CSE422 : Artificial Intelligence

Project : Stress Level Detection

Section: 6

Group:14

Name	ID
Tasnuva Tanzim	21201802
Anika Islam	21101298

Table of Contents

1. Introduction

With every passing day and year, the stress level of a student is varying due to various factors. Some of the factors include : peer pressure, anxiety level, study load, future career concern. Owing to these factors and the student's health, stress level detection is a dire need. If this detection can be correctly accomplished using a machine learning model, then these students can be taken care of. For these reasons, our project has focused on the main factors causing stress to a student to find out which level of stress they are currently in. Using a dataset with these factors collected from students, we have trained some ML models, which we have used to predict the stress level and compared their accuracy score to come to a conclusion which model is most likely to detect stress level correctly.

2. Dataset Description

Source: Kaggle
Link:
https://www.kaggle.com/datasets/rxnach/student-stress-factors-a-comprehensive-analysis/data

Reference:
Student Stress Factors: A Comprehensive analysis. (2023, October 14). Kaggle. https://www.kaggle.com/datasets/rxnach/student-stress-factors-a-comprehensive-analysis/data

Data description:

Our collected dataset has a total of 21 columns, which means a total of 20 features and 1 target. All the features involved are quantitative. A total of 1100 data points have been included into this dataset from students.

Stress_level is the target and all the features involved are listed under the factors they belong to :

Psychological factors : Anxiety_level, Self_esteem, mental_health_history, Depression

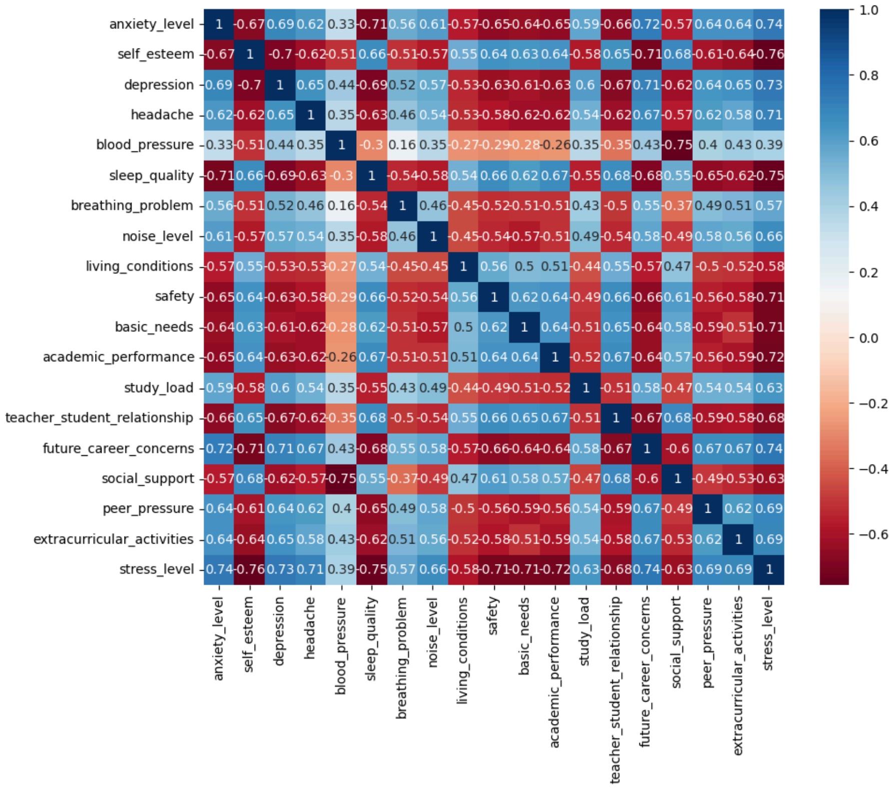
Physiological factors : Headache, Blood_pressure, Sleep_quality, Breathing_problem

Environmental factors : Noise_level,Living_conditions,Safety,Basic_needs

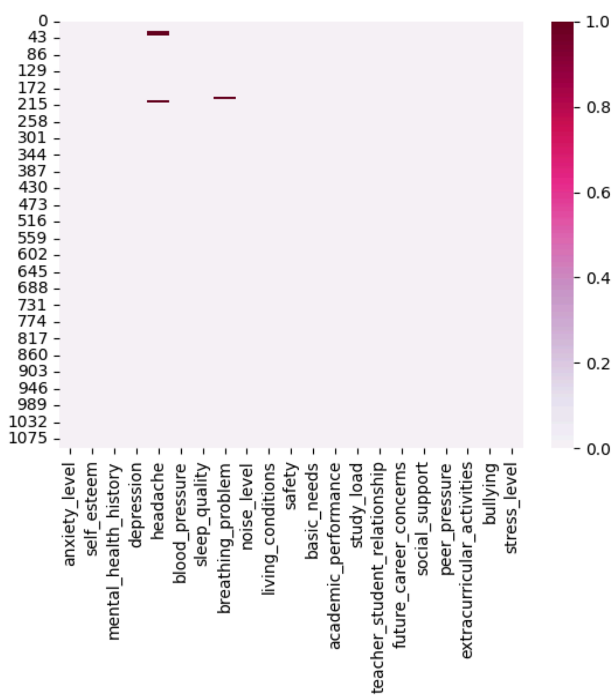
Academic factors : Academic_performance, Study_load, Teacher_student_relationship, Future_career_concerns

Social factors : Social_support, Peer_pressure, Extracurricular_activities, Bullying

To determine whether to implement regression or classification on this problem, we have viewed the target properly and have come up to the conclusion that the outputs such as 0,1 and 2 are the stress level indicating low, moderate and high stress levels respectively. These put forth that three different classes are involved in the target. Thus, the target indicates three different stress levels have been dedicated with numeric numbers to make the process of educating students with different stress levels easier. To deal with qualitative data variables, classification models have been taken into consideration.

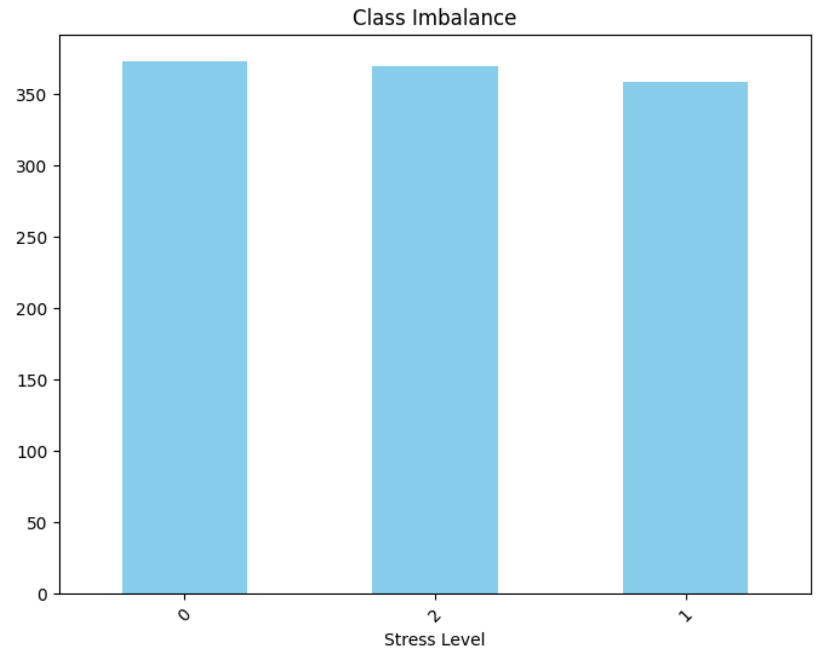


Heatmap displaying the correlation between all features



Null value representation

According to the plot above, breathing problems and headache have some null values which need to be handled.



Class imbalance chart for all stress level in the target

According to the chart, all the stress levels are evenly distributed in the dataset.

3. Dataset Pre-Processing

Null values :
There were no null values in the original dataset. However, we have included in some null values in order to show the pre-processing of null values. Preprocessing of null values is required because null values would have an impact on the learning rate if not removed.

To resolve this problem, we have used the imputing missing values approach to replace the null values with the mean of the non-null values of the columns using SimpleImputer. The columns containing the null values are headache and breathing_problem.

<table>
<tr><th>Before imputing missing values</th><th>After imputing missing values</th></tr>
<tr><td><pre>
anxiety_level   0
self_esteem   0
mental_health_history   0
depression   0
headache   18
blood_pressure   0
sleep_quality   0
breathing_problem   6
noise_level   0
living_conditions   0
safety   0
basic_needs   0
academic_performance   0
study_load   0
teacher_student_relationship   0
future_career_concerns   0
social_support   0
peer_pressure   0
extracurricular_activities   0
bullying   0
stress_level   0
dtype: int64
Index(['headache', 'breathing_problem'], dtype='object')
</pre></td><td><pre>
anxiety_level   0
self_esteem   0
mental_health_history   0
depression   0
headache   0
blood_pressure   0
sleep_quality   0
breathing_problem   0
noise_level   0
living_conditions   0
safety   0
basic_needs   0
academic_performance   0
study_load   0
teacher_student_relationship   0
future_career_concerns   0
social_support   0
peer_pressure   0
extracurricular_activities   0
bullying   0
stress_level   0
dtype: int64
Index([], dtype='object')
</pre></td></tr>
</table>

Categorical Values :

However, there were no categorical values in the original dataset. So, we have included non-numeric values in the mental_health_history column to introduce categorical variables. Preprocessing of categorical variables is needed as ML models can understand only numeric values, so with the existence of non-numeric values, errors can rise while splitting the dataset and training the models.

To overcome this problem, Label Encoder has been used as there are only two possible outcomes in the mental_health_history, where ‘Good’ and ‘Bad’ are encoded with 1 and 0 respectively.

Before Label encoding :

	anxiety_level	self_esteem	mental_health_history	depression	headache	blood_pressure	sleep_quality	breathing_problem	noise_level	living_conditions	...	basic_needs	academic_performance	study_load	teacher_student_relationship	future_ca
0	14	20	Bad	11	2.0	1	2	4.0	2	3	...	2	3	2	3	
1	15	8	Good	15	5.0	3	1	4.0	3	1	...	2	1	4	1	
2	12	18	Bad	14	2.0	1	2	2.0	2	2	...	2	2	3	3	
3	16	12	Good	15	4.0	3	1	3.0	4	2	...	2	2	4	1	
4	16	28	Bad	7	2.0	3	5	1.0	3	2	...	3	4	3	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1095	11	17	Bad	14	3.0	1	3	2.0	2	2	...	3	2	2	2	
1096	9	12	Good	8	0.0	3	0	0.0	0	1	...	4	0	1	1	
1097	4	26	Bad	3	1.0	2	5	2.0	2	3	...	4	5	1	4	
1098	21	0	Bad	19	5.0	3	1	4.0	3	1	...	1	2	5	1	
1099	18	6	Bad	15	3.0	3	0	3.0	3	0	...	3	3	4	3	

1100 rows × 21 columns

After Label encoding:

	anxiety_level	self_esteem	mental_health_history	depression	headache	blood_pressure	sleep_quality	breathing_problem	noise_level	living_conditions	...	basic_needs	academic_performance	study_load	teacher_student_relationship	future_ca
0	14	20	0	11	2.0	1	2	4.0	2	3	...	2	3	2	3	
1	15	8	1	15	5.0	3	1	4.0	3	1	...	2	1	4	1	
2	12	18	0	14	2.0	1	2	2.0	2	2	...	2	2	3	3	
3	16	12	1	15	4.0	3	1	3.0	4	2	...	2	2	4	1	
4	16	28	0	7	2.0	3	5	1.0	3	2	...	3	4	3	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1095	11	17	0	14	3.0	1	3	2.0	2	2	...	3	2	2	2	
1096	9	12	1	8	0.0	3	0	0.0	0	1	...	4	0	1	1	
1097	4	26	0	3	1.0	2	5	2.0	2	3	...	4	5	1	4	
1098	21	0	0	19	5.0	3	1	4.0	3	1	...	1	2	5	1	
1099	18	6	0	15	3.0	3	0	3.0	3	0	...	3	3	4	3	

1100 rows × 21 columns

Similar impact on the dataset:

Correlation of the dataset is checked and the columns with 0.75 correlation have similar impact on the dataset. So, removing a feature with the same correlation would not have a drastic impact on the prediction. As a result, bullying has the most correlation with 0.75 as shown on the heatmap shown above, so this feature has been dropped out.

4. Feature Scaling

In our dataset, columns have large and small valued numeric values. The largely valued numbers may show more importance than the small valued numbers in cases where the latter has to take the dominance. To overcome this, numbers need to be equally distributed using feature scaling.

In our project, we have applied MinMaxScaler and StandardScaler on our training and testing dataset. KNN is trained with these scaled datasets and the accuracy with StandardScaler (88%) is found to be greater than that of MinMaxScaler (87%). So, we have used StandardScaler for feature scaling purposes.

Before scaling :

```
per-feature minimum before scaling:
 anxiety_level                  0.0
self_esteem                    0.0
depression                     0.0
headache                       0.0
blood_pressure                 1.0
sleep_quality                  0.0
breathing_problem              0.0
noise_level                    0.0
living_conditions              0.0
safety                         0.0
basic_needs                    0.0
academic_performance           0.0
study_load                     0.0
teacher_student_relationship   0.0
future_career_concerns         0.0
social_support                 0.0
peer_pressure                  0.0
extracurricular_activities     0.0
dtype: float64
per-feature maximum before scaling:
 anxiety_level                  21.0
self_esteem                    30.0
depression                     27.0
headache                        5.0
blood_pressure                  3.0
sleep_quality                   5.0
breathing_problem               5.0
noise_level                     5.0
living_conditions               5.0
safety                          5.0
basic_needs                     5.0
academic_performance            5.0
study_load                      5.0
teacher_student_relationship    5.0
future_career_concerns          5.0
social_support                  3.0
peer_pressure                   5.0
extracurricular_activities      5.0
dtype: float64
```

After scaling:

```
per-feature minimum after scaling:
 [-1.83058064 -2.01391324 -1.64778344 -1.80761938 -1.41865959 -1.6982084
 -1.91485667 -2.02929326 -2.23967631 -1.95142334 -1.93113626 -1.94598872
 -2.00863607 -1.9124246  -1.74286069 -1.79861142 -1.90237237 -1.93729249]
per-feature maximum after scaling:
 [1.6193598  1.37587078 1.85466188 1.78427984 0.97164213 1.51619786
 1.60996723 1.77624939 2.25602431 1.63643715 1.56269579 1.57140873
 1.80836732 1.70745481 1.54475011 1.09671428 1.59196619 1.57097472]
```

5. Dataset splitting

After preprocessing, the dataset has been split into training and testing datasets with a size of 70% and 30% each using train_test_split function of the sklearn library. We have stratified our dataset using the target column to get rid of the effect of any imbalanced dataset present. Target outcomes are uniformly distributed using stratify.

```
Original dataset: (1100, 19)
Data for Training:  (770, 18)
Data for Testing:  (330, 18)
```

6. Model Training & testing

We have trained 5 ML models using the training dataset and processed with the inference using the testing dataset to obtain the accuracy score for training and testing. The models used are given below :

Logistic Regression

Choosing Logistic Regression for predicting student stress levels was based on its capability to provide probabilistic outputs and ease of interpretation, which are crucial for making informed decisions about interventions. Compared to models like Decision Trees and SVM, Logistic Regression offers a straightforward understanding of how each feature influences the prediction, through its coefficients. Unlike KNN, which relies heavily on the local structure of the data, Logistic Regression assumes a linear relationship between the features and the logarithm of the odds of the outcomes, making it more scalable and generally faster for predictions on large datasets. This model of ours gave training accuracy of 90.0% and testing accuracy of 89.39%.

Naive Bayes

GaussianNB calculates the probability of a data point belonging to each class based on the feature values using Bayes' theorem. It assumes that features are independent and follows a Gaussian distribution, hence "naive", and then selects the class with the highest probability. The classifier shows a training accuracy of approximately 87.9% and a test accuracy of about 90.91%.

Support Vector Machine (SVM)

Choosing a Support Vector Machine (SVM) for your stress level detection project was driven by SVM's strength in handling high-dimensional data and its ability to find the optimal hyperplane for class separation. Unlike Logistic Regression, which assumes a linear relationship, SVM can efficiently manage both linear and non-linear relationships through the use of kernel functions, making it versatile for complex datasets. Compared to KNN, SVM is less sensitive to noise and does not rely on the entire data set for making predictions, leading to potentially better performance in environments where features strongly influence class distinctions. This makes SVM particularly effective for your nuanced and feature-rich dataset. Our Support Vector Machine gave us training accuracy of 91.17% and testing accuracy of 89.7%.

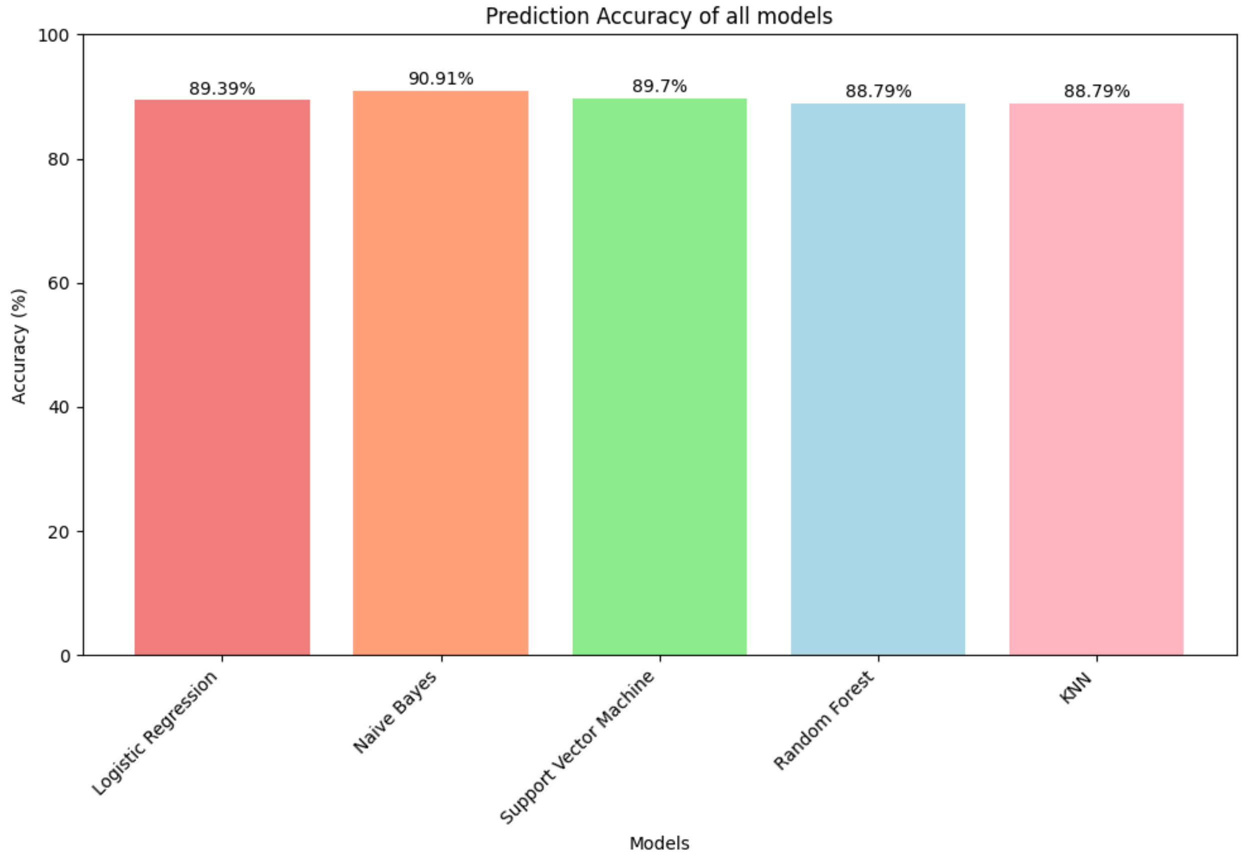
Random Forest

The random forest uses 50 decision trees, each predicting 50 different predictions. Each decision tree selects the best features to split the data based on certain criteria (like information gain for classification tasks) to create branches in the tree. Unlike Logistic Regression, which is limited to linear boundaries, Decision Trees can handle complex datasets with a mixture of categorical and numerical features effectively. Compared to SVM and KNN, Decision Trees are faster to train and provide a clear visualization of how decisions are made, making them ideal for stakeholder presentations. However, they can be prone to overfitting, especially with very detailed trees.This model here, gives training accuracy of 100% and testing accuracy of 87.88%.

KNN Classifier

KNN classifies data points based on the majority class among their k nearest neighbors. It measures distance between points in a multi-dimensional space, assigning the class most common among its k nearest neighbors.We have set the value of k as 5, and got training accuracy of 90.26% and testing accuracy of 88.79%.

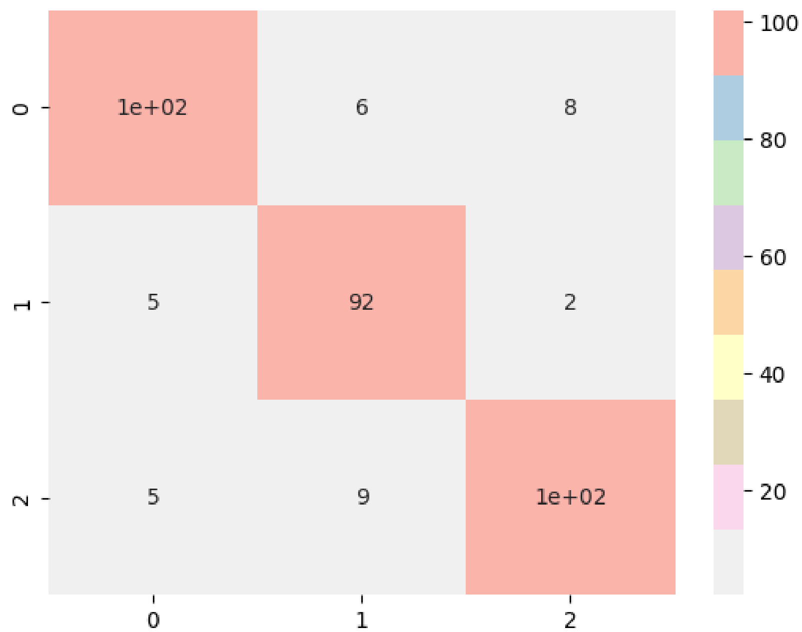
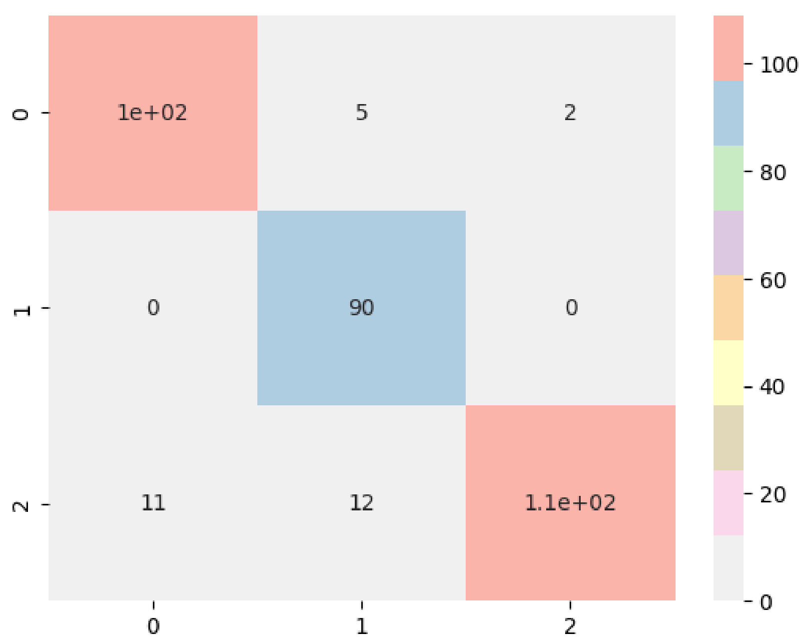
7. Model selection/ comparison Analysis

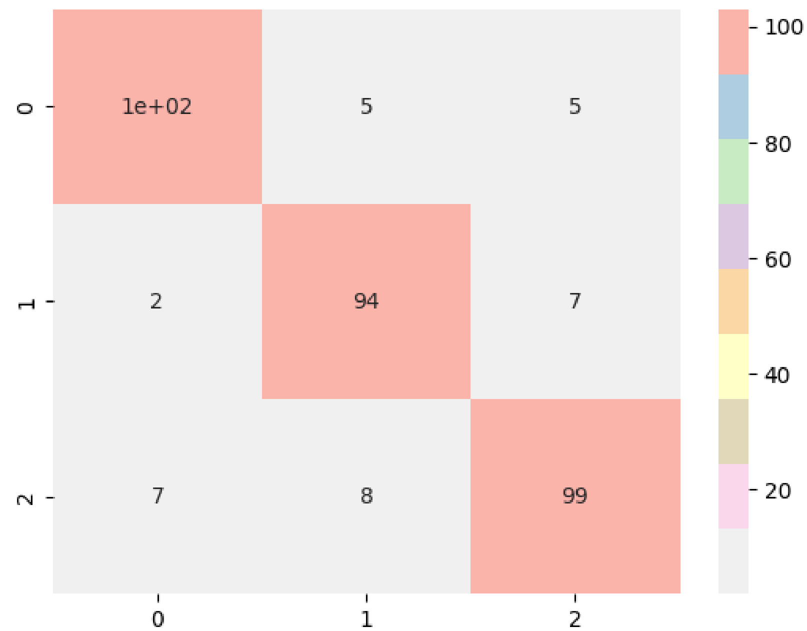
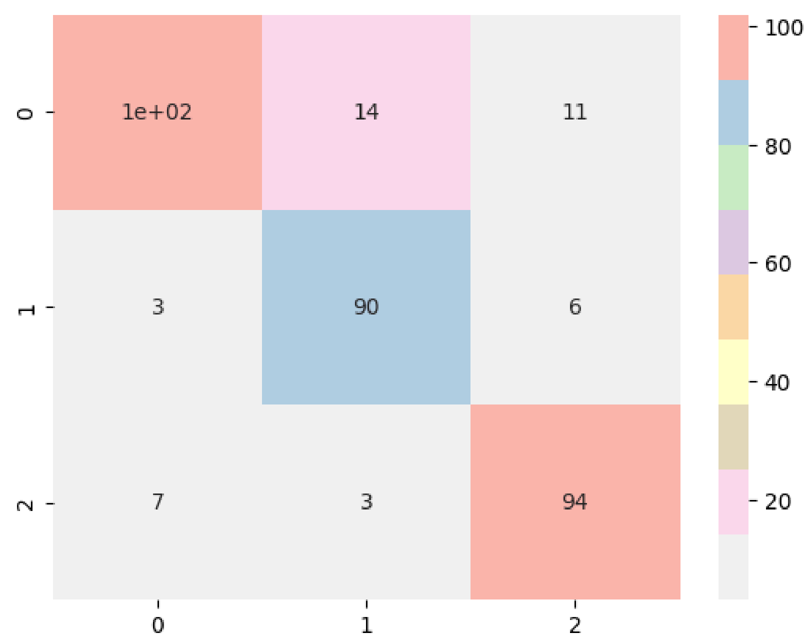


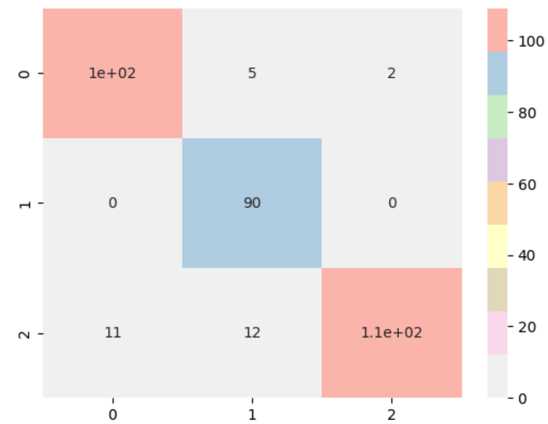
Prediction Accuracy of all models

According to the prediction accuracy chart above, Naive Bayes has provided relatively better accuracy. This model can be used for stress level detection.

Confusion Matrix Table

<table>
<tr><th>Model</th><th>Confusion Matrix</th><th>Visualization of Confusion Matrix</th></tr>
<tr><td>Logistic Regression</td><td>Confusion_matrix:
[[102 6 8]
[5 92 2]
[5 9 101]]</td><td>
</td></tr>
<tr><td>Naive Bayes</td><td>Confusion_matrix:
[[101 5 2]
[0 90 0]
[11 12 109]]</td><td>
</td></tr>
</table>

SVM	Confusion_matrix: [[103 5 5] [2 94 7] [7 8 99]]	
Random Forest	Confusion_matrix: [[102 14 11] [3 90 6] [7 3 94]]	

<table>
<tr>
<td>KNN</td>
<td>Confusion_matrix:
[[103 5 6]
[6 93 8]
[3 9 97]]</td>
<td>
</td>
</tr>
</table>

Classification Report Table

<table>
<tr><th>Model</th><th>Classification Report</th></tr>
<tr><td>Logistic Regression</td><td><pre>
Classification Report
              precision   recall f1-score   support

           0   0.91   0.88   0.89   116
           1   0.86   0.93   0.89   99
           2   0.91   0.88   0.89   115

    accuracy   0.89   330
   macro avg   0.89   0.90   0.89   330
weighted avg   0.90   0.89   0.89   330
</pre></td></tr>
<tr><td>Naive Bayes</td><td><pre>
Classification Report
              precision   recall f1-score   support

           0   0.90   0.94   0.92   108
           1   0.84   1.00   0.91   90
           2   0.98   0.83   0.90   132

    accuracy   0.91   330
   macro avg   0.91   0.92   0.91   330
weighted avg   0.92   0.91   0.91   330
</pre></td></tr>
<tr><td>SVM</td><td><pre>
Classification Report
              precision   recall f1-score   support

           0   0.92   0.91   0.92   113
           1   0.88   0.91   0.90   103
           2   0.89   0.87   0.88   114

    accuracy   0.90   330
   macro avg   0.90   0.90   0.90   330
weighted avg   0.90   0.90   0.90   330
</pre></td></tr>
</table>

Random Forest

```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.80      0.85       127
           1       0.84      0.91      0.87        99
           2       0.85      0.90      0.87       104

    accuracy                           0.87       330
   macro avg       0.87      0.87      0.87       330
weighted avg       0.87      0.87      0.87       330
```

KNN

```
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.90      0.91       114
           1       0.87      0.87      0.87       107
           2       0.87      0.89      0.88       109

    accuracy                           0.89       330
   macro avg       0.89      0.89      0.89       330
weighted avg       0.89      0.89      0.89       330
```

8. Conclusion

In our project, we tested five different classification models to predict the Stress level. All the models we used in the project work in a different mechanism, but the purpose of all of them here is to get which model efficiently uses. Every model could do that in their own way. We have used classification report to obeserve precision, recall, f1_score for all three classes using all the models. The dataset presents a comprehensive collection of human attributes and stress level metrics, suitable for exploratory data analysis and predictive modeling. However, challenges such as null values and categorical data require preprocessing techniques like imputation and encoding for effective analysis. With proper handling, this dataset offers valuable insights into factors influencing stress efficiency and can support the development of robust predictive models.