Software Design Document

for

Fatigue Detection Model

Version 1.0

Prepared by

Team no - 14

Fayiz Umar

Joel James

Gautham Jayakrishnan

Suhana Jabin

TKM College of Engineering

12 April 2023

Table of Contents

1 Introduction

The Fatigue detection model is a machine learning model used for early detection and classification of fatigue levels to reduce accidents caused by fatigue.The model uses various dependent factors such as heart rate,blood pressure,sleeping patterns etc to determine the level of fatigue.Statistical anomaly and mean deviation methods are employed to detect irregular patterns and calculate a threshold value for fatigue prediction.

1.1 Purpose

The purpose of the Software Design Document is to provide a detailed overview of the system design of the model,the architecture and its development. It serves as a guide for the development team, stake -holders,testers,and the end users and provides a common understanding for the model.It outlines the key design decisions, architectural patterns, and technologies used to build the system and helps eliminate potential risks by following it

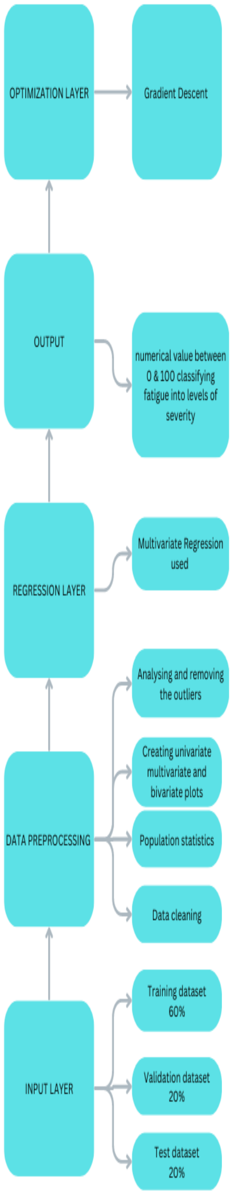
1.2 Scope

The scope of the Software Design Document provides a comprehensive overview of the design and architecture of the model and its technical aspects as well as the design decisions that were made during the development process, ensuring that the software meets the specified requirements and performs as expected. It helps to implement proactive measures to be taken to mitigate them. It is a critical reference document for the development team throughout the software development lifecycle.

1.3 Intended Audience

The Intended Audience of this SDD includes the development team,stakeholders involved testers and the end users.It includes the features that are to be included in the final product which are to be evaluated.

2.System Architecture



2.1 Description

The architecture consist of 6 layers namely
1.Input layer
2.Data preprocessing layer
3.Regression layer
4.Output layer
5.Loss function
6.Optimization
7.Testing

1.Input layer

This layer takes in the input data and passes it to the next layer for processing.

here the dataset is collected from the public domain and is divided into training validation and test set in a ratio of 60:20:20

2.Data preprocessing

In this layer the data is preprocessed by removing unnecessary headers. population statistics is calculated on basics of the data and univariate bivariate and multivariate graphs are plotted and a constant regular pattern is recognised and outliers are identified and removed based on the regular pattern

3.Regression layer

This layer takes the extracted features and uses them to predict the target variable.the graph is plotted using the training dataset

4.Output layer

This layer produces the output which are a numerical value between 0 to 100 which denotes how severe a person's fatigue is where 0 being the lowest and 100 the highest

5.Loss function

This function measures the difference between the predicted output and the true output, and is used to update the model parameters during training. The validation dataset is used and the algorithm is optimized accordingly

6.Optimization

Here we update the model parameters to minimize the loss function during training. Here we use the gradient descent algorithm.

7.Hyperparameter tuning

The machine learning algorithms contain hyperparameters that need to be tuned in order to achieve the best performance. This can be done through techniques such as grid search or random search, where different combinations of hyperparameters are tested on a validation set.

3.Data Pre - processing

3.1 a) Data Dictionary

The input data includes:

1. Age
2. Gender
3. Heart rate

4. Body mass index(weight / (height)^2)
5. Resting heart rate
6. Calories burned per minute.

id	age	gender	height	weight	hear_rate	calories	resting_heart	norm_heart	BMI	HR_max	HRR	HRR%	HR avg	FAS1	age_range
20	20	1	168	65.4	78.531302	0.3445329	59	19.53130238	0.0023172	200	141	13.851988	78.531302	8.4147841	20-0
20	20	1	168	65.4	78.45339	3.2876255	59	19.45339028	0.0023172	200	141	13.796731	78.45339	9.2645578	20-1
20	20	1	168	65.4	78.540825	9.484	59	19.54082508	0.0023172	200	141	13.858741	78.540825	11.160676	20-2
20	20	1	168	65.4	78.62826	10.154556	59	19.62825988	0.0023172	200	141	13.920752	78.62826	11.399049	20-3
20	20	1	168	65.4	78.715695	10.825111	59	19.71569468	0.0023172	200	141	13.982762	78.715695	11.637422	20-4
20	20	1	168	65.4	78.803129	11.495667	59	19.80312949	0.0023172	200	141	14.044773	78.80313	11.875795	20-5
20	20	1	168	65.4	78.890564	12.166222	59	19.89056429	0.0023172	200	141	14.106783	78.890564	12.114168	20-6
20	20	1	168	65.4	78.977999	12.836778	59	19.97799909	0.0023172	200	141	14.168794	78.977999	12.352541	20-7
20	20	1	168	65.4	79.065434	13.507333	59	20.06543389	0.0023172	200	141	14.230804	79.065434	12.590914	20-8
20	20	1	168	65.4	79.152869	14.177889	59	20.1528687	0.0023172	200	141	14.292815	79.152869	12.829287	20-9
20	20	1	168	65.4	79.240304	14.848444	59	20.2403035	0.0023172	200	141	14.354825	79.240304	13.06766	20-10
20	20	1	168	65.4	79.327738	15.519	59	20.3277383	0.0023172	200	141	14.416836	79.327738	13.306033	20-11
20	20	1	168	65.4	79.415173	15.506875	59	20.4151731	0.0023172	200	141	14.478846	79.415173	13.339602	20-12
20	20	1	168	65.4	79.502608	15.49475	59	20.50260791	0.0023172	200	141	14.540857	79.502608	13.373171	20-13
20	20	1	168	65.4	79.590043	15.482625	59	20.59004271	0.0023172	200	141	14.602867	79.590043	13.40674	20-14
20	20	1	168	65.4	79.677478	15.4705	59	20.67747751	0.0023172	200	141	14.664878	79.677478	13.440308	20-15
20	20	1	168	65.4	79.764912	15.458375	59	20.76491231	0.0023172	200	141	14.726888	79.764912	13.473877	20-16
20	20	1	168	65.4	79.852347	15.44625	59	20.85234712	0.0023172	200	141	14.788899	79.852347	13.507446	20-17
20	20	1	168	65.4	79.939782	15.434125	59	20.93978192	0.0023172	200	141	14.850909	79.939782	13.541015	20-18
20	20	1	168	65.4	80.027217	15.422	59	21.02721672	0.0023172	200	141	14.91292	80.027217	13.574584	20-19
20	20	1	168	65.4	80.114652	15.409875	59	21.11465152	0.0023172	200	141	14.97493	80.114652	13.608152	20-20
20	20	1	168	65.4	80.202086	15.39775	59	21.20208632	0.0023172	200	141	15.036941	80.202086	13.641721	20-21
20	20	1	168	65.4	80.289521	15.385625	59	21.28952113	0.0023172	200	141	15.098951	80.289521	13.67529	20-22
20	20	1	168	65.4	80.376956	15.3735	59	21.37695593	0.0023172	200	141	15.160962	80.376956	13.708859	20-23
20	20	1	168	65.4	80.464391	15.361375	59	21.46439073	0.0023172	200	141	15.222972	80.464391	13.742428	20-24
20	20	1	168	65.4	80.551826	15.34925	59	21.55182553	0.0023172	200	141	15.284983	80.551825	13.775996	20-25
20	20	1	168	65.4	80.63926	15.337125	59	21.63926034	0.0023172	200	141	15.346993	80.63926	13.809565	20-26

3.2 Data Cleaning

1. Removing Missing Values.
2. Removing blank rows.
3. Removing Unnecessary Headers.

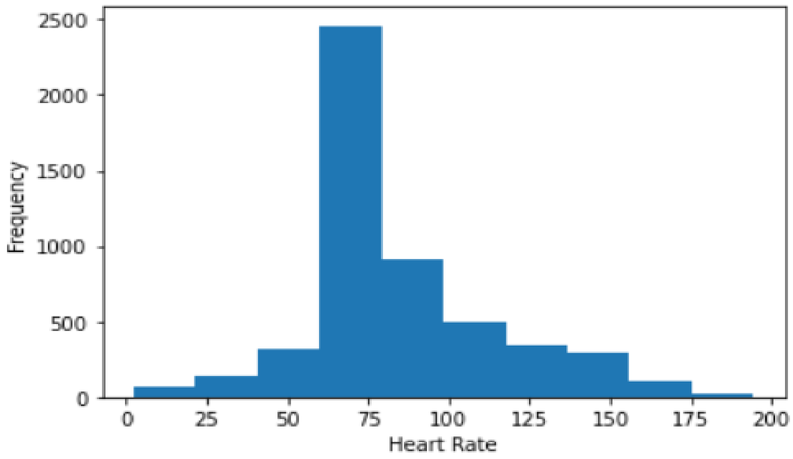
3.3 Calculating Population Statistics

1. Grouping data based on age ,gender factor.
2. Calculating mean of each feature and determining average and standard deviation.
3. Analysis of Outliers in data.
4. Univariate, bivariate and multivariate analysis.

Univariate Plots

In [25]:

```
plt.hist(new['hear_rate'], bins=10)
plt.xlabel('Heart Rate')
plt.ylabel('Frequency')
plt.show()
```

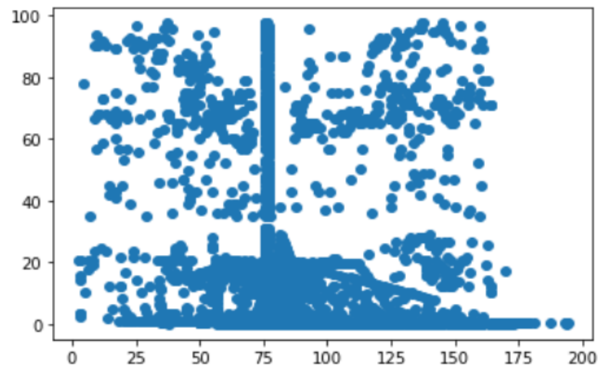


Created a histogram of the heart_rate variable. The bins argument specifies the number of bins to use in the histogram, and the xlabel() and ylabel() functions set the labels for the x-axis and y-axis, respectively.

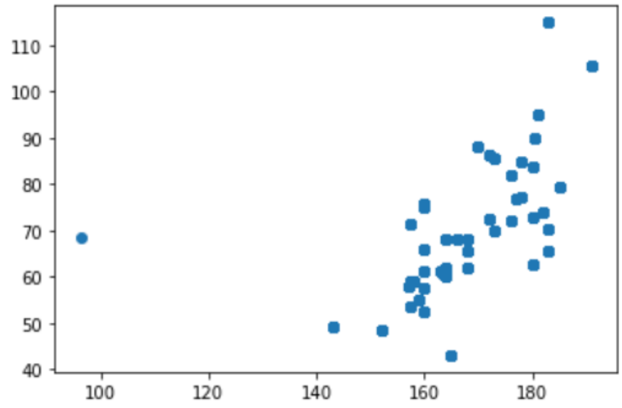
Bivariate Plots

In [85]:

```
plt.scatter(new.hear_rate,new.calories)
plt.show()
```



In [24]:
```
plt.scatter(new.height,new.weight)
plt.show()
```



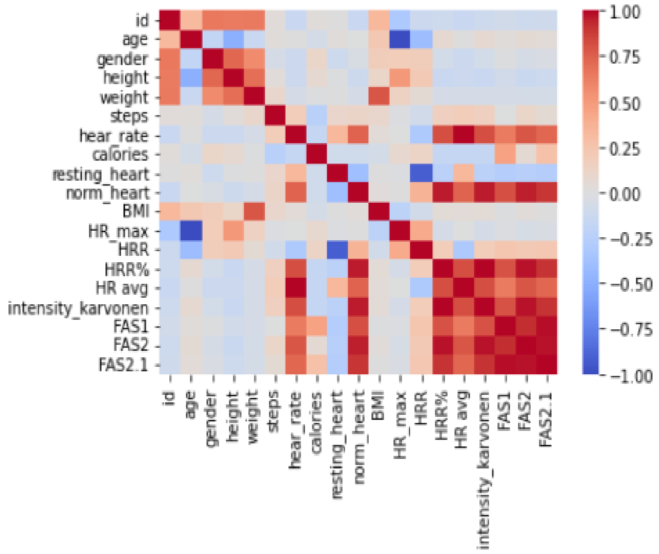
The scatter plot will display each data point as a point on the plot, with the x-axis representing the heart_rate values and the y-axis representing the calories values. Similarly height and weight.

This is useful for visualizing the relationship between heart_rate and calories, which can be helpful in understanding how these variables are related and identifying any patterns or trends in the data.

Similarly , Bivariate plots were generated for all the correlated features.

Multivariate Plots

```
In [86]: sb.heatmap(corr_matrix, cmap='coolwarm')
         plt.show()
```



Heatmap is generated using the Seaborn library in Python to visualize the correlation matrix of the variables age, heart_rate, calories, and BMI in the DataSet named 'new'.
positive correlations shown in shades of red and negative correlations shown in shades of blue. A darker color indicates a stronger correlation, while a lighter color indicates a weaker correlation.

Finding out Regular pattern

```
In [134]: import numpy as np

          # Define age ranges
          age_ranges = [(10, 20), (20, 30), (30, 40), (40, 60), (60, 80)]

          # Add a new column with the age range for each row
          new['age_range'] = pd.cut(new['age'], bins=[a[0] for a in age_ranges] + [age_ranges[-1][1]], labels=[f"{a[0]}-{a[1]}" for a in a

          # Group by id and age_range and calculate the mean and std of hear_rate
          regular_pattern_heart = new.groupby(['id', 'age_range'])['hear_rate'].agg(['mean', 'std'])
```

```
In [135]: regular_pattern_heart
```

Out[135]:

id	age_range	mean	std
0	10-20	NaN	NaN
	20-30	NaN	NaN
	30-40	NaN	NaN
	40-60	NaN	NaN
	60-80	NaN	NaN
...	...	...	...
47	10-20	NaN	NaN
	20-30	NaN	NaN
	30-40	89.054019	28.414681
	40-60	NaN	NaN
	60-80	NaN	NaN

For creating a summary of the heart rate patterns for different age ranges, which is helpful in identifying Regular Pattern in heart rate across the age groups. The resulting regular_pattern_heart DataFrame can be further analyzed and visualized to gain insights into the data.

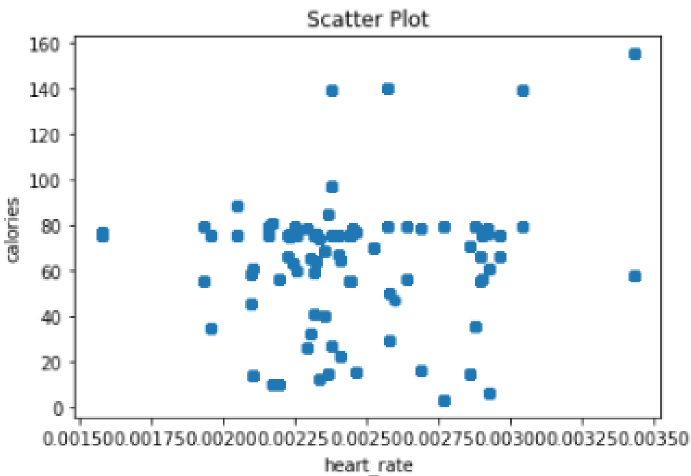
Similarly, the Regular Pattern for every Feature is calculated.

3.4 Analysing the Outliers in the dataset

1. Univariate Outliers : Univariate outliers are the data points whose values lie beyond the range of expected values based on one variable.

2. Multivariate Outliers : While plotting data, some values of one variable may not lie beyond the expected range, but when you plot the data with some other variable, these values may lie far from the expected value.

```
In [83]: plt.scatter(new['BMI'], new['resting_heart'])
         plt.xlabel('heart_rate')
         plt.ylabel('calories')
         plt.title('Scatter Plot')
         plt.show()
```

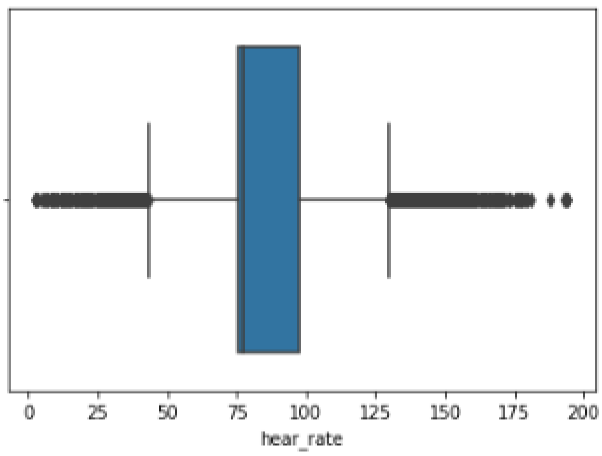


Outliers are the data points that are significantly different from the rest of the data points and are located far away from the main cluster of points. Multivariate outliers are the points that deviate significantly from the general pattern of the data points.

Detecting Outliers using BoxPlot

```
In [31]: import seaborn as sb
         sb.boxplot(x=new['hear_rate'])

Out[31]: <AxesSubplot:xlabel='hear_rate'>
```



In the above graph,we can clearly see that values above 125 are acting as the outliers. Similarly Outliers are detected and eliminated for each feature.

4.Conceptual Architecture

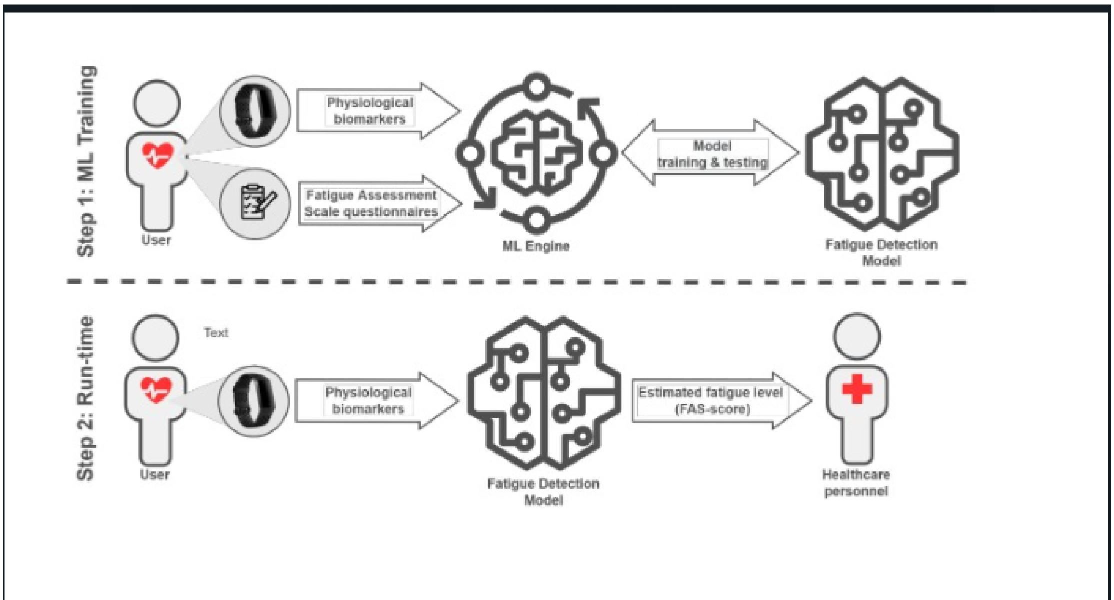


Figure 1: Conceptual architecture of the proposed approach.

Multiclass Classification : Based on population statistics, grouped variations are plotted and fatigue on a range of 0 - 100 are calculated.

The goal is to classify instances into one of three or more classes or categories. A single input data point is assigned to one of several possible output categories.

Here Data from wearables is taken and then divided into training set , test set and validation set.Now the data is then given as input to the ML algorithm and categorical output is predicted.

5.Optimization

After we have calculated the loss function the model is optimized using gradient descent algorithm.The algorithm is used to minimize the cost

function by iteratively adjusting the parameters of the model in the direction of the steepest descent until we get the optimal values.
the gradient descent algorithm is chosen due to its ease of implementation efficiency and flexibility

6.Technological Stack

6.1 Python

Python is widely used in ML for data preparation, model development, evaluation, deployment, and visualization. Libraries like Scikit-learn, TensorFlow, PyTorch, Flask, and Matplotlib offer easy development, evaluation, and deployment of models.

6.2 Jupyter Notebook

Jupyter Notebook is a popular tool for developing and evaluating machine learning models for fatigue detection. It provides an interactive environment for processing physiological signals, implementing machine learning algorithms, and visualizing results.

6.3 Tensorflow

TensorFlow is crucial for fatigue detection due to its deep learning capabilities, flexible architecture, high performance, large community, and diverse deployment options, making it ideal for building custom models, experimentation, optimization, and seamless integration into various environments.

6.4 Pandas

Pandas is crucial in fatigue detection ML models as it preprocesses and manipulates data for clean and organized input. It offers essential functionalities for data cleaning, transformation, and aggregation, enabling insights and exploration of data.

6.5 Matplotlib

Matplotlib is essential for fatigue detection ML models, enabling visualizations for data understanding, identifying fatigue patterns, and interpreting results. Enhances data communication, improving accuracy and interpretability.

7.References

https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=age+and+heartrate&oq=age+and+heartra#d=gs_qabs&t=1681271311472&u=%23p%3D0rn7StCACsUJ

https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=heart+rate+and+gender&oq=heartrate+and+gender#d=gs_qabs&t=1681271620717&u=%23p%3DQoN6rgcxkXoJ

https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=resting+heart+rate&oq=resting+heart+#d=gs_qabs&t=1681271638553&u=%23p%3DQp_IIEse-j0J

https://arxiv.org/pdf/2108.04022.pdf

https://dl.acm.org/doi/pdf/10.5555/1151758.1151781

https://livrepository.liverpool.ac.uk/3007443/2/HRV-Based.pdf