# Predicting the eye state of a subject from EEG data using supervised classification algorithms

**Arnaud Le Doeuff and Ignacio Dorado**


## Abstract

In this paper, we investigate how the eye state (open or closed) can be predicted by measuring brain waves with an EEG. We used a corpus containing the activation strength of the fourteen electrodes of a commercial EEG headset as well as the manually annotated eye state corresponding to the recorded data. The classifiers that we have selected for the classification tasks were: Support Vector Machine, Random Forest, and K-nearest Neighbors. We have implemented the classification process using the scikit-learn. We have learned and evaluate the classifiers using K-fold Cross Validation and Walk-Forward Validation to show the importance of temporal ordering.


## 1 Description of the problem

The task we have to solve is the classification of the ”EEG Eye State Data Set”, available from the UCI machine learning repository [1].

All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. The eye state was detected via a camera during the EEG measurement and added later manually to the file after analysing the video frames. ’1’ indicates the eye-closed and ’0’ the eye-open state. All values are in chronological order with the first measured value at the top of the data.

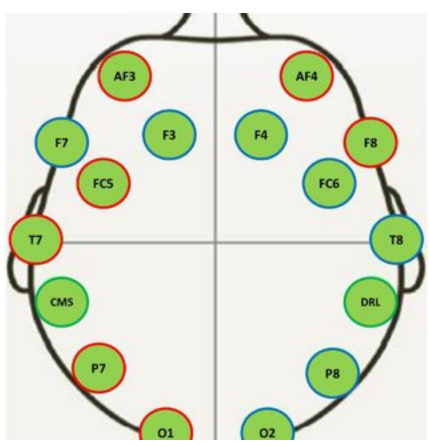


Figure 1: Overview of the sensor position and the corresponding behavior group. Blue corresponds to a maximum increase and red to a minimum decrease when opening eyes.

The database has the following characteristics:

- 15 attributes.

- 2 classes:'0 ' indicates the eye-closed, '1 ' indicates the eye-closed'.
- 14980 instances.
- The data is almost perfectly balanced. The number of instances in each class are: 8257, 6723.

# 2 Description of our approach

## 2.1 Preprocessing

We plot the data using matplotlib and the plot shows some clears outliers that prevent us from visualizing properly the data. We remove all values that are displaced from the mean 4 times the standard deviation or more. After this preprocessing we get a new dataset which can be ploted and visualized easily.

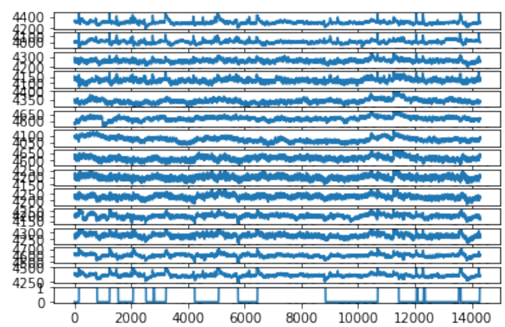


Figure 2: Plot each feature in a different graph, without outliers.

## 2.2 Non temporal approach

We divide the dataset into training set and testing set. 70 percent of the data is used for the training set and the other 30 percent for the testing set. The dataset is shuffled before the split in order for distribution to be random.

We defined our own implementation of K-fold cross validation in order to show the details of the algorithm. Cross validation is performed over only over the training set using 10 different folds. Since not all the folds have the same amount of data, calculating the mean of all individual accuracies would give the same weight to a set with fewer predictions than to another one with more of them and would result in a small error. To solve this problem we weight each of the individual accuracies according to its amount of predictions.

We train and evaluate the 3 classifiers (Support Vector Machine, Random Forest, and K-nearest Neighbors) using our cross validation function.

This approach throws great results, but it is methodologicaly wrong. When shuffling the dataset we are not respecting the order in which measurements were taken and we are using information from the future to predict previous examples. We can not keep working using this approach.

## 2.3 Temporal approach (using K-fold Cross Validation)

This time we split the dataset without shuffling it. We respect the chronological order and use the first 70 percent of measurements for training and the rest for testing.

We train and evaluate the 3 classifiers (Support Vector Machine, Random Forest, and K-nearest Neighbors) using our cross validation function.

Accuracies obtained are significantly lower. This results are really poor and those classifiers are not able to properly predict any data. But we can do better

### 2.4 Temporal approach (using Walk-forward validation)

We can allow the model to see previous information from the past (but never from the future). by using Walk-forward validation which evaluate a new model for each new example.

We define a function to train and evaluate models using Walk-forward validation. It also receive as a parameter the window of examples from the past that are available in each training step. KNN classifier (which has previously proven to be the best model for this dataset) was trained and evaluated, obtaining an accuracy score of 1.

## 3 Other results

Feature selection was also performed in order to reduce execution time (since accuracy can not be better than 1). Using K-Best algorithm for a number of 1 feature the training and evaluation of KNN classifier using walk-forward validation output an accuracy of 1. This may indicate that 1 variable is important enough to let us predict every example using Walk-forward validation. But it probably indicate that something went wrong during the process.

TPot was also tested, but it did not do well, since the temporal approach is really hard for any classifier that does not use Walk-forward validation.

## 4 Conclusions

In our project, the model evaluation methodology must take the temporal ordering of observations into account.

This means that it is methodologically invalid to use k-fold cross-validation that does not stratify by time (e.g. shuffles or uses a random selection of rows).

This also means that it is methodologically invalid to use a train/test split that shuffles the data prior to splitting.

We saw this in the evaluation of the high skill of the model with k-fold cross-validation and shuffled train/test split compared to the low skill of the model when directly adjacent observations in time were not available at prediction time.

## References

[1] Oliver Roesler. UCI repository of machine learning databases [https://archive.ics.uci.edu/ml/datasets/eeg+eye+state].baden-wuerttemberg cooperative state university (dhbw), stuttgart, germany. *Department of Information and Computer Science*, 55, 1998.