16 & -0.35 & -0.08 & 0.02 & 0.68 & 5.5 \times 10^{-17} \\ -0.34 & -0.28 & -0.70 & 0.45 & -0.09 & 0.27 & 0.03 & 1.94 \times 10^{-16} \\ 0.34 & -0.36 & 0.42 & 0.69 & 0.26 & -0.09 & 0.06 & 3.05 \times 10^{-16} \\ -0.35 & -0.29 & 0.35 & -0.12 & -0.25 & -0.28 & -0.02 & 0.07 \end{bmatrix}$$

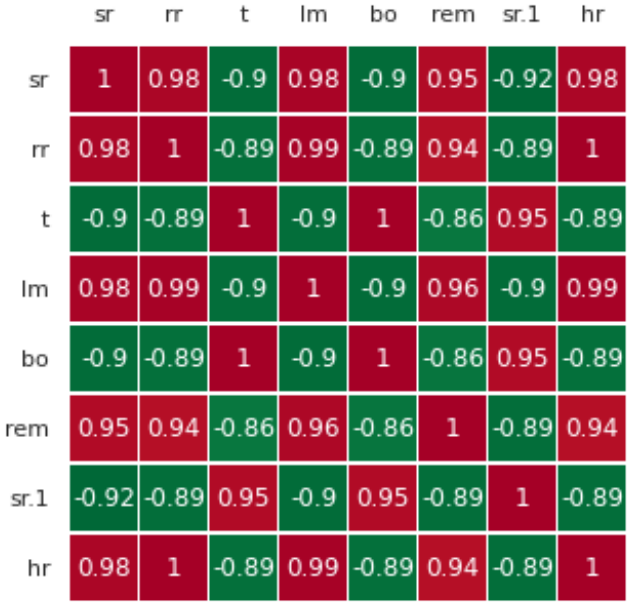

Fig. 2: Correlation matrix

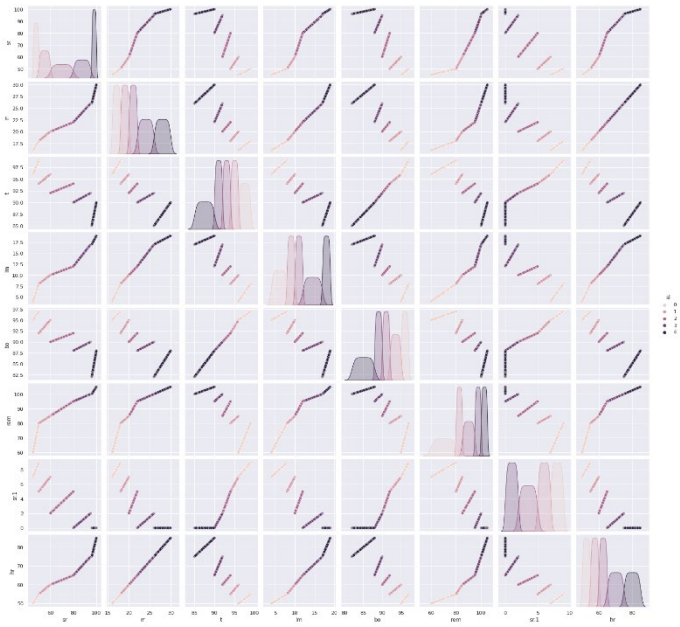

Fig. 3: Pair Plot

and the corresponding eigenvalues are:

$$\lambda = \begin{bmatrix} 7.50 \\ 0.326 \\ 0.0982 \\ 0.0504 \\ 0.0235 \\ 0.0003 \\ 0.0001 \\ 1.14 \times 10^{-31} \end{bmatrix}$$

Fig 4 and Fig. 5 demonstrate the scree plot and pareto plot of the PCs, respectively. The scree plot and pareto plot display the amount of variance explained by each principal component. The percentage of variance experienced by j -th PC can be evaluated using the following equation:

$$j = \frac{\lambda_j}{\sum_j^p \lambda_j} \times 100, j = 1,2, \dots p, \quad (8)$$

where λ_j denotes the eigenvalue and the variance of the j -th PC.

For the human stress detection PCA library generated two principal components using the PCA technique (with n=2). Fig. 4 and Fig. 5 plot the number of PCs vs. the explained variance. It can be observed from both figures that the variance of the first two PCs contributes to 97.77% of the amount of variance in the original dataset, i.e., the first PC holds 93.7% of the variance ($l1$ = 93.7%) and the second PC holds 4.07% of the variance ($l2$ = 4.07%). The scatter plot presents that the elbow is located on the second PC. These two observations imply that the dimension of the feature set can be reduced to two ($r = 2$).

Z1 denotes the first principal component. The first principal component Z1 is given by:

$$z_1 = -0.3590X_1 - 0.3573X_2 + 0.3481X_3 - 0.3597X_4 + 0.3483X_5 - 0.3490X_6 + 0.3490X_7 - 0.3573X_8 \quad (9)$$

It can be observed from the first PC that X_3 (t), X_5 (bo), X_7 (sr.1) contributes to the most in first PC. However, none of the features have a negligible contribution to the first PC.

The second principal component Z2 is given by:

$$z_2 = -0.193X_1 - 0.291X_2 - 0.507X_3 - 0.276X_4 - 0.495X_5 - 0.282X_6 - 0.363X_7 - 0.291X_8 \quad (10)$$

From the second PC z_2 it can be seen that X_1 (sr), X_2 (rr), X_8 (hr) have the highest contribution in the second PC. Furthermore, none of the features make a negligible contribution to the second PC.

Fig. 6 represents the PC coefficient plot. It depicts the amount of contribution each feature makes on the first two PCs. The figure supports the previous calculation of PCs, and it can be clearly seen from the figure that sr.1, bo, and t has the highest contributions in the first PC. On the other hand, sr has the greatest contribution to the second PC. And rr and hr make equal contribution on PC2. From the graph, it can also be noted that sr.1, bo, and t features have both positive and negative coefficients and are located on the bottom right side of the plot, far away from the clusters of features on the left side. On the other hand, sr, hr, rem, and lm are located on the negative side of the plot, but their location is far away from the other features.

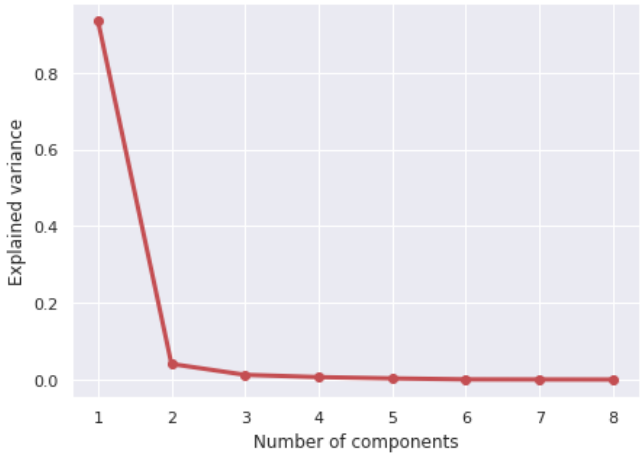

Fig. 4: Scree Plot

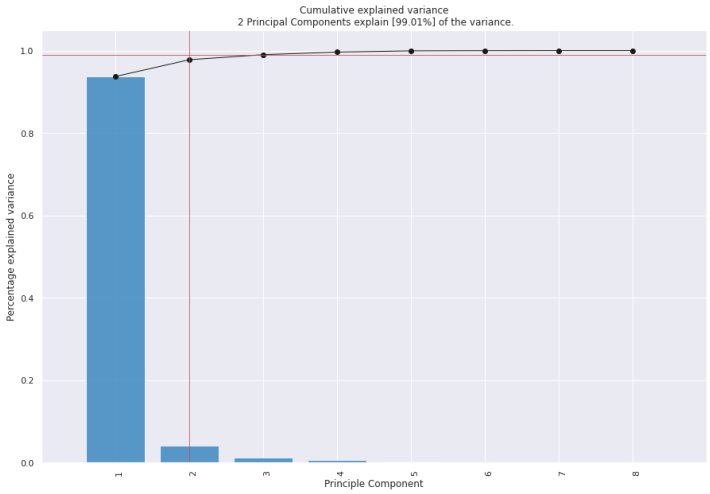

Fig. 5: Pareto Plot

The Biplot in Fig. 7 displays a different visual representation of the first two PCs. The axes of biplot represents the first two PCs. The rows of the eigenvector matrix is shown as a vector. Each of the observations in the dataset is drawn as a dot on the plot. The vectors for features namely, sr.1, bo and t show very small angle with the first PC and very large angle with the second PC. These evident supports that the analysis of the PC coefficient plot of Fig. 6. It implies that these three features have a large contribution to the first PC and very small contribution to the second PC. On the other hand, the vectors for (sr), (rr) and (hr) shows the opposite phenomenon. They create a larger angle with the first PC and smaller angle with the second PC. It means that they are more related to the second PC rather than the first one. Furthermore, the vectors which follow the same direction are positively correlated with each other. For instance, sr.1, bo and t are facing in the same direction.

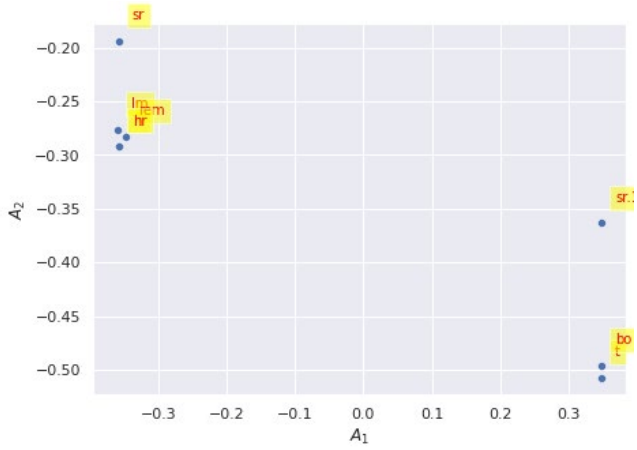

Fig. 6: PC coefficient plot

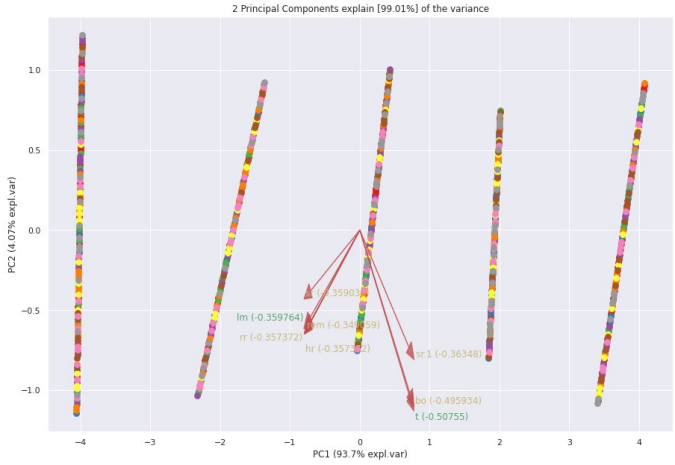

Fig. 7: Biplot

VI. Classification Results

In this section, the performance of three popular classification algorithms on the human stress dataset is discussed. In order to observe the effects of PCA on the human stress dataset, the classification algorithms are applied to both the original dataset and the PCA-applied dataset with two PCA components. The classification is performed using the PyCaret library in Python. The original dataset is split into a training and testing set with a proportion of 70% and 30%, respectively. For the sake of reproducibility, the session id is set to 123. Using PyCaret, it is possible to create a performance comparison table among all available classification algorithms on the target dataset and find the best model with the highest accuracy. It can be observed from Fig. 8 that, before applying PCA, the best three classification models with the highest accuracies on the stress detection dataset are Logistic Regression (LR), K-Nearest Neighbor (K-NN), and Naïve Bayes Classifier (NB).

On the other hand, Fig. 9 demonstrates the comparison among the classification models after applying PCA. Here, it can be seen that the best three models, which give the highest accuracy on the transformed dataset is LR, K-NN, and NB, which are the same as before applying PCA. Hence, these three algorithms are used for the evaluation. purposes during

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.537
knn	K Neighbors Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.022
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.015
lda	Linear Discriminant Analysis	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.016
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.165
rf	Random Forest Classifier	0.9975	1.0000	0.9975	0.9978	0.9975	0.9969	0.9970	0.211
lightgbm	Light Gradient Boosting Machine	0.9975	1.0000	0.9975	0.9978	0.9975	0.9969	0.9970	0.165
dt	Decision Tree Classifier	0.9899	0.9937	0.9903	0.9910	0.9899	0.9873	0.9876	0.014
gbc	Gradient Boosting Classifier	0.9898	1.0000	0.9899	0.9910	0.9898	0.9872	0.9876	0.443
ridge	Ridge Classifier	0.8964	0.0000	0.8974	0.9228	0.8975	0.8706	0.8777	0.012
svm	SVM - Linear Kernel	0.7501	0.0000	0.7449	0.6924	0.6931	0.6871	0.7180	0.018
ada	Ada Boost Classifier	0.6086	0.8605	0.6003	0.5004	0.5150	0.5113	0.6049	0.102
dummy	Dummy Classifier	0.2172	0.5000	0.2000	0.0473	0.0777	0.0000	0.0000	0.015
qda	Quadratic Discriminant Analysis	0.1996	0.0000	0.2000	0.0399	0.0665	0.0000	0.0000	0.019

Fig. 8: Comparison among classification models before applying PCA

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.060
knn	K Neighbors Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.031
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.017
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.017
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.238
qda	Quadratic Discriminant Analysis	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.017
lda	Linear Discriminant Analysis	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.021
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.198
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.097
gbc	Gradient Boosting Classifier	0.9975	1.0000	0.9975	0.9978	0.9975	0.9969	0.9970	0.419
svm	SVM - Linear Kernel	0.8888	0.0000	0.8836	0.8736	0.8663	0.8607	0.8768	0.025
ridge	Ridge Classifier	0.8254	0.0000	0.8211	0.8328	0.8204	0.7816	0.7857	0.017
ada	Ada Boost Classifier	0.6085	0.8592	0.6000	0.4799	0.5118	0.5113	0.6053	0.106
dummy	Dummy Classifier	0.2172	0.5000	0.2000	0.0473	0.0777	0.0000	0.0000	0.017

Fig. 9: Comparison among classification models after applying PCA

```
tuned_lr_pca = tune_model(lr_pca)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Mean	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SD	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 10: LR metrics score after hyperparameter tuning

the rest of the experiment. The original and the transformed dataset is used to train, tune, and evaluate these three algorithms. Both experiments (classification algorithms) applied to the original dataset and the transformed dataset. can be found in the Google Collaborate Notebook. However, in this report, we only focus on the results obtained after applying PCA (transformed dataset).

In order to improve the performance of a model, hyperparameter tuning plays an important role. Hyperparameter tuning with PyCaret involves three steps: creating a model, tuning it, and evaluating its performance. At first, a classification model per algorithm is produced. Then the tune_model() function is used for tuning the model with ideal hyperparameters. This function tunes the model using effective hyperparameters on a predefined search space and scores it using stratified K-fold cross validation. By default, PyCaret applies 10-fold stratified K-fold validation to the three algorithms. Moreover, the LR model is tuned with an L2 penalty, a regularization technique to prevent overfitting. For K-NN, the number of k-nearest members is tuned, and the var_smoothing is tuned for NB classifier. It can be observed from Fig. 10 that tuned LR model metrics are better than the base model metrics (before hyperparameter tuning).

Since the stress dataset is a binary classification problem, precision and recall can evaluate the performance of each class individually. Precision and recall are two measurements which together are used to evaluate the performance of classification. Precision is defined as the fraction of relevant instances among all retrieved instances, whereas recall, represents the fraction of retrieved instances among all relevant instances [10]. The obtained results from precision and recall is presented using the confusion matrices Fig. 11, Fig 12 and Fig 13. The confusion matrix is defined as the matrix providing the mix of predicted vs. actual class instances [11]. It illustrates correct and incorrect predictions with count values and breaks down for each class. Fig. 12, 13 and 14 shows the confusion matrix tables for the three algorithms which were applied on transformed dataset. The confusion matrices for the original dataset can be found in the Google Colab notebook. In the figures, the horizontal axis represents the class prediction and vertical axis represents the true/actual label. First of all, the comparison model on Fig. 9 shows that LR performs similar to all other compared models including K-NN and NB. This observation is also supported by the confusion matrices of Fig.11, 12 and 13.

It can be observed from Figs. 8 and 9 that we got a classification rate of 100%, which is good accuracy. Precision is the frequency with which a model's predictions are correct. All three algorithms predicted that a user is stressed 100% of the time in our prediction case. Also, if there are users who have stress in the test set and the three algorithms can identify it 100% of the time. All these observations are evidence of the benefits of applying PCA and hyperparameter tuning.

Another of the measurements of the performance evaluation is the F1-score. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean [12]. It is a great metric to compare the results among the classifiers. For example, classifier A has a higher recall, and classifier B has a higher precision. In this situation, the F1-score helps determine the better classifier. The function of the F1-score can be defined as follows:

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \quad (11)$$

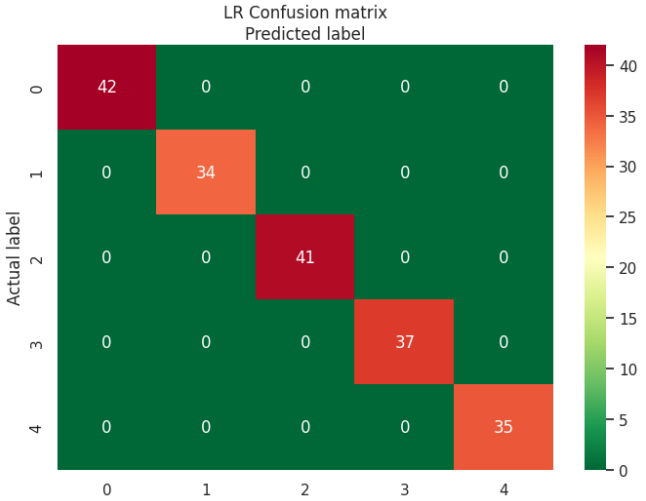

Fig. 11: Confusion matrix for LR on transformed dataset

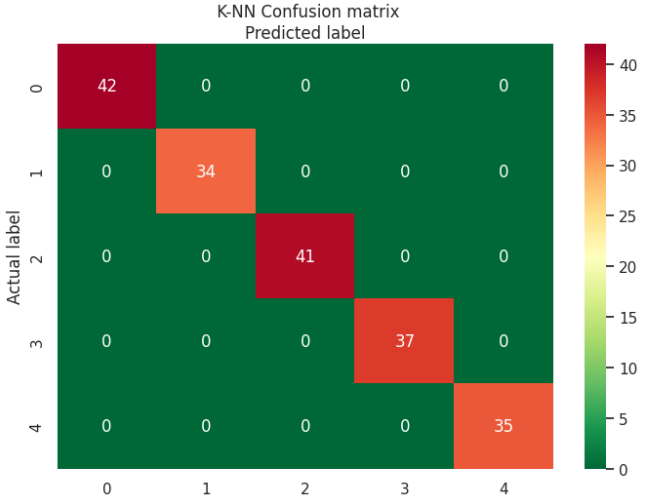

Fig. 12: Confusion matrix for K-NN on transformed dataset

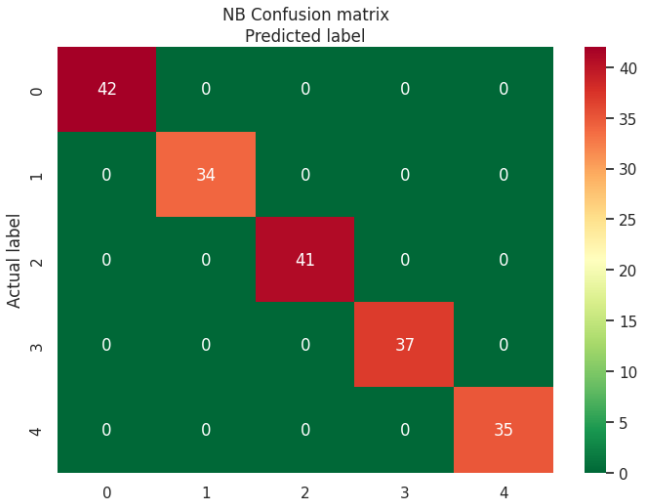

Fig. 13: Confusion matrix for NB on transformed dataset

From Fig 11, 12, and 13, we have the confusion matrices of the three classification algorithms applied to the

transformed dataset. Here, the diagonal elements are the correctly predicted samples. A total of 189 samples were correctly predicted by three of the algorithms.

As the final analysis step, the receiver operating characteristic (ROC) curve for the LR algorithm is shown in Fig. 14. A ROC curve is a graph that demonstrates the performance of a classification model at all classification thresholds. This curve plots two parameters: the true positive rate and the false positive rate. These parameters are the main components of the confusion matrix. Hence, the ROC curve and confusion matrix are closely related and can be considered as different visual representations of the same measurement. ROC curves for K-NN and NB can be found in the Google Colab notebook. The ROC curve of LR in Fig. 14 reflects the results of the confusion matrix. It plots the false-positive rate (x-axis) versus the true-positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. It also shows the graph of the macro- and micro-average curves. The ROC curve and AUC values show that LR is the best at predicting both classes and can predict 100% accurately. This result mirrors the results obtained from the confusion matrix.

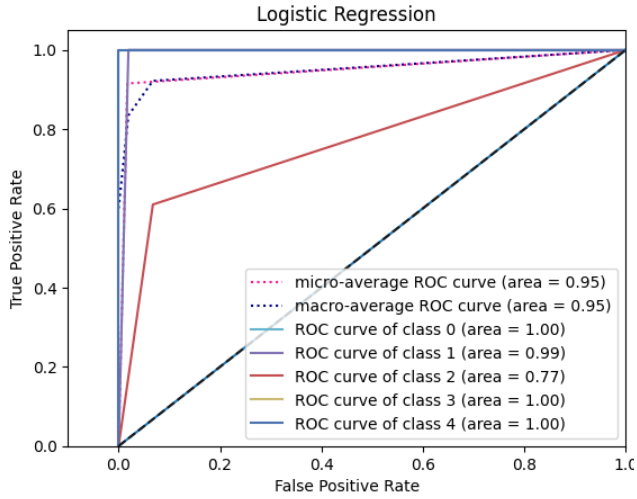

Fig. 14: ROC curve LR

Therefore, it is observed that the three algorithms are capable of successfully classifying the stress level classes.

VII. Explainable AI With Shapely Values

Model interpretability is an important metric in the context of ML. There are several ways of enhancing the explainability of a model, and feature importance is one of them. Feature importance helps estimate the contribution of each feature in the prediction process. Hence, in order to get an overview of the most important features on the PCs, we use the SHAP values by importing the open-source "shap" library of Python. SHAP (Shapley Additive Explanations) is a method that was first introduced by Lundberg and Lee [12] in 2016. SHAP explains individual predictions based on the optimal Shapley values using the concept of game theory. Specifically, SHAP can explain the prediction of an instance x by computing the contribution of each feature to the prediction [13]. Borrowing the concept of game theory, each feature of a dataset acts as a player in a coalition. A player can be an individual feature value, e.g., for tabular data, or a group of feature values. Shapley values describe how to adequately distribute the prediction among the feature set.

The shap library of Python is still in its development stage, and it only supports tree-based models (i.e., decision tree, random forest, and extra trees classifier) for binary classification problems. Since the human stress detection dataset is a binary classification problem, shape analysis cannot be performed on LR, KNN, or NB. Therefore, for the shape analysis, I have chosen a fourth model of the transformed dataset, which is "Extra Tree Classifier (ET)". Similar to other models, at first, an DT model is created and tuned with ideal hyperparameters. The tuned model is then passed to the shape library, which generates the interpretation plots. In our case, each PC acts as a player in the coalition.

Fig. 15 displays the summary plot of SHAP values. The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a PC and an instance. The y-axis represents the PCs and Shapley values are positioned on the x-axis. More specifically, component_1 represents the first PC, component_2 represents the second PC and component_3 represents the third PC. We can observe that there are no overlapping points in the direction of y-axis which indicates the distribution of the Shapley values per PC. All the PCs are ordered according to their importance. This observation supports the pareto plot and scree plot which indicates that the first PC holds the most feature variance. The red color indicates high PC value and blue color indicates low PC value. To interpret the summary plot, we can say that a low level of PC value has a high and positive impact on the human stress detection. Similarly, a high level of PC value has a low and negative impact on the human stress detection.

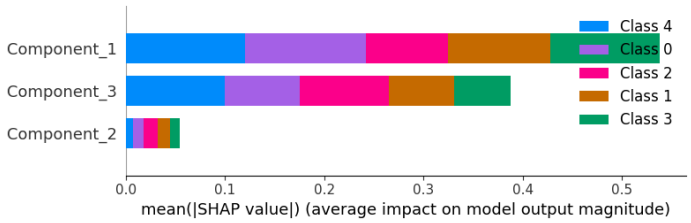

Fig. 15: Summary plot

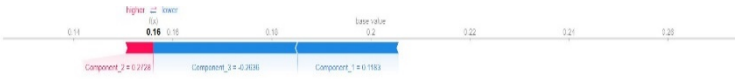

Fig. 16: Force plot for a single observation

Fig.16 represents the force plot for a single observation. It should be noted that this plot can only be used for one observation. For this particular example, the 32nd observation is chosen. This plot depicts the features that each contribute to pushing the model output away from the starting point. The base value is the value that would be predicted if no features were known for the current output [12]. In other words, it is the mean prediction of the test set. Here, the base value is 0.2. In the plot, the bold 0.16 is the model's score for this observation. Higher scores lead the model to predict 1, and lower scores lead the model to predict 0. The red color indicates that the second PC pushes the model for a higher prediction, and the blue color on the first and third PCs indicates that they are pushing the prediction to be lower.

However, this plot is only an output for this observation. It does not describe the predicted output of the entire model.

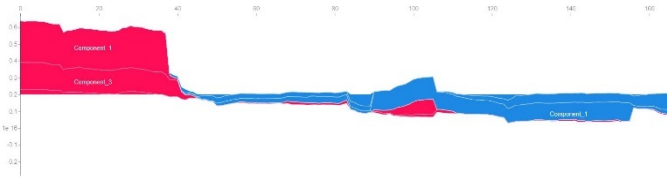

Fig. 17: Combined force plot

Fig. 17 displays the combined force plot of all PCs. This is a horizontal stack of all individual force plots that have been rotated by 90 degrees. In this plot, the y-axis is the x-axis of the individual force plot. There are 170 data points in the transformed test set; hence, the x-axis has 170 observations. This combined force plot shows the influence of each PC on the current prediction. Values in the blue color are considered to have a positive influence on the prediction, whereas values in the red color have a negative influence on the prediction.

VIII. Conclusion

In conclusion, PCA and three popular classification algorithms are applied to the stress detection dataset. The stress detection dataset holds information on attributes of sleep and stress level to identify them as low, medium low, medium, medium high, and high. At first, PCA is applied to the original dataset. The first two PCs account for 97% of the data variance. Hence, the feature set is reduced from 8 to 2. Extensive experiments are conducted on the first two PCs, and different plots are generated to validate the obtained results from different perspectives. To move forward, three classification algorithms, LR, K-NN, and NB, are applied to the original dataset as well as the transformed dataset with the first three components. Each algorithm is tuned with the ideal hyperparameter settings, and performance evaluation is conducted by comparing confusion matrices, ROC curves, and F1-scores. It is observed that after hyperparameter tuning, the performance metrics score of each algorithm is 100%. The LR, K-NN, and NB algorithms also performed well on the original dataset. Interestingly, after applying PCA LR, K-NN, and NB still performed the best and showed the best performance metrics. Finally, several interpretation plots are generated using explainable AI shapely values to improve the model's interpretability. To summarize, all three algorithms can successfully determine the stress level for human stress detection.

References

[1] E. Patterson, Stress Facts and Statistics [Online]. Available: https://www.therecoveryvillage.com/mental-health/stress/stress-statistics

[2] T. Chandola, E. Brunner and M. Marmot, Chronic stress at work and the metabolic syndrome: prospective study, pp.521-525, 2006. doi: https://doi.org/10.1136%2Fbmj.38693.435301.80

[3] StackExchange [Online]. Available: https://stats.stackexchange.com/questions/268704/why-do-we-use-pca-to-speed-up-learning-algorithms-when-we-could-just-reduce-the

[4] Z. Jaadi, A step by step explanation of principal component analysis (PCA) [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[5] Logistic Regression [Online]. Available: https://christophm.github.io/interpretable-ml-book/logistic.html

[6] Advantages and disadvantages of logistic regresion [Online]. Available: https://iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/

[7] IBM, K-Nearest Neighbors Algorithm [Online]. Available: https://www.ibm.com/topics/knn

[8] S. Sayad, Naïve Bayesian, [Online]. Aailable: https://www.saedsayad.com/naive_bayesian.htm

[9] Geeksforgeeks, Naïve Bayes Classifiers [Online]. Aailable: https://www.geeksforgeeks.org/naive-bayes-classifiers/

[10] C. Goutte and E. Gaussier, “A probabilistic interpretation of precision, recall and f-score, with implication for evaluation,” in European conference on information retrieval. Springer, 2005, pp. 345–359.

[11] I. Markoulidakis, I. Rallis, I. Georgoulas, G. Kopsiaftis, A. Doulamis, and N. Doulamis, “Multiclass confusion matrix reduction method and its application on net promoter score classification problem,” Technologies, vol. 9, no. 4, p. 81, 2021.

[12] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” Advances in neural information processing systems, vol. 30, 2017.

[13] C. Molnar, Interpretable Machine Learning, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book.