# Driver Drowsiness Detection Using ECG Signals and Machine Learning Models

**Meshal Mansour Alnasheet**

Higher Institute of Telecommunication and Navigation, HITN, PAAET, Shuwaikh Industrial, Kuwait (Specialist trainer A)
*mm.alnasheet[at]paaet.edu.kw*

**Abstract:** ***Fatigue and drowsiness are responsible for a significant percentage of road traffic accidents. There are several approaches to monitor the driver's drowsiness, ranging from the driver's steering behavior to analysis of the driver, e.g. eye tracking, blinking, yawning or electrocardiogram (ECG). This paper describes the development of a low-cost ECG sensor to derive heart rate variability (HRV) data for the drowsiness detection. The work includes the hardware and the software design. The hardware has been implemented on an Arduino using ECG AD 8232 model attached to a Raspberry Pi device for processing purposes. The digital ECG signal is transferred to a Raspberry Pi embedded PC where the processing takes place, including QRS-complex, heart rate and HRV detection as well as visualization features. The compact resulting sensor provides good results in the extraction of the main ECG parameters. Different machine learning algorithms are implemented to classify the ECG signals into mainly two categories (Sleep and Awake). Support Vector Machine using the Radial Bias Function Kernel (RBF) achieved accuracy of 95% in inference stage. Another Decision Tree classifier has been also designed and also produced a high accuracy of 98% during the evaluation phase.***

**Keywords:** Machine Learning, Signal Processing, ECG Signals, Drowsiness Detection, Roads Safety

## 1. Introduction

Safety driving is one of the most required factors and services now days in car production and development society. Mega auto mobile manufacturing companies are trying to provide a very big added value to their systems in order to ensure the highest safety and security for drivers and people using the cars. Drivers fatigue is estimated to cause more than 10% of traffic accidents based on the USA reports showing that between 10% and 25% of accidents happening in the USA are caused by drivers' fatigue or driver sleepiness. The increasing number of accidents caused by fatigue drivers has enforced the institutions, vehicle developers and manufacturers to find a reasonable solution for this issue and provide a reliable technology to be attached and installed on their car's prototypes to warn, notify and wake up the driver once the drowsiness has been detected. Another relevant context can be also the development of the self-driving cars where the service of driver drowsiness detection can be also plugged and used to provide another high quality and valuable factors to the self-driving cars. In this paper, it is targeted to design, develop and implement a standalone system that can deal and interact with different electrical, electronic devices and software solutions to detect and specify if the driver in fatigue mode or sleepiness mode. This system should be depending on different technologies and methods in order to finalize the requirements using different mathematical models and hardware prototypes. Different approaches also have to be discovered and well-studied suck like developing smart based system depending on Machine Learning and Signals/Image processing, or developing a fully hardware based solution using different sensing mechanisms to interact with the driver and specify the drowsiness case. Another factor is also to be achieved is to build, design and implement a warning/notification system that can notify the driver at least once the drowsiness case has been detected. Expected to have a final prototype that can detect drowsiness, notify driver and realize the most important safety factors for the driver and anyone using the car.

## 2. Related Work

**In 2008, Hong Su et. al. [1] described 'A Partial Least Squares Regression-Based Fusion Model for Predicting the Trend in Drowsiness'.** They designed and implemented a new method of characterizing driver drowsiness with a set of multiple eyelid movement features generated by a set of fusion sensors based on an information fusion technique-partial least squares regression (PLSR), in order to discover the strong relations among eyelid movement patterns and, the tendency of the drowsiness. The predictive PLSR model could perform properly on the extracted features coming from the eyelid fusion sensors readings and could as well finalize an acceptable accuracy of drowsiness detection up to 95%. **In June, 2010, Bin Yang et. al. [2] described 'Camera-based Drowsiness Reference for Driver State Classification under Real Driving Conditions'.** They have design a computer vision based solution monitoring the driver's face and detecting the eye movements using different image processing and machine learning models to extract eyes, ears and mouth of driver to classify the status of both eyes of the driver (Closed, Open).The Cascade Face Mask model has been used to detect the face, eyes, ears and mouth of the driver and then a simple machine learning model has been trained using Support Vector Machine (SVM) did the classification process and classified all incoming frames from camera into (Sleep or Awake). As a summary, the camera based sleepiness measures provide a valuable contribution for a drowsiness reference, but are not reliable enough to be the only reference. **In June, 2012, A. Cheng et. al. [3] described 'Driver Drowsiness Recognition Based on Computer Vision Technology'.** "They designed a stable drowsiness recognition system using eye-tracking and image processing. A stable and robust eye detection algorithm is designed to address the problems caused by changes in illumination and driver posture. Six different values are calculated along with percentage of eyelid closure, maximum closure duration, blink frequency, average opening level of the eyes, opening velocity of the eyes, and closing velocity of the eyes. These measures were combined using Fisher's linear discriminated functions based on a stepwise kernel method to reduce the correlations. The system has been tested using a driving simulator and the results could achieve 86% of accuracy using six different drivers. **In June, 2014, Eyosiyas et. al. [4] described 'Driver Drowsiness Detection through HMM based Dynamic Modeling'.** They have designed a completely new approach for analyzing facial expressions of the driver using Hidden Markov Model (HMM) to detect drowsiness. The complete design and system were implemented using a virtual simulated driving environment and the overall system accuracy achieved over 90% of classification rate.

## 3. Methodology

### 3.1 Design Overview

Our proposed solution was designed based on the ECG module Ad8232 heart signal recorder. Which is communicating with an Arduino microcontroller in which the signal is being sent over USB serial port to a Raspberry Pi computer where all of the processing, and decision making procedures are implemented. Below we present the high level design of entire solution and explaining each block in details:

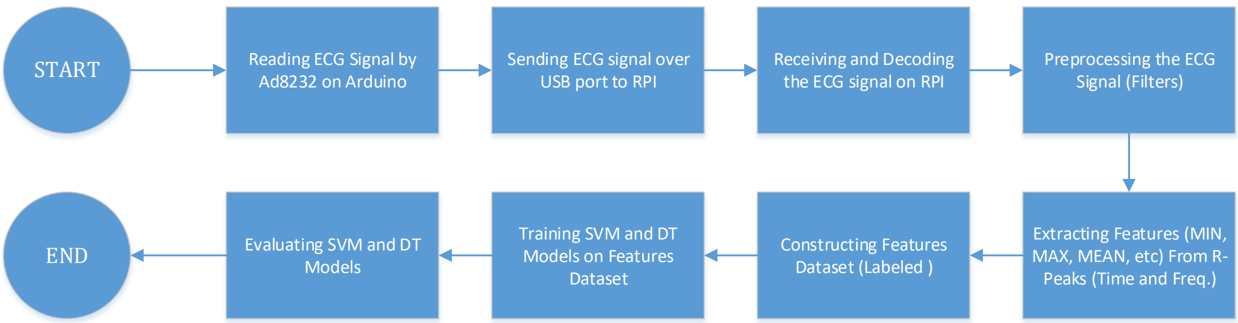


**Figure 1:** System High Level Diagram

**a) Generating ECG Signal**

The selected ECG module is connected on 2 digital pins (10, 11) as well as on any analog pin (A0) where the analog signal produces the signal of ECG and the digital ports applying the pulse train job in order to construct a sin-wave signal. All readings coming from the module are in range of 0-1023 value (0-5 volts). This signal is being taken by Arduino. The Arduino device is connected with a Raspberry Pi-4 (RPI) computer over an USB cable on USB port where the signal of ECG is written as a serial byte array with 10 millisecond delay for stability purposes. The RPI device receive the signal as a byte array, so the RPI applies a decoding processes in order to retrieve the numerical data signal and store it in a vector data structure. The following fig (2) shows the circuit connections including all components used in our project and its schematic diagram as well

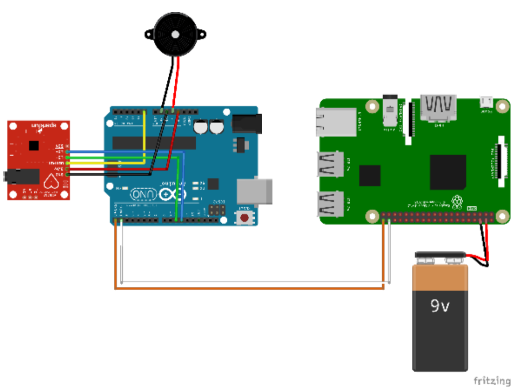


**Figure 2:** Circuit Diagram

The following table includes all components with all connection specifications:

**Table 1:** Components Table

| Component | Input / Output | Digital / Analog | Pin | Power |
|---|---|---|---|---|
| Raspberry Pi 4 Model B | Both | Both | - | 12v |
| ECG Ad8232 | Input | Analog/Digital | Analog – A0 Digital (10-11) | 5v |
| Digital Buzzer | Output | Digital | D4 | 5v |
| Arduino Uno R3 | Both | Both | - | 5v |

Using the ECG-Ad8232 module and Arduino MCU, we have recorded 100 ECG signals from 5 different drivers in awake mode as well as 20 signals from single driver in sleep mode. These signals are saved and stored locally on the Raspberry Pi device where they are used in the further features extraction, training and testing processes. The sampling rate of each recorded signal is 256Hz as well as each ECG recorded signal contains 5000 samples in range of [0-1023] as per the ECG module specifications.

**b) Preprocessing ECG Signal and Peaks Calculation**

In this stage the signal is received and decoded properly. Not all ECG signals are coming in best form with 0 noise. That's why we have to apply a kind of cleaning process on the signal to remove any possible noise. 4 different filters where combined and used together to remove any possible noise (Low-Pass, High-Pass, Band-Pass, Notch) filters. Once the ECG signal has been filtered and all possible noise are eliminated, then a kind of R-Peaks models is implemented to extract the peaks vector of the signal from time domain and frequency domain. All sleeping features are expected to locate in this peak vector. Figure (3 and 4), present a sample of a filtered ECG signal with all related R-Peaks values in time domain from awake and sleeping drivers' signals.

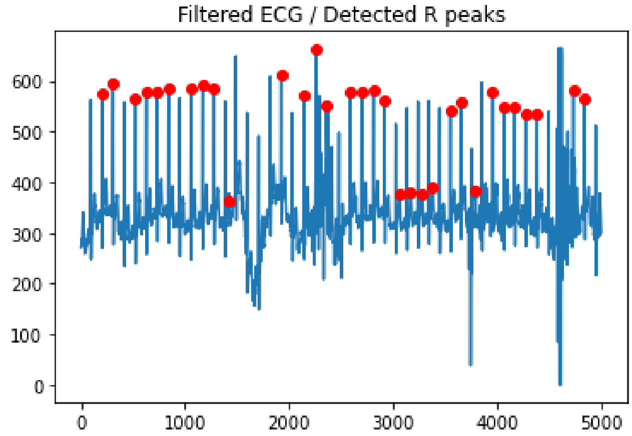


**Figure 3:** Filtered ECG with R-Peaks Indices from Awake Signal in Time Domain

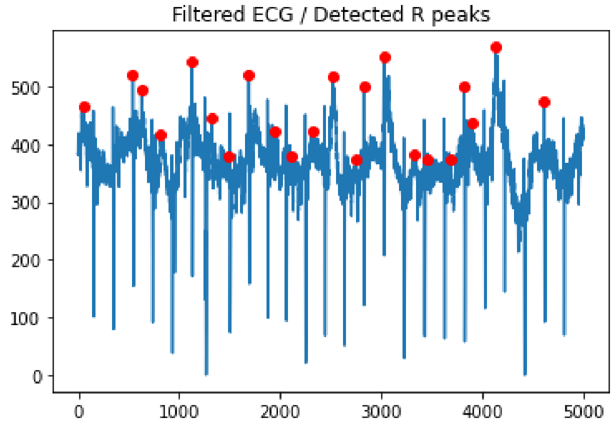


**Figure 4:** Filtered ECG with R-Peaks Indices from Asleep Signal in Time Domain

**c) Extracting Statistical Features / Creating Dataset / Features Normalization**

From the peak vector, the following statistical values (MAX, MIN, MEAN, MEDIAN, STD) are calculated from the Peak-Time-Domain and Peak-Frequency-Domain. Each Feature's vector extracted from each received signal is being stored in a Dataset with its related binary label (0 or 1) (sleep or awake). This dataset will be used in the training phase. A Z-Score normalization model has been applied on the training dataset so that we ensure the features consistency and stability. Below in Table (2) we show a capture of some basic statistics about the generated and normalized training dataset.

**Table 2:** Summary statistics about the normalized training dataset

| | MAX_TIME | MIN_TIME | STD_TIME | MEAN_TIME | MEDIAN_TIME | MAX_FREQ | MINT_FREQ | STDT_FREQ | MEAN_FREQ | MEDIAN_FREQ |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 | 1.200000e+02 |
| **mean** | 2.622902e-16 | 5.870304e-16 | 6.707597e-16 | 6.268365e-15 | -2.983724e-16 | 2.817191e-16 | 9.992007e-17 | -1.155557e-15 | -3.854324e-15 | 2.018756e-15 |
| **std** | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 |
| **min** | -2.106129e+00 | -4.913109e+00 | -1.717971e+00 | -1.441264e+00 | -1.290854e+00 | -1.332548e+00 | -2.270394e+00 | -9.784267e-01 | -3.428982e+00 | -2.722804e+00 |
| **25%** | -6.343694e-01 | -3.606871e-01 | -6.761964e-01 | -8.928131e-01 | -1.006107e+00 | -7.115424e-01 | -3.122150e-01 | -5.561899e-01 | -6.458232e-01 | -4.964511e-01 |
| **50%** | 3.202108e-01 | 2.129056e-02 | -2.293695e-02 | 9.092803e-02 | -6.349534e-02 | -3.038117e-01 | 3.335675e-01 | -2.742284e-01 | 2.547079e-02 | 4.094442e-02 |
| **75%** | 9.199182e-01 | 4.471017e-01 | 6.581241e-01 | 8.125744e-01 | 6.598279e-01 | 4.112853e-01 | 6.668745e-01 | 1.448789e-01 | 5.969749e-01 | 5.783399e-01 |
| **max** | 1.029956e+00 | 3.001968e+00 | 3.016541e+00 | 2.337159e+00 | 2.571234e+00 | 2.845124e+00 | 1.135588e+00 | 4.811606e+00 | 2.354564e+00 | 3.111776e+00 |

**d) Training and Evaluating SVM and DT models**

Once the dataset is created, a Support Vector Machine learning model with an RBF Gaussian Kernel is designed and implemented to train over the dataset and construct a trained model for further classification and testing purposes. Another Decision Tree based model is also created and trained over the training dataset to provide more stability to our proposed design. Both models are saved locally and used in further evaluation processes. Once the SVM and DT models are trained and stored locally, the system will start receive new signals from the ECG device and apply on them all previous stages (filtering, features extraction) and then apply the SVM trained model to find the label (0 or 1) (sleep or awake). The below flowchart describe the entire training process:

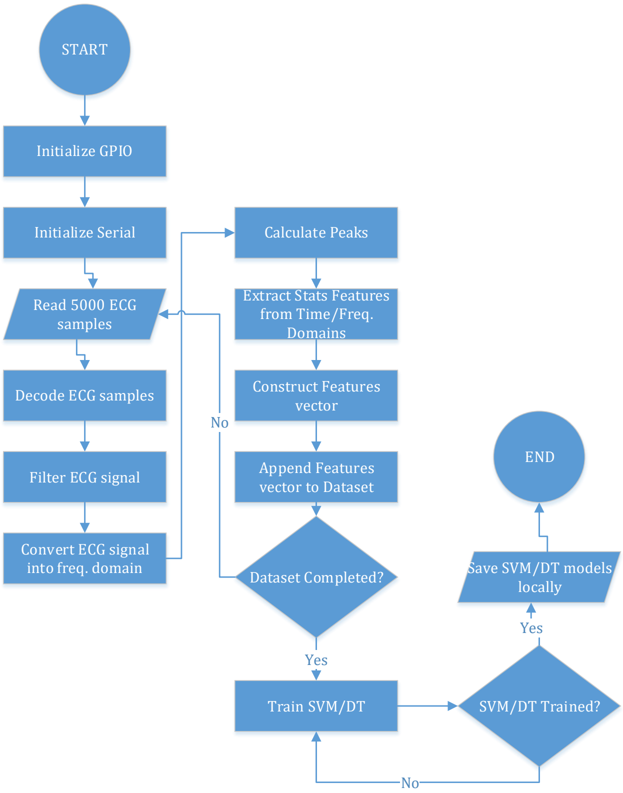


**Figure 5:** ML Training Process on RPI

List of functions implemented in the training phase are described in the below table:

**Table 3:** Functions in Training Phase

| Function/Procedure | Input | Output | Description |
|---|---|---|---|
| Create Dataset | Folder Containing all recorded ECG signals from all drivers | A data frame labeled Matrix (Dataset.csv) | Imports all signals recorded from all drivers. Applies Filters (High, Low, Bandpass and Notch) on each individual signal. Converts each signal into frequency domain. Calculate peaks vector. Extracts Statistical features from time and frequency domains of peaks vector. Store features with related label into a matrix. Normalize the final dataset matrix using z-score normalization. Applies outliers' detection on Dataset on each feature. Save the Dataset locally as "csv" file. |
| **createMLmodel** | Training Dataset | A stacked struct containing SVF-RBF kernel trained model with DT model. | Imports training dataset. Splits training dataset into 70% training and 30% testing. Call built-in SVM model with a Gaussian RBF kernel and DT model. Trains SVM model till learning rate 0.0000001 reached. Train DT classifier. Validate models with 30% created testing set. Calculates confusion matrix for both models. Save models locally for testing purposes. |

The below flowchart describe the entire evaluation process:

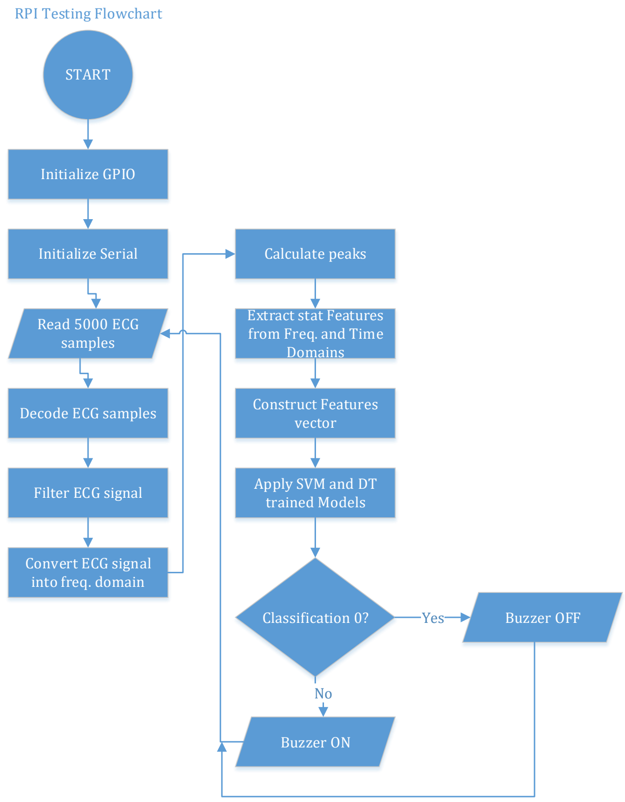


**Figure 7:** RPI Testing Flowchart

List of functions implemented in the evaluation phase are described below

**Table 4:** Functions in Testing Phase

| Function/File Code | Input | Output | Description |
|---|---|---|---|
| Receive ECG signal | Port Address and Baud Rate | An ECG signal vector containing 5000 samples | Creates a serial instance on ttyACM0 port number and with 9600 bit/s baud rate. Reads incoming serial data byte by byte. Decodes serial data. Stores serial data into an ecg_signal_vector. |
| Predict Driver Status | Serial ECG signal | Predicted driver status (0 awake OR 1 asleep) and Sound buzzer output. | Applies Filters (High, Low, Bandpass and Notch) on ECG serial signal. Converts ECG signal into frequency domain. Calculate peaks vector. Extracts Statistical features from time and frequency domains of peak vector. Stores features into a features vector. Normalize the final features vector using z-score normalization. Imports the trained SVM/DT models from local drive. Calls SVM/DT model on Features vector instance. Predicts the status of driver. Runs buzzer in case predicted value is asleep. |

### e) Classification Results and Discussions

Two machine learning models were implemented, designed and tuned in order to classify the incoming ECG recorded signal from the AD8232 device attached to the driver chest. The Decision Tree (DT) and Support Vector Machine (SVM) classifiers perform perfectly during the training phase as well as the testing phase. Below we list all evaluation criteria, parameters, matrix, and factors.

- **Accuracy an Area Under Curve (AUC)**

**Table 5:** ACC/AUC Comparison

| Model | Accuracy | AUC | F1 | Prec. |
|---|---|---|---|---|
| DT | 0.9533 | 0.9880 | 0.9535 | 0.9353 |
| SVM | 0.9621 | 0.9901 | 0.9884 | 0.9533 |

- **Confusion Matrix**

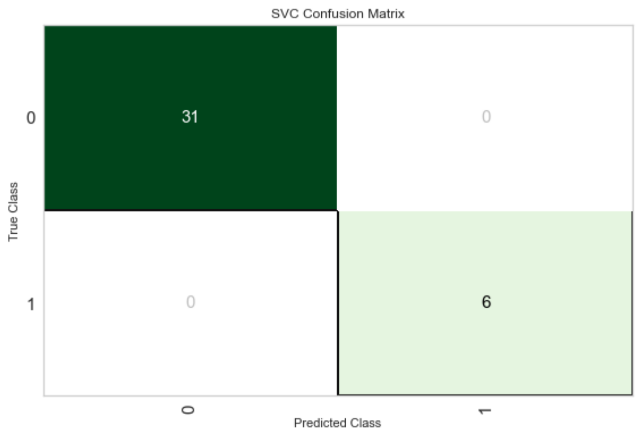


**Figure 8:** SVM Confusion Matrix

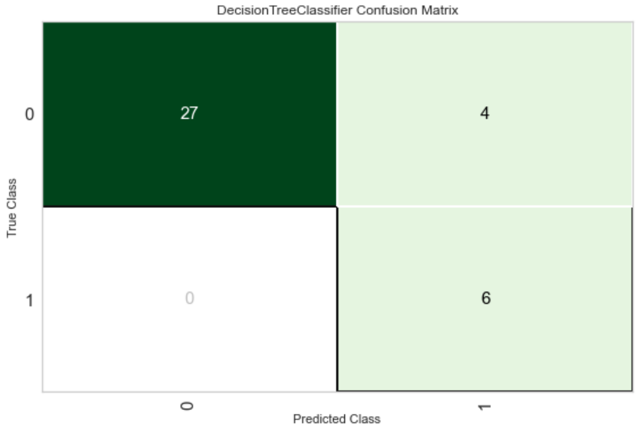


**Figure 9:** DT Confusion Matrix

- **Learning Curve**

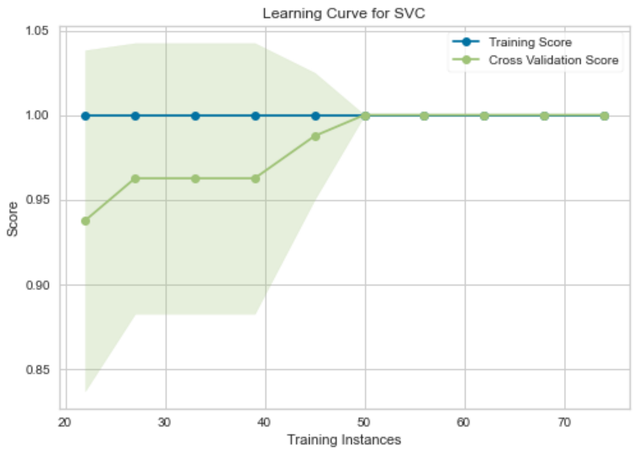


**Figure 10:** SVM Learning/Cross Validation Curves

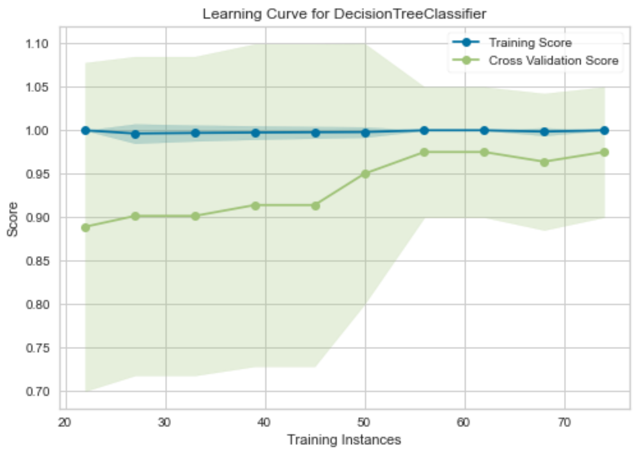


**Figure 11:** DT Learning/Cross Validation Curves

- **Decision Boundaries and Visual Decision Tree**

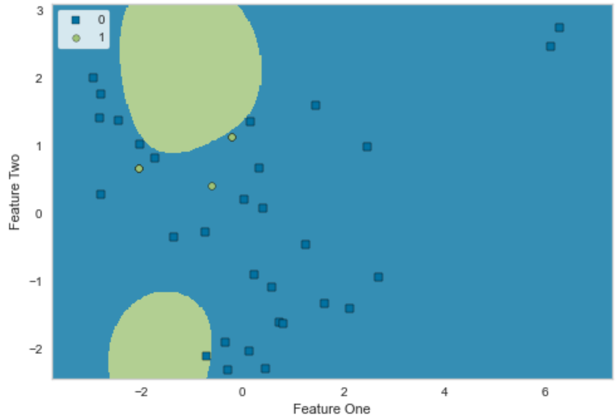


**Figure 12:** SVM Decision Boundaries

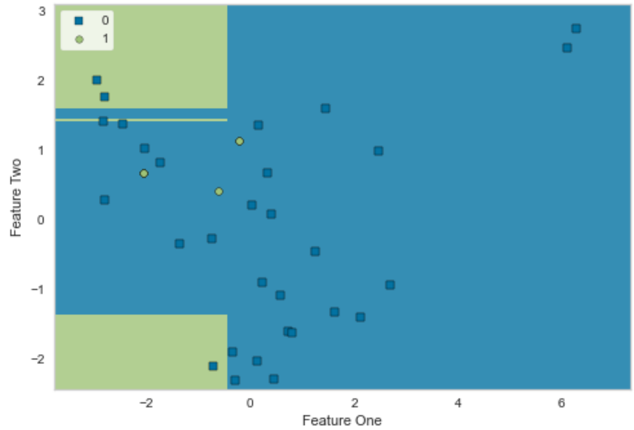


**Figure 13:** DT Decision Boundaries

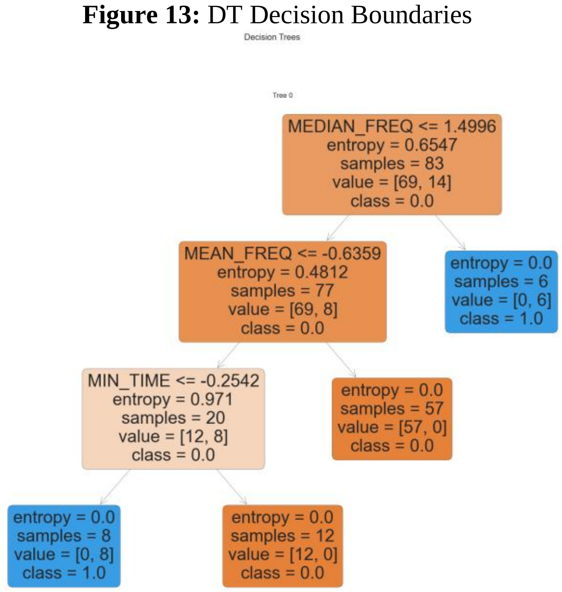


**Figure 14:** DT Representation

Based on all previous evaluation criteria and parameters, we can conclude that both models are performing in a very good range with high accuracy and scoring. SVM is showing more accuracy and robustness in terms of the AUC and this is because of the strength behind the RBF kernel which can deal with different data distributions and dimensions.

## 4. Conclusion and Future Work

Machine learning based solutions provide a huge added value in terms of complex data classifications and pattern recognition, in our proposed design, the SVM and DT based models could help us potentially by the high dimension of the recorded ECG signal and by the very tiny differences between asleep and awake signals. Statistical features have very simple and direct format so that they have enriched the design by providing some very clear stats about each recorded signal. What we actually have noticed is that the features distributions between awake and asleep signals are almost same, but the major difference was the power of the statistical features in the asleep signal, in which it was 100 times higher than it is in the awake signal. Filtering process is extremely important and highly recommended because of the high noise existence inside the ECG signal coming from the Ad8232 module, which helps also to reduce the error during the classification and prediction process. The communication between Arduino and RPI is done by the USB port and was fair enough to finalize the main requirement. Mainly the design is stable and accurate, and some points can be added in the future just to improve the design and enhance the result of the classifier. Implementing another set of machine learning models like ANN and NB can may provide better than SVM accuracy and stability. Changing the communication between Arduino and RPI to become wireless can also simplify the prototype design at the driver end.

## References

[1] Hong Su and Gangtie Zheng, "A Partial Least Squares Regression-Based Fusion Model for Predicting the Trend in Drowsiness" IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 38, NO. 5, SEPTEMBER 2008.
[2] Fabian Friedrichs and Bin Yang, "Camera-based Drowsiness Reference for Driver State Classification under Real Driving Conditions" 2010 IEEE Intelligent Vehicles Symposium University of California, San Diego, CA, USA June 21-24, 2010.
[3] Zhang, Wei; Cheng, Bo; Lin, Yingzi," Driver drowsiness recognition based on computer vision technology." Published in: Tsinghua Science and Technology (Volume: 17, Issue: 3) Page(s):354 - 362 Date of Publication: June 2012
[4] Eyosiyas Tadesse, Weihua Sheng, Meiqin Liu," Driver Drowsiness Detection through HMM based Dynamic Modeling." 2014 IEEE International Conference on Robotics & Automation (ICRA) Hong Kong Convention and Exhibition Center May 31 - June 7, 2014. Hong Kong, China.

## Author Profile

**Meshal Mansour Alnasheet** works as Specialist Trainers (A) in Higher Institute for Communication and Training, PAAET, Kuwait, in Telecommunication and Navigation Institute in computer department since 1999, one of public authorities and applied education and training institutes