Machine Learning model for EEG eye state classification

S. Parvizi

Abstract

What is presented in this work is a Machine Learning (ML) model that, given an EEG signal, can predict eye states (i.e., eyes opened or closed) in a time window from 0.1 to 1 second with over 90% accuracy. The findings in this study will contribute to an ideal solution to the problem of EEG based eye state classification.

1 Introduction

If we consider just epilepsy diagnoses, approximately 30% of the patients EEG is misread and consequently misdiagnosed because of human mistakes [1, 2, 3]. This percentage is certainly higher if not only epilepsy, but every EEG diagnosable diseases are considered. Luckily, AI architectures applied to our daily life are rapidly increasing, in fact EEG eye state classification has been successfully applied in the areas of infant sleep-waking state identification [4], driving drowsiness detection [5], epileptic seizure detection [6], bipolar mood disorder classification (BMD) [7], and ADHD patients (Attention Deficit Hyperactivity Disorder) [7].

The fast and powerful methods that we rely on in ML, such as *train-test splits* and *k-fold cross validation*, do not work in the case of time series data. This is because they ignore the temporal components inherent in the problem. The work [8] from which this project takes inspiration, may present the same methodological flaw in how this time series is evaluated. KNN will seek out the *k* most similar rows in the data set and calculate the output state as the prediction. By not respecting the temporal order of instances when evaluating models, it allows them to use "future" information to make the prediction. This concern specifically the KNN algorithm.

Use of *walking forward validation*, which is proven to be the most effective validation methodology in time series classifiers [9], is what characterizes this work.

2 Method

2.1 EEG Signal

All information used in this work is from one persistent EEG recording[1] collected with a 14 channels headset[2] and each record was labelled manually. The eye state was recognized by camera during the EEG continuous measurement and added later physically to the record in the wake of examining the feature outlines; "1" means an eye-shut and "0" the eye-open state. The record has a duration of 117 seconds with a sampling rate of 128 Hz which imposes a minimum threshold on the recognition rate of algorithm. Consequently there's a total of 14976 measurements (128*117) and 14 features in the data-set, where each of them is the information acquired by one single sensor. The fifteenth channel represents the output.

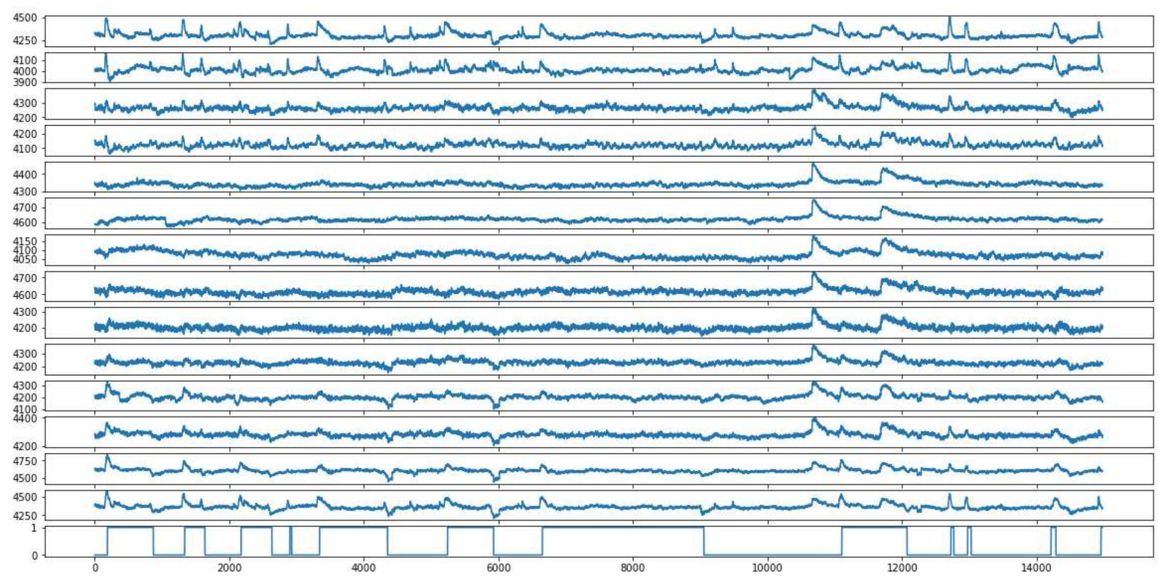

FIGURE 1: Visualization of whole data set.

[1]The database corpus was collected from UCI Machine Learning Repository
[2]Emotiv EPOC Headset

2.2 Model

For this work it was chosen the random forests algorithm, developed nearly 15 years ago [10], which creates many trees on the subset of the data and combines the output of all the trees reducing overfitting problem in decision trees and the variance, improving therefore the accuracy. It is known, in fact, to be one of the best ready-to-go algorithms available for classification, having incredible success across a variety of learning disciplines. For instance, it have fared well in machine learning competitions involving biology such as Ishwaran and colleagues work [11]. Moreover, random forests work poorly any time that one variable is much more important than all the others, which is not the case of this work.

2.3 Data reshape

As said before, temporal components must be considered when analysing time-series, and to do that data can be reshaped so that the value at the previous time-step is used to predict the value at the next time-step. The `sliding_window` function shown in Fig.2, do that using as paramaters the dataset, the number of data (`array_lenght`) corresponding to the specific time-step (e.g. if 1sec is equal to 128 data, 0.5s is equal to 64, etc.), and the total amount of data in the dataframe. After that, it will `model.fit` and `model.predict` a training set and a validation set of lenght equal to `array_lenght`, adding the accuracy value to the list `f1_tot`. This process is looped until it covers the whole dataframe. At the end, the function will return the `f1_tot` mean.

```
def sliding_window(dtset,array_lenght,num_dati_nel_dtframe):
  model = RandomForestClassifier(n_estimators=45)
  array_start = 0
  f1_tot = []
  for x in range(array_lenght,num_dati_nel_dtframe,array_lenght):
    xtrn_no_output = dtset.drop(['output'],axis=1)
    xtrn = xtrn_no_output[(array_start):(x)]
    ytrn = dtset.iloc[(array_start):(x),-1]
    xval = xtrn_no_output[(x):((x)+array_lenght)]
    yval = dtset.iloc[(x):((x)+array_lenght),-1]
    model.fit(xtrn,ytrn)
    prev = model.predict(xval)
    f1_value = f1_score(yval,prev,zero_division=1)
    f1_tot.append(f1_value)
    array_start = x
  accuracy = np.mean(f1_tot)
  return accuracy
```

FIGURE 2: Sliding window function

2.3.1 Walking Forward validation

Since non-time-series techniques randomly sample the datasets to create test and training sets, they do not preserve such differences which exist in practice. As said by Falessi et al. "*Thus, walk-forward is not only more accurate than 10-fold cross-validation and bootstrap but is also the only one that respects the property of the data, i.e., it preserves the order of data. [...] Therefore, the accuracy measured by non-time-series techniques is poorly realistic of any classifiers' practical adoptions*" [9].

In walk-forward, the dataset is divided into parts which are than chronologically ordered and, in each run, all data available before the part to predict is used as the training set, and the part to predict is used as test-set. Afterwards, the model accuracy is computed as the average among runs. For instance, Fig.3 illustrates the walk-forward technique; the parts used for training are in orange, the ones used for testing are in gray, the ones not used are in white. Fig.3 describes a dataset related to a hypothetical project of five parts, and four runs. In the first run, the first part is used as training, and the second as testing, in the second run the first two parts are used as training and the second as testing, and so on. The accuracy is averaged among the four runs.

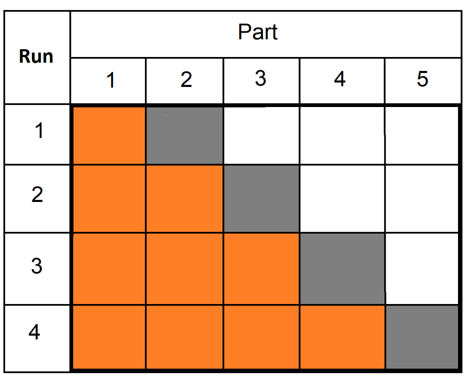

FIGURE 3: Example of walking forward

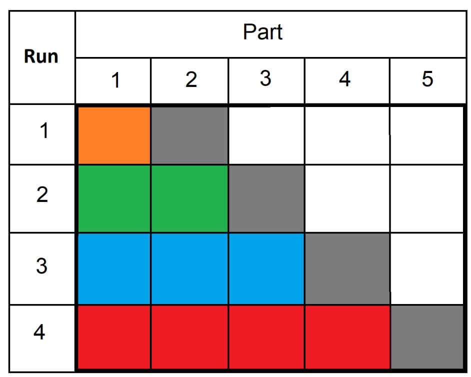

FIGURE 4: Corresponding walking forward

The dataset of this work was also splitted into 5 parts like the example (Fig.3). The `walking_forward_5` function includes and loops the `sliding_window` function four times.

```
def walking_forward_5(dtset,datas_in_onefifth):
    Y_datasVAL = []
    Y_datasTEST = []
    for times in range(1,5,1):
        TrnVal_set = dtset[:(times*datas_in_onefifth)]
        Test_set = dtset[(times*datas_in_onefifth):((times+1)*datas_in_onefifth)]
        for time in range(12,132,12):
            accuracy_per_tenth_s = sliding_window(TrnVal_set,time,(times*datas_in_onefifth))
            Y_datasVAL.append(accuracy_per_tenth_s)
        for time in range(12,132,12):
            accuracy_per_tenth_s = sliding_window(Test_set,time,datas_in_onefifth)
            Y_datasTEST.append(accuracy_per_tenth_s)
    accuracy_T = np.mean(Y_datasTEST)
    accuracy_V = np.mean(Y_datasVAL)
    return accuracy_V, accuracy_T
```

FIGURE 5: walking_forward_5 code

3 Conclusion and Discussion

3.1 Results

The results obtained are showed in Fig.6. As said before, the dataset of this work was splitted into 5 parts and a comparison can be made with the w-f example (Fig.4). Moreover, the overall accuracy of the model was measured averaging data from `sliding_window` function from 0.1s to 1s for all four runs, with a result of 90.55%

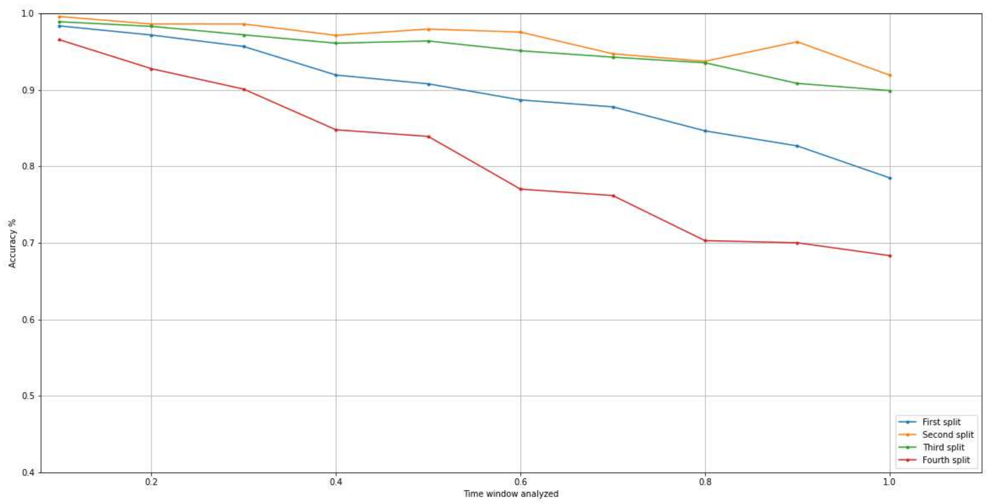

FIGURE 6: Walking forward results at different time windows analyzed

Nevertheless, a further analysis was made to understand where the predictions were inaccurate: Fig.7 showed that these match precisely the moments of transition from eyes opened to eyes closed and viceversa.

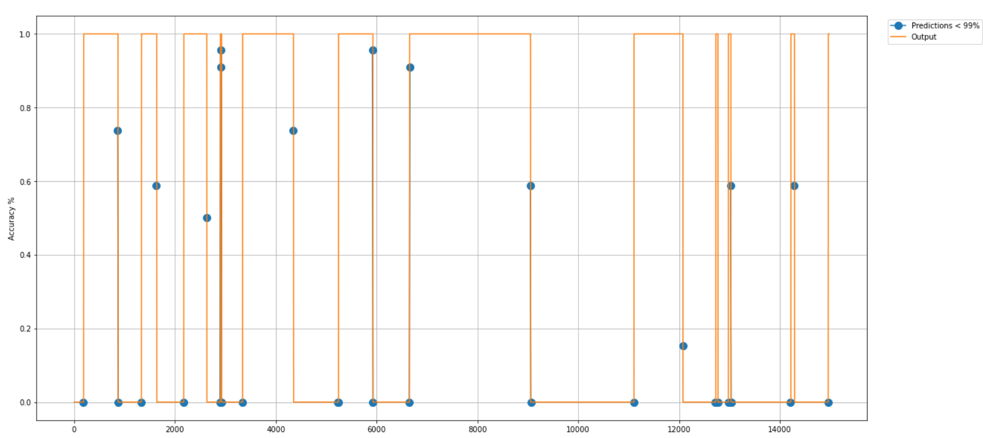

FIGURE 7: Example of walking forward

3.2 Problems

The main problem with the present study is that it involves only a single subject which raises the question whether results are generalizable. Another issue concerns the channels quantity on the headset: less sensors would reduce production cost of required EEG devices and also speed up instance-based classification, but would also provide less data.

It is also necessary to consider that the headset used in this work, would not be available in a hospital setting (where is used most of the time) because the minimum amount of channels that must be used are 16 and not 14. In fact, according to LICE[3] *"Sixteen simultaneous recording channels are currently considered the minimum number required to show the areas that produce the most normal and abnormal EEG patterns"*. Furthermore, in the 10-20 International System, which is used as the standard model for daily EEG, the minimum channels required are even 19 [12].

In the future a number of different algorithms could be empirically tried in order to be certain that Random Forest is actually the most suitable algorithm for this task.

[3] "*Standard Electroencephalography and Activation tests*"

References

[1] Benbadis, S. R.; Kaplan, P. W.; "*The Dangers of Over-Reading an EEG, Journal of Clinical Neurophysiology*", July 2019 - Volume 36 - Issue 4 - p 249 doi: 10.1097/WNP.0000000000000598

[2] Benbadis SR. "*"Just like EKGs!" Should EEGs undergo a confirmatory interpretation by a clinical neurophysiologist?*", Neurology. 2013; 80(1 Suppl 1):S47-S51.

[3] Selim R Benbadis (2010);"*The tragedy of over-read EEGs and wrong diagnosesof epilepsy, Expert Review of Neurotherapeutics*", 10:3, 343-346, DOI: 10.1586/ern.09.157

[4] P. A. Estévez, C. M. Held, C. A. Holzmann et al., "*Polysomnographic pattern recognition for automated classification of sleep-waking states in infants*", Medical and Biological Engineering and Computing, vol. 40, no. 1, pp. 105–113, 2002.

[5] M. V. M. Yeo, X. Li, K. Shen, and E. P. V. Wilder-Smith, "*Can SVM be used for automatic EEG detection of drowsiness during car driving?*", Safety Science, vol. 47, no. 1, pp. 115–124, 2009.

[6] K. Polat and S. Güneş, "*Classification of epileptiform EEG using a hybrid system based on decision tree classifier and fast Fourier transform*", Applied Mathematics and Computation, vol. 187, no. 2, pp. 1017–1026, 2007.

[7] K. Sadatnezhad, R. Boostani, and A. Ghanizadeh, "*Classification of BMD and ADHD patients using their EEG signals*", Expert Systems with Applications, vol. 38, no. 3, pp. 1956–1963, 2011.

[8] "*A First Step towards Eye State Prediction Using EEG*" Rösler et Suendermann, 2013

[9] Falessi, D., Huang, J., Narayana, L. et al. "*On the need of preserving order of data when validating within-project defect classifiers*", Empir Software Eng 25, 4805–4830 (2020). https://doi.org/10.1007/s10664-020-09868-x

[10] Breiman L."*Random forests*", Mach Learn. 2001;45:5–32.

[11] Ishwaran H, Kogalur UB, Lauer MS. "*Random survival forests*", Ann Appl Stat. 2008;2:841–860.

[12] "*American Electroencephalographic Society Guidelines for Standard Electrode Position Nomenclature*", Journal of Clinical Neurophysiology, April 1991