Understanding the Dataset

The given dataset contains 3 independent variables HR, respr and Time respectively, and 1 dependent variable “Label”. The independent variable determines the value of the dependent variable which is also known as “Target/Label”. It is 0 for no stress and 1 for stress. Thus, independent variables determines whether a participant is stressed or not. The dataset contains the target label thus, it is the case of supervised machine learning. And the target label has values 0 and 1 which is discrete, hence, it is a case of classification task.

Data Exploration

Imbalanced dataset is the one which contains uneven distribution of target class variable which can be determined by finding imbalanced ratio which is 2.05 in the given dataset. That is non stressed data is more than stressed data, thus it is imbalanced dataset.

Choosing an ML Package

There are many open-source machine learning packages available. Such as scikit-learn, TensorFlow, PyTorch, XGBoost, LightGBM, Keras, Pandas etc. For this task scikit-learn package was chosen as it provides wide range of algorithm with consistent API, making it easy to switch and test different algorithm.

Short overview of its main features:

1. Scikit-learn provides a consistent and easy-to-use API, which is uniform among algorithms, making switching between them easier.
2. Scikit-learn offers a large number of supervised and unsupervised machine learning methods, including linear and logistic regression, support vector machines, decision trees, random forests, k-nearest neighbours etc.
3. Scaling, normalisation, one-hot encoding, and feature selection are among the methods provided by the library for data preprocessing and feature engineering.
4. For model evaluation and model validation there are functions available in scikit-learn.

Data Pre-processing

In the given dataset, one of the independent variable i.e. HR, has 44 missing values, which has been handled by dropping those data. Next, data standardisation is performed for preparing the data. It has been used to transform the data such that each feature has zero mean and unit variance.

Algorithm Selection and Application

The two classification algorithm chosen for this task are XGBoost Algorithm and Random Forest Algorithm.

Random Forest Algorithm:

The random forest algorithm basically works in two parts. One it creates decision trees out of each record in dataset. Two the final output is generated by the majority of voting. Entropy is a metric used in information theory that evaluates the impurity or uncertainty in a set of data. It determines how a decision tree splits data.

For given dataset, entropy can be calculated as follows:

$$E = -\left[p(0) * log2(p(0)) + p(1) * log2(p(1))\right]$$

Where, p(0) = proportion of instances in class “0” of dataset
p(1) = proportion of instances in class “1” of dataset

Information gain aids in determining the order of attributes in decision tree nodes.

$$IG(A) = E(D) - \Sigma[(|Dv|/|D|) * E(Dv)]$$

Where, IG(A) is the Information Gain for attribute A
E(D) is the entropy of the original dataset D
Dv represents the subset of the dataset for which attribute A has the value v
|Dv| is the number of instances in the subset Dv
|D| is the total number of instances in the dataset D

1. On the pre-processed dataset, apply random forest algorithm.
2. In this algorithm, randomly some number of data points are chosen from the training set.
3. Many hyperparameters are available to improve the prediction or the speed of the model to perform faster.
4. Decision tree is created for each data point and thus, each tree will generate its own result.
5. Now for classification task, the final output is based on majority voting from each decision tree.

XGBoost Algorithm:

XGBoost stands for extreme gradient boosting. This algorithm basically works by learning from the other models or the decision trees.
The boosting equation for Xgboost algorithm can be calculated as follows:

$$Ft(x) = Ft - 1(x) + \nu * h(x, \theta t)$$

Where, Ft(x) = the final prediction after t boosting iterations
Ft-1(x) = the prediction from the previous iteration (t-1)
v = the learning rate, which controls the step size in each iteration. It's a hyperparameter that ranges between 0 and 1
h(x, θt) = the weak learner (typically a decision tree) for the t-th iteration with parameters θt

1. When Xgboost is applied on the dataset, it first generates the decision trees in a sequence.
2. The first decision tree picked is the weakest learning tree among all.
3. After the initial tree makes its prediction, the next tree is generated based on the results of previous tree.
4. The new tree generated always learns from its previous tree thereby correcting the errors made by it. Hence, we always get stronger and better new decision tree.
5. While training the model, weights are assigned to each datapoints, higher the weight weaker the data point. This allows the algorithm to focus, learn and correct the prediction.
6. To create the final result, XGBoost aggregates the results of many decision trees. The final forecast is often a weighted sum of all the trees projections.

References:

https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

https://www.datacamp.com/tutorial/random-forests-classifier-python

Model Evaluation

The dataset has been split using train_test_split() function, which comes with library scikit-learn. The proportion of data used in train set is 70% and the data used in test set is 30%. After splitting the data, both the models were trained on the training set provided. Once the model gets trained, prediction were made by passing the test set.

To test the performance of the predictions made by both the algorithm, appropriate evaluation metric has been used such as accuracy score, precision score, recall and f1 score.

Random Forest Algorithm gave the following results:

```
Accuracy: 0.9831663801790054
Precision: 0.9880515570608712
Recall: 0.9597002650095952
F1 Score: 0.9736695716669757
```

XGBoost Algorithm gave the following results:

```
Accuracy: 0.9505660601031356
Precision: 0.9395317979338451
Recall: 0.9058759024033629
F1 Score: 0.922396947985484
```

Comparative Analysis

Performance of both the algorithm are as follows:

	MLA Name	MLA Accuracy	MLA Precision	MLA Recall	MLA F1 Score
0	Random Forest Algorithm	0.9837	0.9885	0.9608	0.9745
1	Xgboost Algorithm	0.9506	0.9395	0.9059	0.9224

Both the algorithms perform quite similar on the dataset. Random Forest achieves approximately 98% accuracy which means it correctly predicts the target label for majority of dataset. Whereas xgboost gives accuracy of 95% which it also predicts correctly expect in few cases where it predicted incorrectly compared to random forest.

The reason both perform similar might be that dataset used here does not have many features and it is also simple in structure. Also, default parameters have been used in both the model which can also be one of the reason as default settings are designed to perform well on variety of datasets.

STRENGTHS-

Random Forest Algorithm has high accuracy and is less likely to overfit because of the bragging and feature unpredictability.
XGBoost algorithm is more efficient as it takes only 19.7 ms to make one prediction.

WEAKNESS-

Random Forest Algorithm is less efficient than Xgboost on the given dataset as it takes around 479 ms to make one prediction, thus, it is slower for making predictions compared to XGboost. XGBoost algorithm may necessitate additional work in terms of hyperparameter adjustment and comprehension of its parameters.