<u>ML Assignment 1 - Exploring Data and Applying ML Algorithms</u>

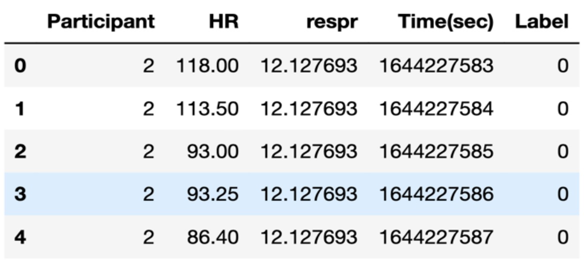
1. Introduction

Python and **Jupyter Notebooks** are the preferred language and platform for machine learning due to its extensive ecosystem of libraries and tools which provide powerful and user-friendly frameworks for building and training machine learning models. Not only this, Notebook merges together code, data visualisation and documentation, also allows cell by cell execution making it easier to debug and develop complex codes.

Pandas, a Python library, is imported for data exploration and data visualisation in tabular format. It allows the developer to read and manipulate data files and creates data frames to visualize the data as a table.

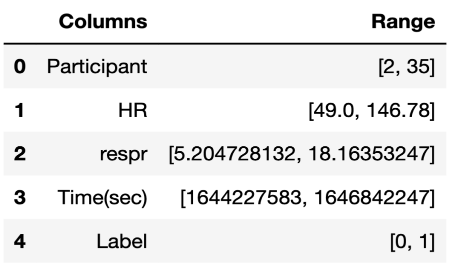
2. Data Understanding and Exploration

- Top 5 rows of the dataset:



	Participant	HR	respr	Time(sec)	Label
0	2	118.00	12.127693	1644227583	0
1	2	113.50	12.127693	1644227584	0
2	2	93.00	12.127693	1644227585	0
3	2	93.25	12.127693	1644227586	0
4	2	86.40	12.127693	1644227587	0

- Shape-1,12,515 rows and 5 columns
- Columns- Participant, HR (Heart Rate), respr (respiratory rate), Time(sec), Label
- Range of each feature:



	Columns	Range
0	Participant	[2, 35]
1	HR	[49.0, 146.78]
2	respr	[5.204728132, 18.16353247]
3	Time(sec)	[1644227583, 1646842247]
4	Label	[0, 1]

- Participant, HR, respr and Time(sec) – Continuous variable
- Label – Categorical Variable {0 (Not-stressed),1 (Stressed)}

According to the objective, we have to predict Label, that makes it the target/output variable or the dependent variable. All the remaining features are considered as Independent variables as these are the attributes of our data that are used to make predictions or classifications.

Independent Variables – [“Participant”, “HR”, “respr”, “Time(sec)”]

Dependent Variable – [“Label”]

Classification

Classification machine learning is implemented in supervised learning of categorical variable as the target variable, employed whenever there is a need to make decisions or predictions by assigning data points to predefined categories or classes.

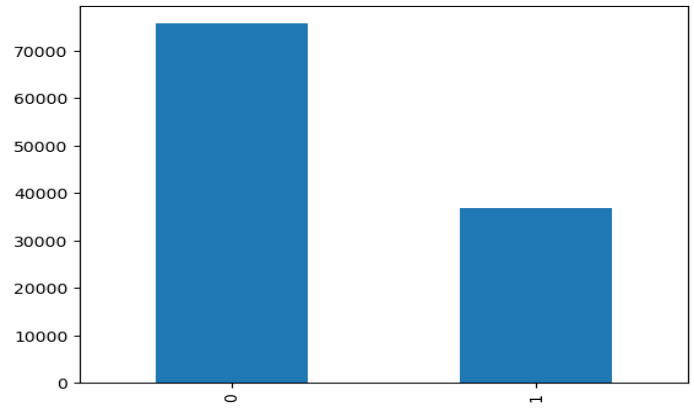
As our dependent variable is categorical (Label – {0 and 1}), hence we need classification machine learning techniques to classify the category of our target variable "Label".

Data Balance

Data imbalance is a term used in a situation when a category is present in significantly low amount, hence the model would not be able to predict it well as they have not seen enough examples of the minority class during training. This will create problems of Bias and poor generalisation.

2:1 ratio of classes in a dataset is often considered imbalanced, although the degree of imbalance may not be as severe as in datasets with more extreme imbalances. Minority class (label=1) is 33.3% while majority class (Label=0) is 66.7%. While it's not as extreme as situations where the ratio is, for example, 10:1 or 100:1, it can still present challenges for machine learning models depending on the objective.

In our case, **the dataset will tend towards being balanced more than being imbalanced**, so we don't have to use any data balancing techniques.



```
Frequency of 0s and 1s are:
0    75701
1    36815
```

3. Machine Learning Packages

Some of the Open-source packages available for machine learning practices:

- **Scikit-learn (sklearn):** Scikit-learn is one of the most widely used machine learning libraries in Python.

- **TensorFlow:** TensorFlow is an open-source machine learning framework developed by Google. It is known for its flexibility and scalability and can be used for various machine learning tasks, including classification.

- **PyTorch:** PyTorch is another popular deep learning framework in Python. It offers a dynamic computational graph.

Scikit-Learn (sklearn)

Choosing Scikit learn as it provides a comprehensive set of tools for classification, including various algorithms like logistic regression, decision trees, random forests, support vector machines, K-nearest neighbour, etc. It also offers utilities for pre-processing data, feature selection, and model evaluation which are used later. It can easily be imported in Notebooks with python kernel.

4. Data Pre-processing

Missing values: There were 44 rows where 'HR' value were NaN, these rows are dropped than imputing a mean or median value of 'HR' as the mean will include heart rates of different participants, which might make these rows outliers.

Null Values:

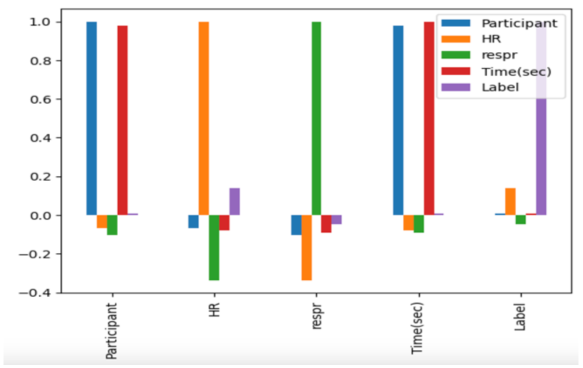
Participant	0
HR	44
respr	0
Time(sec)	0
Label	0

	HR	respr	Minutes
0	2.967592	-0.16628	-1.478119
1	2.614156	-0.16628	-1.477987
2	1.004059	-0.16628	-1.477855
3	1.023694	-0.16628	-1.477723
4	0.485686	-0.16628	-1.477591

Time formatting: In the dataset, Time format was UNIX, it was converted to Date and Time format and hence date was dropped. Then the Time is converted to minutes of that date.

Feature Scaling: As in this assignment, Random forest and Decision trees are used, which works on if-else conditions making them independent of scales of features. KNN is also implemented to get a wider perspective which uses distance, making scaling of the features important and necessary. Without scaling the features, distance based algorithms like KNN will give Bias to larger values in the dataset. We have used Standard Scaler feature of sklearn's pre-processing library.

Feature Selection: Plotting correlation bar graph of each feature. We found Participant as a redundant feature as it was strongly correlated with Time feature, hence dropped participant from Independent/Input variables. Time was an interacting feature as alone it was weakly correlated with dependent variable but with other features gives good contribution.



```
train_y.value_counts()
```

```
0    68091
1    33133
Name: Label, dtype: int64
```

```
test_y.value_counts()
```

```
0    7566
1    3682
Name: Label, dtype: int64
```

Splitting the data

Used the train_test_split() to split the dataset with test size as **10% or 0.1** value.

Also used **stratify** argument for dependent variable so that data will split with same proportion of categories in test and train as in the complete dataset to avoid data imbalance after splitting.

5. Algorithm Selection

Decision Tree:

- It's a supervised learning algorithm that builds a tree-like model of decisions and their possible consequences. Each internal node of the tree represents a decision or test on a specific feature, each branch represents the outcome of that test, and each leaf node represents a class label (in classification).
- Tree created with if-else conditions to reach the final prediction. Used under controlled depth of 25 layers. Sklearn.tree contains DecisionTreeClassifier.

Random Forest:

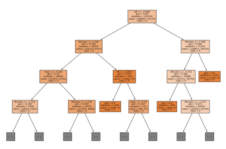
- Random Forest consists of a collection of decision trees, where each tree is trained on a random subset of the training data and may use a random subset of features. These randomization techniques introduce diversity among the trees.
- Forest of multiple decision trees, used under controlled depth of 25 layers and trees limit of 20 trees. Sklearn.ensemble contains RandomForestClassifier.

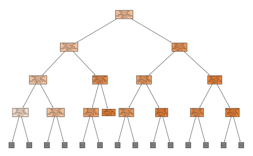
6. Model Application and Evaluation

- First, the classifier is initialised.
- Then the train_X with train_y is fitted in the model.
- Predict() is used to predict the outcomes of test_X and appended into pred_y.
- Lastly, pred_y and test_y are compared and classification metrics like Accuracy, score, F1-Score, Recall, Precision and confusion matrix are applied to analyse the model’s performance.

7. Comparative Analysis

```
Decision Tree

Training Score: 0.9762111752153639
Accuracy: 0.9696834992887624

Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      7743
           1       0.93      0.98      0.95      3505

    accuracy                           0.97     11248
   macro avg       0.96      0.97      0.97     11248
weighted avg       0.97      0.97      0.97     11248
```





```
Random Forest

Training Score: 0.9983501936299691
Accuracy: 0.9766180654338549

Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      7643
           1       0.95      0.97      0.96      3605

    accuracy                           0.98     11248
   macro avg       0.97      0.98      0.97     11248
weighted avg       0.98      0.98      0.98     11248
```

- As the tree depth was defined to control computation budget and also to avoid overfitting in both the algorithms, plotted trees with training score and classification report for both decision tree and random forest.
- Decision tree offered faster computation than random forest, but as random forest is ensemble of multiple random decision trees making it a better model compared to a single decision tree. It can be seen in the report that random forest has a better mapping to the training data with almost a perfect training score.
- There is a significant difference in accuracy and other metrics between the two models as we keep on lowering the tree maximum depth. This avoids the model to overfit.
- Random forest predicts the minority category (Label=1) better than the decision tree, as ensemble models handles imbalance in the data better than other models.

References:

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