D. Pre-processing of the EEG Signals

The recorded brain signals are subjected to pre-processing to remove artifacts, segmentation of frames, and frequency bands segmentation. The frame signals are band-pass filtered within 0.1 to 100 Hz (frequency range) and split into six frequency band signals i.e. delta (δ), theta (θ), alpha (α), beta (β), gamma1 (γ1), and gamma2 (γ2). The segmented frequency band signals are used to extract relevant features using four different nonparametric feature extraction methods.

E. Feature Extraction

The pre-processed frequency band signals reside valuable information in classifying driver's vigilance (fatigue and alert) model, which is the data involves are too big and consumes high computational power to process and resulting in increases of the computational time. This valuable information can be transformed into a reduced-size of a set of features using a relevant feature extraction algorithm. The transformed features set represent desirable information to visualize, verification, and classification of the model. Thus, feature extraction is the process of identifying dominant characteristics of the bioelectric signals and representing the feature vectors with minimum data size and minimal loss of information in classifying the driver's vigilance level.

Henceforward, in this research, the pre-processed signals were analyzed in non-linear frequency domain i.e. power spectral density feature extraction algorithm in the frequency domain. The pre-processed signals were also used to extract spectral estimation features using periodogram, Welch, Lomb-Scargle, and Thompson's multitaper method. Fig. 4 represents the overview of the feature extraction approaches used in this research.

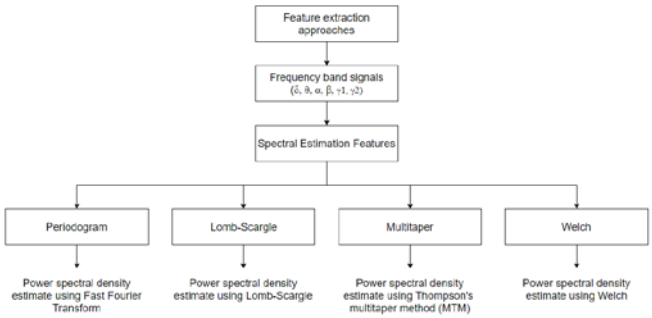

Fig. 4. Overview of the feature extraction approaches used in fatigue and alertness model

The extracted features are correlated with the corresponding driving tasks (Alert Driving and Fatigue Driving) and consequently used as input to feature classifiers for the classification of Alert Driving/Fatigue Driving tasks. The database of extracted features is then validated using k-fold cross-validation and subjected to machine learning algorithm for the purpose of classification.

F. Feature Classifiers

Feature classifier is an essential part of most classification and pattern recognition algorithm. To develop an adaptive fatigue and alertness model, a classifier model has to be developed to recognize the fatigue and alertness state corresponding to the designed data acquisition protocol. Finding the appropriate classification algorithm is quite challenging because there is no perfect method to accurately estimate the output that fits all. Therefore, selecting the suitable algorithm involves trading off one advantage against another, including accuracy, complexity, and model speed. Higher predictive accuracy can be achieved from the classification learning process by selecting only the relevant characteristics of the data.

1) Multilayer Neural Network (MLNN)

Training in MLNN involves adjusting the values of the weights and biases of the network to enhance network performance. Through the training procedures, the MLNN adjusts the synaptic weights in response to the input, so that the actual output response is approximately the same as the desired response. In this study, the Levenberg–Marquardt algorithm is used for the neural network training.

The Levenberg–Marquardt algorithm [15], [16] was developed by Kenneth Levenberg and Donald Marquardt for solving the problem of minimizing a nonlinear function. In neural network training, the Levenberg–Marquardt algorithm combines the features from both gradient descent method and Gauss-Newton method and it is considered as one of the most efficient training algorithms [17] and widely used in backpropagation method. The training procedures using the Levenberg–Marquardt algorithm [18] is defined as follows:

- Step 1: Initialize weights (w_{ij})
- Step 2: Evaluate the total error using the sum squared error (SSE) using (3).
- Step 3: Update new weights using (4).
- Step 4: With the new adjusted weights, evaluate the total error using the sum squared error (SSE).
- Step 5: If the new total error is increased as a result of the update, then retract the step, and increase the parameter μ by a factor β, update the weights by Δw and proceed to step 6. If the new error is decreased as a result of the update, then accept the step, and decrease the parameter μ by a factor β and go to step 5.
- Step 6: If the current total error is less than the required value, stop training, otherwise go to step 2 with the new weights.

$$E_i = \frac{1}{2}\sum_{j=1}^{N^L}\left(t_{ij} - y_{ij}\right)^2 \tag{3}$$

Where E_i is the sum squared error, t_{ij} is the target values, N^L is the length of the training pattern, and y_{ij} is the output.

$$w_{i+1} = w_i - (J_i^T J_i + \mu I)^{-1} J_i e_i \tag{4}$$

Where w_{i+1} is the new adjusted weights, $J_i^T J_i + \mu I$ is the new approximation of Hessian matrix, J_i is the Jacobian matrix, μ is the combination coefficient (always positive), and I is the identity matrix.