

# Classification Algorithms in Sleep Detection—A Comparative Study

Aleksandra H. Pasiieczna, Jerzy J. Korczak

Wrocław University of Economics

ul. Komandorska 118/120, 53-345 Wrocław, Poland

Contact: www.pasiiecznapl@gmail.com (AHP), jerzy.korczak@ue.wroc.pl (JJK)

**Abstract**—This paper presents a comparison of different machine learning algorithms applied to automatic sleep detection which uses electroencephalogram signals as a differentiating basis. The Single-Layer Perceptron, Multi-Layer Perceptron, Support Vector Machine, Boosted Tree and the Multi-Agent (comprising of the earlier models) models are developed and analyzed with training and testing datasets. The results of the models are evaluated using a cross-validation technique. The models are compared with each other using the Cohen's index, the True Positive Rate and True Negative Rate. The models are very successful with sleep stage detection reaching up to 94 %, and Cohen's index reaching up to 0.69, showing considerable promise for deployment and future studies.

## I. INTRODUCTION

**R**OAD accidents are responsible for many deaths and economic losses around the globe. Drowsiness is a cause of about 40,000 non-fatal injuries and 1550 fatalities annually in the United States alone [1]. A press release by the National Sleep Foundation in 2009 indicates that 1.9 million American drivers have had a car crash or 'near-miss' due to fatigue or drowsiness [2]. The National Sleep Foundation has data which shows that even some flight and train accidents have drowsiness and fatigue as probable causes [3].

In general, to detect a driver's sleep stage the frequency spectra of brainwave signals are used as a differentiating factor. A good detection algorithm should accurately detect a drowsy driver, while not causing many false alarms. In principle, a model may detect drowsiness all the time, i.e., a success rate of 100 % for sleep detection, but this will create many false alarms, which would affect the driving behavior. A false alarm is triggered when a wake stage is detected as a sleep stage. For our classification purposes, a sleep stage is defined as the first stage of sleep. Earlier work using multichannel electroencephalogram (EEG) signals obtained 92.8 % internal sleep stage classification accuracy [4]. This work did not classify wake stages, which would be included in our models. The aim of our project is to obtain similar or better results for sleep detections using only one channel, i.e. fewer data input parameters. It is more important for us to detect a sleep stage than have fewer false alarms, which is why the sleep detection success rate has a high benchmark.

To train and test the algorithms before grading them, the dataset used was obtained from the DREAMS project [5]. This dataset consisted of raw polysomnographic (PSG) signals. A data-mining plan was designed for transforming the dataset,

for training and testing the models, and for evaluation of the models. This plan expanded on the existing widely-used CRISP-DM model [6].

In the DREAMS project, PSG data was collected from sleeping patients, which included the electroencephalograph signals, the backbone of our project. Automatic classification of the sleep stages and sleep disorders was a major part of the different studies performed by the DREAMS project team. They showed that it is possible to perform automatic classifications using machine learning algorithms.

In 2001, Van Hese P. *et al.* [7] performed an automatic detection of the sleep stages using only EEG data. They used a modified K-means algorithm, and the parameters they used for classification were the parameters of Hjorth – Activity, Mobility and Complexity – expressed in terms of the frequency spectrum. The mobility was a measure of the central frequency, the complexity was a measure of the bandwidth of the signal, and the activity was a measure of the variance. They showed that clusters were created and it was possible to distinguish between the different stages, but that extra information, like electrocardiograph (ECG) and electrooculograph (EOG) data, was necessary for a clear discrimination between the stages.

Zhovna I. and Shallom I. proposed in 2008, an automatic sleep stage detection and classification of multichannel EEG signals [4]. They used a method based on the Multichannel Auto Regressive (MAR) model, which incorporated the cross-correlation information existing between the different EEG signals. Their approach involved training and testing phases, including a continuous unknown 7-hour subject dataset. They obtained a promising rate of detection of 92.8 %. However, they had not trained the system to detect wake stages or movements.

A more recent study in 2013 by Malaekah E. and Cvetkovic D. tried to perform an automatic detection scheme to classify the sleep stage 1 and the wake stage using the EEG sub-epoch approach [8]. They divided 30 second epochs into 6 second sub-epochs from which the Relative Spectral Energy Band (RSEB) was calculated and used to create the feature space. The RSEB for a given frequency band is defined as the ratio of the Power Spectral Density (PSD) of the band divided to the total power (sum of PSD of all bands). Their best results showed an average of 77 % and 55.8 % success of detecting the wake and first sleep stages respectively.

The work done by different research groups shows that EEG signals are good indicators of sleep and wake stages. For manual classification, there are two references, [9] and [10], both of which rely on the frequency of the EEG signals. Following up on the works of [7] and [8], it seems that using the complete frequency spectrum instead of few parameters might improve the classification accuracy.

The main goal of the paper is to create an automatic detection mechanism using data mining approach. In the project, the standard CRISP DM methodology was applied [6]. The first step is to understand the important physiological processes of sleep and their mathematical representation as signals. The following section provides a brief description of the brainwave signals along with their classification based on the frequency range. Then the data preparation phase is explained followed by an introduction to the classification models used in this project. In the third section, the experimental results for the different models are tabulated in terms of the true positive and true negative rates, also known as sensibility and specificity respectively. The paper ends with the conclusions drawn from our work and perspectives to continue research in this domain.

## II. DATA SOURCES AND CLASSIFICATION MODELS

### A. EEG signals

An electroencephalograph (EEG) represents electrical signals which reflect the electrical activity of the neurons. Based on their frequency range, an EEG signal is divided into bands denoted by Greek letters [11]:

- Beta waves –  $\beta$ -waves are brainwaves with frequencies between 13 and 30 Hz. They indicate full awareness and high brain activity.
- Alpha waves –  $\alpha$ -waves have frequencies between 8 and 13 Hz. These are generally linked to relaxed states.
- Theta waves –  $\theta$ -waves lie in the frequency range of 4 to 8 Hz. These are connected to Non-Rapid Eye Movement (NREM) sleep.
- Delta waves –  $\delta$ -waves are below 4 Hz in the frequency spectrum. They generally correspond to slow wave sleep stages.

There are two standards to classify EEG data into sleep stages. The older Rechtschaffen and Kales (R&K) standard is composed of the following stages – Wake stage, REM stage, Sleep stage S1, Sleep stage S2, Sleep stage S3, and Sleep stage S4 [9]. In the more recent American Academy of Sleep Medicine (AASM) standard, the stages are classified as – Wake stage, REM stage, Sleep stage N1, Sleep stage N2, and Sleep stage N3 [10]. However, the DREAMS dataset contains additional stages. There is one sleep stage movement (transition) present in the R&K classification. The additional sleep stages are defined as *unknown sleep stages* by the classifying experts. The terminology used to denote the sleep stages is defined as:

- 0 = Wake stage or REM (R&K and AASM)
- 1 = Sleep stage 1 (R&K and AASM) or Sleep stage 2 (R&K and AASM)

- 2 = Sleep stage 3 (R&K and AASM) or Sleep stage 4 (R&K) or unknown sleep stage (R&K)
- 3 = Sleep stage movement (R&K) or an unknown sleep stage (AASM)

The reason why data points which might not seem as a wake stage or sleep stage S1/N1, are included is that when one model predicts a sleep stage 1, the other model might predict a sleep stage 3, an unknown sleep stage, or a sleep stage movement (transition).

### B. Data Preparation

The dataset used in the current work was collected as part of the DREAMS project in Belgium and consists of 32-channel polygraph whole-night polysomnographic readings of 20 healthy subjects [5]. At least three of the 32 channels were EEG channels. The data was collected with a sampling frequency of 200 Hz and stored in the standard European Data Format (EDF). The channel chosen by us for the project was the channel CZ-A1, a central lobe channel. Only the first sleep and wake stages were selected from the files along with the corresponding EEG data. Points, which did not have sleep stage 1 or the wake stage in one rating model but in the other, were included, and the total number of data points after this phase was 32924. Each data point consists of one hypnogram rating based on the R&K model, and one hypnogram rating based on the AASM model and 1000 raw EEG signal points corresponding to a manual rating of a 5-second time-window.

These data points were then read into the statistical software R for further processing. This is a very large volume of data ( $32924 \times 1000$ ) to be provided for a machine learning algorithm. To reduce the size, feature selection was performed. Each data point was first transformed as per the Fourier transform to obtain the frequency spectrum from 0 Hz to 100 Hz (half the sampling frequency) with an accuracy of 0.2 Hz (inverse of the length of the signal). The sleep manuals indicate that frequencies above 30 Hz do not provide information for sleep stages and thus frequencies above 30.5 Hz were rejected. The size of the data was still large ( $32924 \times 153$ ). The frequencies were then averaged to the nearest integer frequency to reduce the dimension of the problem. After the data selection, transformation, and feature selection, the size of the data was  $32924 \times 31$  (31, including the 0 Hz value). This final data was passed to the R platform, a statistical software package extensively used for analyzing large data.

### C. Quantitative Indicators Used for the Models

It is important to describe the quantitative indicators used to grade the models in this work. Two models for each machine learning algorithm were created, one corresponding to the AASM scoring method and the other to the R&K scoring method. Thus, for consistency purposes, it is vitally important to *not* compare the AASM models with the R&K models.

There are many indicators, which can be used to grade a classification model. In our paper, the confusion matrix (see Table I) is first created, where TN stands for *True Negatives*, TP for *True Positives*, FN for *False Negatives* and FP for

TABLE I  
EXAMPLE OF A CONFUSION MATRIX

		Model's Output	
		0	1
Manual Rating	0	TN	FP
	1	FN	TP

*False Positives.* These values can be used to grade the models, however they depend directly on the number of sleep and wake states. Since the database consists of unequal number of sleep and wake states, and to thus properly normalize the results, the following indicators, along with the Cohen's Index, were used:

- *True Positive Rate* (TPR) – This value is known as the sensitivity of the model, and provides an estimate for the successful sleep detection rate by the model. It is defined,

$$\text{TPR} = \frac{\text{TP}}{(\text{TP} + \text{FN})}, \quad (1)$$

and is also linked to the *False Negative Rate* (FNR) through  $\text{TPR} = 1 - \text{FNR}$  [12].

- *True Negative Rate* (TNR) – This value is known as the specificity of the model, and provides an estimate for the successful wake detection rate by the model. It is defined as,

$$\text{TNR} = \frac{\text{TN}}{(\text{TN} + \text{FP})}, \quad (2)$$

and is also linked to the *False Positive Rate* (FPR) through  $\text{TNR} = 1 - \text{FPR}$  [12].

- *Cohen's Index* ( $\kappa$ ) –  $\kappa$  is an indicator that measures the inter-rater agreement for categorical objects, and that takes into account the agreements (TP and TN) occurring by chance [13]. It is defined as:

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \quad (3)$$

where  $p_o$  is the observed agreement between the raters, given as:

$$p_o = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (4)$$

and  $p_e$  is probability of chance agreement, described as:

$$p_e = \frac{(\text{TP} + \text{FN})(\text{TP} + \text{FP})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})^2} + \frac{(\text{TN} + \text{FN})(\text{TN} + \text{FP})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})^2}. \quad (5)$$

It can be easily verified that  $\kappa = 1$  when the two raters are in complete agreement. In general, a value of  $\kappa > 0.6$  indicates a good level of agreement between the raters.

In classification models, there is a trade-off between the TPR and TNR, since increasing one generally reduces the other. Thus it is necessary to study both simultaneously, making the TPR and TNR important indicators of the models. These are simple indicators and to consider random agreement,  $\kappa$  was also used to grade the models.

The following subsections will present the classifying algorithms used in our experiments – Single-Layer Perceptron, Multi-Layer Perceptron, Support Vector Machine, XGBoost and Multi-Agent model – whose classification accuracies were compared, in order to choose the best model. The SLP is a simple linear classifier that calculates the sum of the weighted input variables. It is simple to implement and forms a good reference model. The MLP, also known as neural networks, consists of layers of neurons. This provides the next level of complexity and allows for non-linear behavior to be captured. The SVM models have better predictive performance than the MLP, since they create multiple support vectors, which provide more flexibility in the classifying criterion. The XGBoost model is a variation of the gradient boosting model, an ensemble model that is a collection of decision trees added to minimize a loss function. The Multi-Agent classifiers are two algorithms proposed by us that consists of all the previous models to better classify the EEG data, one by a majority vote and the other by treating the sleep and wake states on a more equal footing.

#### D. Single-Layer Perceptron

E. Malaekah and D. Cvetkovic [8] had used the ratios of the EEG bands to perform classification of the sleep and wake states. The classifying criterion was manually tuned in their case. Instead of using the ratios of the EEG bands and manually tuning the ratio to match a specific quantity, we used an automatic learning algorithm that can find the classifying criterion without any human interference.

A Single-Layer Perceptron (SLP) is an algorithm for binary classification of an input vector. It is a linear classifier that makes its decisions by a scalar product of the input vector with a set of weights. In machine learning, a perceptron is considered to be the simplest neural network consisting of one neuron. Mathematically, the SLP algorithm can be expressed as:

$$y(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}. \quad (6)$$

Here,  $\vec{x}$  is the input vector,  $\vec{w}$  is the weight vector and  $b$  is the bias. The output of the perceptron is either 0 or 1 and is determined by the condition provided by the scalar product. The bias alters the position of the decision line. The weight vector is changed by learning algorithms. The learning algorithm used here was the Particle Swarm Optimization (PSO) [14]. PSO optimizes a problem by having a set of candidate solutions (called particles), and moving these particles in the feature space by simple rules for updating the particles' positions and velocities. This optimization technique does not make assumptions about the parameters being optimized and so can be used for noisy and irregular problems. PSO possesses certain advantages of Monte Carlo techniques, where a large space of solutions is searched by the various particles, as well as the advantages of classical optimization techniques, such as gradient descent, where the particles are guided by a 'force' towards the final solution.

Each particle is attracted by its best known position, as well as the best position among all the particles. There is also an inertia term, which keeps the particle moving along a straight line. These three ‘forces’ act together in different proportions to guide the particle towards better solutions. A PSO algorithm consists of the following main steps, along with part of the code from our implementation in R:

- Initialize  $N$  particles’ positions `particle_pos` and velocities `particle_vel` randomly within the boundaries of the feature space. `part` indicates the particle index, maximum  $N$ , and  $i$  the dimension, 31 dimensions in our case.

```
particle_pos[part,i] <- runif(1, min=-10,
max=10) #Random starting positions
particle_vel[part,i] <- runif(1, min=-5,
max=5) #Random starting velocities
```

- Update each particle’s best position `best_particle_pos` with the randomly assigned starting position.

```
best_particle_pos[part,i] <-
particle_pos[part,i]
```

- Find the particle with the best position and update the global best position, `coeffs_aasm` or `coeffs_rnk`, with its position. This step depends on the error function and with random starting positions. In our case, the error function was the sum of absolute errors between the expected result and the actual result. This is similar to using the mean of squared errors, since only the absolute value of the error is considered. `#perr` is the particle error.

```
if(perr[part] < global_min_err) {
global_min_err <- perr[part]
coeffs_aasm <- particle_position[part,]
}
```

- Until the stopping criterion is satisfied, i.e. number of iterations run or error criterion is below threshold, do the following

- Update each particle’s position by adding the velocity to it.

```
particle_position[part,] <-
particle_position[part,] +
particle_velocity[part,]
```

- Update each particle’s best position with its current position if and only if the current position is better than the particle’s best position. `#min_perr` is the particle’s minimum error.

```
if (perr[part] < min_perr[part]) {
min_perr[part] <- perr[part]
best_particle_position[part,] <-
particle_position[part,]
}
```

- Find the particle with the best current position and update the global best position if and only if this best current position is better than the global best position.

```
if(perr[part] < global_min_err) {
global_min_err <- perr[part]
coeffs_aasm <- particle_position[part,]
}
```

- Update each particle’s velocity as a sum of three contributions multiplied by factors indicating their proportions. The first contribution is the difference between the particle’s best position and its current position. The second contribution is the difference between the global best position and the particle’s current position. The final contribution is simply the previous velocity of the particle.

```
particle_velocity[part,] <-
particle_velocity[part,] +
runif(31,min=0,max=0.4) *
(best_particle_position[part,]
- particle_position[part,]) +
runif(31,min=0,max=0.7) * (coeffs_aasm
- particle_position[part,])
```

- Update the number of iterations or calculate the error.

- The global best position is the best solution.

The factors which multiply the three contributions, the inertia term, the force due to the global best position and the force due to the particle’s best position, are chosen by the user. These control the ‘swarming rate’ and hence the behavior and response of the system. In our implementation, the inertia term was multiplied by 1, the force due to the particle’s best position was multiplied by a random number between 0 and 0.4 (excluding the extreme values), and the force due to the global best position was multiplied by a random number between 0 and 0.7. The inertia term was kept as a unit value because we wanted the particles to move to not stop swarming. The maximum values of 0.4 and 0.7 were chosen so that the particles were attracted towards their individual best positions and the global best positions, but the contribution of these attractions did not exceed the contribution of the inertia term. The higher average value of the attractive force towards the global best position indicated a preference of each particle to explore positions near the global best position, which is the best known solution so far. The non-zero value of the force towards the individual particle’s best position ensured that the particles explored positions near their own best positions. The final values were chosen based on the “goodness” of the results.

### E. Multi-Layer Perceptron

In our case, the data does not satisfy the condition of linear separability; therefore non-linear classifiers were considered.

A well-known neural network model, a Multi-Layer Perceptron (MLP) was chosen and its architecture was evaluated.

Neural networks contain at least two layers, with the possibility of additional *hidden* layers. Earlier experiments [15] were performed with two and three hidden layers, the latter of which gave maximum sleep and wake detection rates of about 69 % and 90 % respectively. Compared to the architecture with three hidden layers, this architecture drastically decreased the number of connections and thus the learning time for the neurons [15]. Based on these results, a neural network with two hidden layers 31-46-10-1 was chosen. The learning algorithm used for this project was the default algorithm, *resilient backpropagation* with weight back tracking. The error function minimized in this project was the mean squared errors.

Neural networks have one major disadvantage – risk of *overfitting*, when a model begins to describe the noise. The underlying relationship between the input variables and the output value may fail to have a predictive power completely. In our case 1,953 weights had to be found using 16,462 training points. Thus there is a high possibility of overfitting in our neural network models, which can lead to poor predictions. Unfortunately, reducing the number of neurons in the hidden layers led to a drastic decrease in the performance of the neural network, while increasing it led to overfitting, where the testing phase results were measurably poor as compared to the training phase results. Therefore, the data parameters for our neural network could not be classified correctly without increasing the risk of overfitting. Due to a lower number of sleep stages, the neural network models did better at detecting the wake stages.

#### F. Support Vector Machine

Support Vector Machines (SVMs) are supervised learning algorithms used for classification and regression analysis [16], [17]. An SVM model maps data points in the feature space as categories which are as far apart as possible, by constructing a hyperplane or a set of hyperplanes in a space, generally of a higher dimension than the feature space. Generally, two or more hyperplanes are selected with no points between them, and then the distances between these hyperplanes are maximized.

Two types of SVM models, a simple SVM model and a multi-class SVM model, were used. The first one interprets the hypnogram sleep stage rating as having a “meaning”, i.e. this value can be expressed as the output of a mathematical function of the SVM. The second model merely assigns the hypnogram value as a label for a given SVM input vector. This model is hence more robust to clustering of data that is not linearly separable.

Before starting the training of the SVMs, tuning was performed for two parameters,  $\gamma$  and  $cost$ , using the functionality provided in the package `e1071` through the function `tune.svm()` on 10 % of the data. The parameter  $\gamma$  is internal to the implementation and is required for all the non-linear kernels, while  $cost$  is the cost of violating

the constraints of the SVM model. The results of this tuning indicated a value of 0.1 for  $\gamma$ , and a value of 1 for  $cost$ , which were recommended by the SVM implementation based on the 10 % of the data. The same values were obtained for both AASM and R&K classification methods, and were used for the final training of the models. In the multi-class variant of SVM (unlike in the default-SVM), the wake states were given a weight of 0.2 and the sleep states were given a weight of 1.0, to compensate the influence of the higher number of wake states in the dataset.

#### G. Extreme Gradient Boosted Model

Extreme Gradient Boosted Model (XGBoost) is a supervised machine learning algorithm used for classification and regression analysis, which comprises of an ensemble of ‘weak’ prediction models, generally in the form of decision trees. It is an efficient implementation of the gradient boosting framework, and has been known to be many times faster than standard implementations.

The boosting process deals with transforming the weak prediction models into a strong model. For example, decision trees with only one or two levels would be considered weak, since their accuracies are slightly better than random classification, and thus are more robust to overfitting. A boosted model creates an ensemble of these models with the intention of creating a strong learner. In many implementations, the weak learners are functions of the loss (error) functions, which the ensemble is trying to minimize. The gradient boosting model consists of calculating the loss function at each iteration and correcting itself by adding the gradient of the loss function in the next iteration [18], [19].

In this work, the maximum depth (nodes) was set to 5 to allow better decision trees at each level. This reduced the number of iterations, i.e. the number of additional trees required to correct the loss function, which was set to 40. The learning process was made more conservative (less overfitting) by scaling the contribution of each tree by 0.15, compared to the default value of 0.3. These values were selected after running multiple tests on the data, and were used for the AASM and R&K models.

#### H. Multi-Agent Model

To maximize the sleep and wake detection rates, two Multi-Agent (MA) models were developed. Typically, an MA model consists of many ‘agents’, each of which is an independent classification model. In our work, these MA models were constructed from all the aforementioned algorithms [20], and for this project, a democratic MA model and a weighted MA model were chosen. The democratic MA model determined the outcome of a given input by a majority vote, while the weighted MA model determined the outcome of a given input by weighing the outputs of each of the agents.

The democratic MA model did not have any training or testing phase, due to it directly relying on its component models. The outputs from the agents were directly used to

analyze this model on the training and testing data sets. Two models were made, one for the AASM classification and one for the R&K classification.

The training phase for the weighted MA model consisted of optimizing the weights using an error function, and the optimization was achieved with the Particle Swarm Optimization (PSO) as discussed in the SLP model. The error function chosen was the sum of the FNR and FPR. Since the rates are independent of the number of states, this error function is not biased towards one of the sleep or wake states. Minimizing this function would thus maximize the sleep and wake detections simultaneously ( $TPR = 1 - FNR$  and  $TNR = 1 - FPR$ ).

The parameters of the PSO algorithm were identical to those used to determine the weights of the SLP, with the exceptions that the number of dimensions were reduced to 5 and the bias was not required. Two such models were made for each training set, one corresponding to the AASM classification and the other corresponding to the R&K classification.

### III. EXPERIMENTAL RESULTS

The data was divided randomly into training and testing sets for cross-validation purposes to check how well the model can perform under more general data. We performed 4-fold cross-validation, where the data was divided into four equal random sets. Earlier experiments [15] indicated similar results to 2-fold cross-validation results, and 4-fold cross-validation has more training and testing phases, which is why it was used to obtain more results. The final results (averaged) are presented for comparison purposes. To compare the models, all learning algorithms were provided the same data for training and testing.

Since there were two scoring models (R&K and AASM) for each data point, two models were made for each learning algorithm, e.g. MLP\_Rnk and MLP\_AASM. Because of the two *different* standards, it only makes sense to compare all the AASM models together and all the R&K models together. After splitting the data and dividing them as *train-sets* and *test-sets*, the data workspace in R is stored for future use and reference.

In the confusion matrix tables, two outlier points (Outlier 1 and 2) are obtained because of having the same data point classified as sleep stages other than 0 or 1 in one of the two models. These points were neglected to better analyze our results. The confusion matrix was reinterpreted for comparison purposes in terms of TPR, TNR and  $\kappa$ . These quantities are provided for the 4-fold cross-validation studies in table II.

A general observation is that the multi-class SVM (R&K) performs the best, followed closely by the multi-class SVM (AASM). In addition, the TPR for the multi-class SVM is about 94 %, better than previous results of 92.8 % classification accuracy obtained by I. Zhovna and I. Shallom [4], where the cross-correlation information existing between the multichannel EEG signals was used. The TNR, on the other hand, is about 75 %, slightly lower than the 77 % obtained by E. Malaekah and D. Cvetkovic [8]. This is not so far behind, and thus the multi-class SVM is a good choice for

further study. As perspectives, the multi-class SVM might be improved by providing more data points corresponding to sleep stages so that the FPR is reduced.

Looking at the Cohen's indices for the different algorithms, we see that the MA, XGBoost, MLP and SVM models are at least 0.60. However, in our study, we place more emphasis on sleep detection, i.e. a TPR higher than 90 %. This is only achieved by the multi-class SVM. It should be pointed that the number of sleep states are lower than those of wake states, which is why the XGBoost and the MA models have a TNR greater than the TPR.

From the tables, we infer that the default SVM does better than the SLP and MLP methods. Both default SVM models (AASM and R&K) have a TPR of over 70 %, similar to the TNR of the multi-class SVM, and the same can be said for the TNR of the default SVM models. Thus, when compared with the results of the multi-class SVM, the TPR and TNR values appear to be swapped. This could be due to the larger number of the wake stages.

A value of  $\kappa > 0.60$  indicates a good agreement between the different raters, and thus the SVM, XGBoost and MA models performed well. Comparison of the Cohen's indices shows that the default SVM, XGBoost and MA models perform well with  $\kappa > 0.65$  for both scoring methods.

To better understand the trade-off between TPR and TNR, the Receiver-Operator Characteristic (ROC) curves for all the models were plotted as shown in Fig. 1. The FPR and TPR form the  $x$ - and  $y$ -axes of the graph respectively, and a random guessing model would have  $TPR = FPR$  as shown with the blue dotted-dashed diagonal in the figure. Points above the diagonal have  $TPR > FPR$ , indicating a better predictive performance and a 'perfect' classification model would be at the (0,1) point on the plot, i.e.  $FPR = 0$  and  $TPR = 1$ .

Based on the ROC curve, we see that the weighted MA (PSO) model does the best in terms of the trade-off, being closest to the (0,1) point, followed closely by the democratic variant. This is due to the fact that the error function in the PSO algorithm treats sleep and wake stages on an equal footing. In the democratic MA model, there is no training beyond that for each of the individual agents. As a result, the bias towards the wake states exists due to the 'majority' of the models (MLP, XGB and SVM) being biased towards the wake states, a consequence of the previously mentioned larger number of wake states.

Depending on the requirement, detection of one stage might be more critical and the 'best' model has to be accordingly chosen. The purpose of this project was to focus on sleep detection over wake detection and thus, based on the Cohen's index and the requirement of  $TPR > 90$  %, the multi-class SVM is a better model.

It is important to underline that the R&K models have performed better than the AASM models. The R&K standard for sleep scoring was developed in 1968 [9], while the AASM standard was developed recently, in 2007 [10]. These standards have differences which might have affected the manual scoring of the dataset used. To choose between them, inputs from

TABLE II  
COMPARISON OF RESULTS (4-FOLD CROSS-VALIDATION)

	AASM Criteria			R&K Criteria		
	TPR (%)	TNR (%)	Cohen's Index ( $\kappa$ )	TPR (%)	TNR (%)	Cohen's Index ( $\kappa$ )
SLP	72.3	76.9	0.47	73.0	77.1	0.46
MLP	69.0	88.2	0.58	70.5	89.3	0.61
Default-SVM	73.8	90.5	0.65	73.9	92.8	0.68
Multi-class SVM	94.0	67.0	0.53	94.0	74.9	0.60
XGBoost	74.5	90.6	0.66	76.2	92.2	0.69
Multi-Agent (Democratic)	77.2	87.8	0.65	79.1	90.1	0.69
Multi-Agent (PSO)	82.0	85.7	0.66	83.3	88.3	0.69

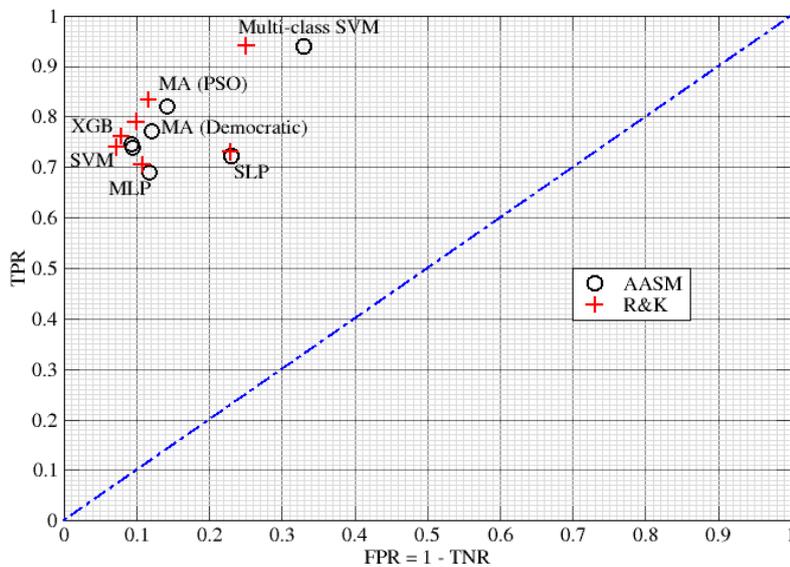


Fig. 1. ROC curve for AASM (black circles) and R&K (red crosses) classification models. The blue dotted-dashed line indicates the ROC curve for a random guess model.

a neuroscientist or more *in vivo* data based on a standard of quantifying drowsiness are required. In addition, using a more *balanced* dataset with similar numbers of wake and sleep stages might help develop the methods further.

#### IV. CONCLUSIONS AND PERSPECTIVES

The main objective of this work was to develop an automatic sleep detection system to warn a sleepy driver and prevent road accidents. With this purpose, the DREAMS database [5] was used and the Fourier transforms of the EEG signal formed the input vector space for the machine learning algorithms. Multiple experiments were performed on different models, such as SLP, MLP, SVM, multi-class SVM and XGBoost. A MA model was subsequently developed that treated the sleep and wake stages equally. This model was observed to be the closest to a perfect classification as can be confirmed

with the ROC curve. In addition, the multi-class SVM model for the R&K scoring system maintains the benchmarks of  $\text{TPR} \geq 90\%$  and  $\text{TNR} \geq 70\%$ , and the AASM scoring system is not far behind. Using these benchmarks, the potential reduction in economic costs due to loss of human lives was calculated for France to be about 1,104 million EUR for the year 2014 [15].

For future work, the detection models will need to be tested across different datasets to confirm their reliability and to ensure that the models do not fail when presented with new data, either through future laboratory studies or *in vivo* experiments. Pragmatically, the models can be said to be relatively compatible with different datasets, a desirable property, which is useful in the event of retraining the models for certain medical cases where the EEG signals might not be the same as those in an average human being.

Furthermore, it might be fruitful to validate our results from a strictly medical perspective, with the ability to quantify *drowsiness* rather than the first sleep stages, preferably using EEG signals. Obtaining a more *balanced* dataset with equal wake and sleep stages would ensure that there is no bias of the models towards a particular stage. Including more physiological data, like body pulse or blood pressure, may improve the model, since they also decrease when a person tends to fall asleep.

#### REFERENCES

- [1] National Highway Traffic Safety Administration. Drowsy Driving and Automobile Crashes. [http://www.nhtsa.gov/people/injury/drowsy\\_driving1/Drowsy.html](http://www.nhtsa.gov/people/injury/drowsy_driving1/Drowsy.html) (Accessed 08 May, 2014).
- [2] 1.9 Million Drivers Have Fatigue-Related Car Crashes or Near Misses Each Year. National Sleep Foundation (2009). <http://www.sleepfoundation.org/media-center/press-release/19-million-drivers-have-fatigue-related-car-crashes-or-near-misses-each> (Accessed 08 May, 2014).
- [3] Crashes Where Fatigue Was a Contributing Factor. National Sleep Foundation (2012). <http://sleepfoundation.org/sites/default/files/Crashes%20Fatigue%20a%20Factor.pdf> (Accessed 20 May, 2015).
- [4] Zhovna I. and Shallom I. D.: Automatic detection and classification of sleep stages by multichannel EEG signal modelling. *Engineering in Medicine and Biology Society*, 2008. 30th Annual International Conference of the IEEE, 2665-2668 (2008).
- [5] Devuyt S., Dutoit T., Kerkhofs M.: DREAMS Project, The DREAMS Sleep Subjects Database. <http://www.tcts.fpms.ac.be/~devuyt/Databases/DatabaseSubjects/> (Accessed 16 April, 2014).
- [6] KDnuggets Polls, *Data Mining Methodology* (2007). [http://www.kdnuggets.com/polls/2007/data\\_mining\\_methodology.htm](http://www.kdnuggets.com/polls/2007/data_mining_methodology.htm) (Accessed 20 May, 2015).
- [7] Van Hese P., Philips W., De Koninck J., Van de Walle R., Lemahieu I.: Automatic detection of sleep stages using the EEG. *Engineering in Medicine and Biology Society. Proceedings of the 23rd Annual International Conference of the IEEE*, 2, 1944-1947 (2001).
- [8] Malaekah E. and Cvetkovic D.: Automatic detection of the wake and stage 1 sleep stages using the EEG sub-epoch approach. *Engineering in Medicine and Biology Society (EMBC) 2013, 35th Annual International Conference of the IEEE*, 6401-6404 (2013).
- [9] Rechtschaffen A., Kales A.: *A Manual of Standardized Terminology, Techniques, and Scoring System for Sleep Stages of Human Subjects*. US Department of Health, Education, and Welfare Public Health Service (1968).
- [10] Iber C., Ancoli-Israel S., Chesson Jr. A. L., Quan S. F.: *The AASM Manual for the Scoring of Sleep and Associated Events*. American Academy of Sleep Medicine (2007).
- [11] Sucholeiki R.: *Normal EEG Waveforms* (2014). <http://emedicine.medscape.com/article/1139332-overview#aw2aab6b3> (Accessed 14 April, 2014).
- [12] Lalkhen A. G. and McCluskey A.: Clinical tests: sensitivity and specificity. *Continuing Education in Anaesthesia, Critical Care & Pain*, 8, 221-223 (2008).
- [13] Cohen J.: A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 37-46 (1960).
- [14] Kennedy J. and Eberhart R.: Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 4, 1942 (1995).
- [15] Pasiczna A. H.: *An Approach to Driver Sleep Detection*. Master Thesis Report, Wrocław University of Economics (2015).
- [16] Boser B. E., Guyon I. M., Vapnik V.: A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, 144-152 (1992).
- [17] Cortes C., Vapnik V.: Support-Vector Networks. *Machine Learning* 20, 273-297 (1995).
- [18] Mason L., Baxter J., Bartlett P. L., Frean M. R.: Boosting Algorithms as Gradient Descent. *Advances in Neural Information Processing Systems* 12, 512-518, *MIT Press* (2000).
- [19] Friedman J. H.: *Greedy Function Approximation: A Gradient Boosting Model*, IMS 1999 Reitz Lecture (1999).
- [20] Dietterich T. G.: *Ensemble Methods in Machine Learning*. *Lecture Notes in Computer Science* 1857, 1-15 (2000).