Data Mining,
Differences between trial phases

Diego Hernandez
dc.hernandez@ua.pt
Rodrigo Pereira
rodrigo.pereira@ua.pt

DETI
Universidade de Aveiro
Aveiro, Portugal

Abstract

During a research, a group of students were given a set of problems. The initial questions had feedback about their answer and the rest of the questions hadn't. Electroencephalography (EEG) signals were registered while the participants performed the tasks.

The proposed hypothesis to be proved is the following: students have different behaviours when receiving immediate feedback on their answers rather than not. It is then validated the following by making it possible to predict, given the metrics measured on the subject, if the problem belongs to the set of training problems or the testing one.

The collected data goes to a set of processes in order to strive for a better data quality. Processes such as: Normalization and Feature Selection. Then it is used PCA to obtain the Optimal Principal Components. As a final step, it is used Regression Learning Method and additionally it is modeled a Neural Network. The later is trained with the available cleaned data so it can make the proposed predictions.

In the end of this research, it seems clear that, in general, the behaviors of the students, represented by the EEG signals, were different when their tasks had feedback and when they hadn't.

1 Introduction

The Raven Test is at it's core a nonverbal IQ group test typically used in educational settings, in this particular case applied to students from different two different fields of study. One thing that caught our interest was the fact that the different questions, 48 in total, were divided into two different groups, the first one, composed of the first 12 questions, called the training group where the participants would receive feedback after giving an answer, and a second group (last 36 questions) where there was no feedback given.

Our theory is that the signals collected from this two different phases are not equal, since receiving feedback on the answer would increase or decrease moral depending if they got the question right or wrong. We hypothesize that changes in the subject's mood will have an influence in the signals measured.

With this work we attempt to prove our hypothesis, and if good results are shown, create the possibility of predicting if a question was made on the training or testing set of questions according to the metrics measured.

2 Data Selection and Pre-Processing

As mentioned in the introduction the dataset chosen was the RAVEN dataset. As our objective was to determine if there was any difference between trials in the testing and training phases, we obviously did not need all the data provided in that dataset. We ended up using the trial by trial data without distinction between right or wrong, and only from the DEI group, we chose to only consider one of the groups so any possible difference between them would not weight on our results.

We started by generating a CSV file with 23 columns in total (22 for the features and 1 for the classification), and about 942 rows (941 for the different entries and 1 for the feature names).

Then was time to load that data to our Notebooks. For that end we made use of the `read_csv()` method provided by the `pandas` library, which allows us to transform that CSV data into a `DataFrame` object, which allows us to work with most data mining libraries and is easily converted to and from the typical `nparray` from `numpy`

After that it was time to do some pre-processing. We started by removing any entries that had an incomplete set of features, either by having them empty or with value equal to `NaN`. After that we separated our features columns from the classification one. And finally we transformed our classification to a Boolean value where `Training` was encoded as `True` and `Testing` as `False` to facilitate some processes ahead.

3 Principal Component Analysis

We started by using Principal Component Analysis to visualize the data in a lower dimension, and see if we could see differences between the two classes. Before we applied PCA we needed to normalize the features, for that we use the `StandardScaler` provided by the `sklearn` library, which standardizes the data by removing the mean and scaling to unit variance.

After that we applied PCA with only two Principal Components and plotted the result [Figure 1], as expected we could not see any differences, since the sum of the variance ratio for the two components only amounted to 26%.

We then experimented with three components [Figure 2], but the sum of the variance ratios only rose by 9% to a total of 35%.

Since we where experimenting with PCA we wanted to visualize the results of Kernel PCA compared to the normal PCA to see if the 2 classes would diverge a bit more, but as expected independently of the kernel function three principal components where just too little and a lot of variance was being lost [Figure 3].

We ended up making use of `sklearn` PCA feature that allowed us to calculate the number of Principal Components needed to maintain a percentage of variance, in our case we told the algorithm to keep 90% of the total variance, and ended up with 13 principal components. So in the end the we reduced dimensional by 9 while keeping 90% of the variance. Further in our work we will test using this 13 features we got and compare them to using all the initial features.

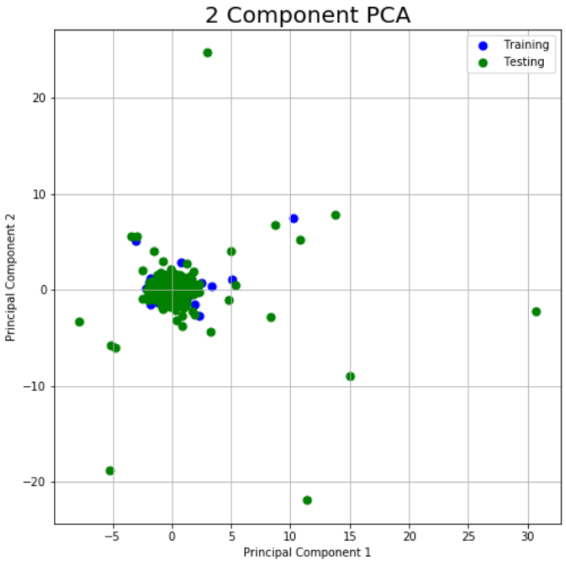

Figure 1: PCA with 2 Principal Components

4 Regression Learning Method application

Now that we reduced our dimensionality we wanted to try and separate the two classes. And hopefully end up with a model that given data from a new trial would rightfully predict to which category that trial belonged.

We chose to use Random Forest Learning Method. To do so we had first to split our data into training and testing data, this to allow us to verify at the end if the model was rightfully classifying the data or if he was

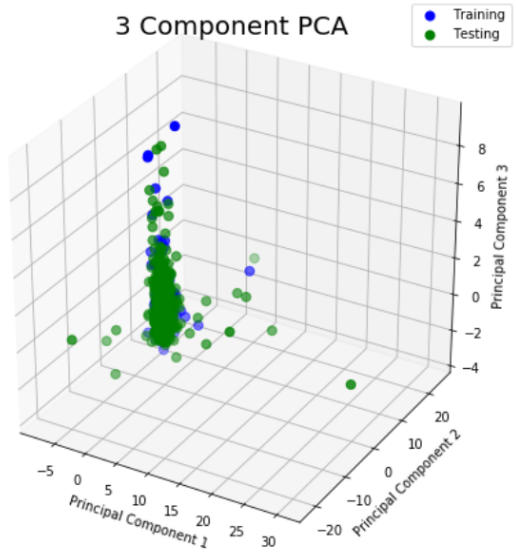

Figure 2: PCA with 3 Principal Components

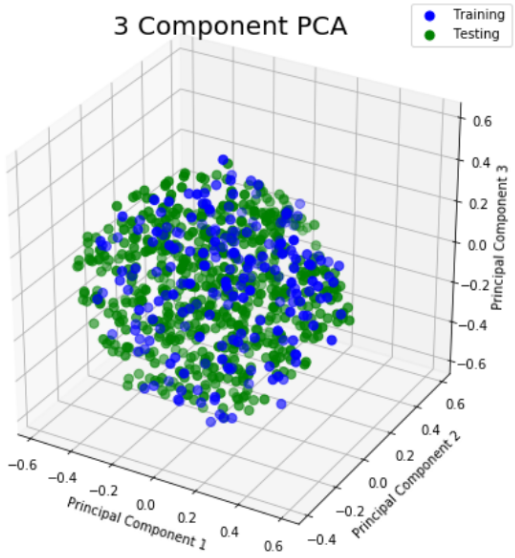

Figure 3: KPCA with 3 Principal Components, using Radial basis function kernel

to biased to the training data given. Since we had a imbalanced dataset (the number of entries of one class was much larger than the other), we strayed away from using methods such as holdout, since the metrics we would measure at the end would be very dependent on the division made. The method we chose to adopt was K-Fold Cross-Validation, that would iteratively split differently the training and testing sets (in our case we used 20 different splits) making so the metrics given at the end would end up being a mean from the different iterations.

As mentioned in the previous chapter we also wanted to test how much we were losing by using 13 Principal Components as features instead of the original 22. We tested for the two cases to compare metrics:

Metric	With 22 Features	With 13 Features
Accuracy	0.73988	0.73861
Precision	0.28750	0.26000
Recall	0.04938	0.05356
F1 Score	0.08172	0.08294

Table 1: Comparison between metrics from Random Forest, with and without dimentionality reduction from PCA

Making use of the fact that we can obtain the importance of each feature with the Random Forest method, we obtained those values and plotted in order to support further conclusions as seen in [Figure 4]

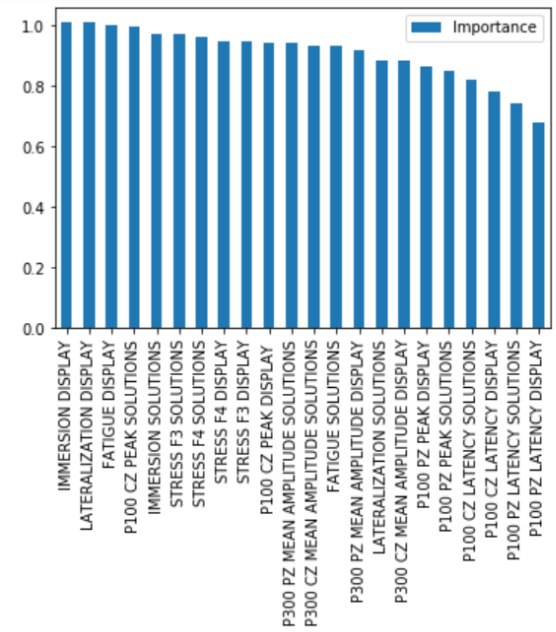

Figure 4: Random Forest Feature importance

5 Neural Network

A neural network has been modeled during this investigation. A neural network is a series of algorithms that strives to recognize the relationships in a data set. It adapts itself to input changes, generating the best possible result without having to redraw the output criteria.

5.1 Implementation

Initially, it has been done each of the following: It has been trained and tested a neural network model; Then used a grid search to optimize the hyper parameters; and finally, submitted predictions for the test set.

First, after the loading and minor adjustments of our data, we normalize the data with a standard scaler.Then, it has been adopted PCA, in order to find the principal components to take into consideration to model and train our Neural Network. The resulting principal components were in total 13, instead of 22 features. Later the data has been organized in a random manner and splitted the data in order to have a training set and a test set. We use 20% of the total data as a test set.

We additionally tested Pearson Correlation in order to select only the features that would contribute most to the quality of the resulting model. It eliminates those attributes that are redundant and do not add additional value to the model. It has been applied to select different ranges of attributes, but it haven't accomplished any upgrades in the accuracy.

Furthermore, since the data has been cleaned and scaled we feed it to our neural network. It has been created a simple model architecture. We have one input layer, that feeds into a hidden layer with nodes, as default, and an output layer which is used to predict if the user is at the Training phase our Testing Phase. The output layer has a Sigmoid activation function, in order to have as output values between 0 and 1. Since our problem is a binary classification problem a binary cross entropy loss function was used. It has been created a function with parameters to facilitate the optimization of the neural network's hyper parameters.It is to note that our neural network allows for a dropout module (which is also tested for optimization). It ignored units, also known as neurons of the neural network, during the training phase of certain set of neurons which is chosen at random, not being considered during particular forward of backward pass.The reason of implementing a dropout was: A fully connected layer occupies most parameters, and therefore neurons develop codependency with each other during training, which restricts the individual power of each neuron, leading to over-fitting of the training data.

Before we proceed on training our neural network with the cleaned data-set, it is used GridSearch to find out what are the optimal values for the neural network's hyper parameters, which are: Batch size, Epochs, Optimization Algorithm, Hidden Layers, number of Neurons, and Dropout.

Finally, we create our final model, and train it. We use K-fold cross */

validation in order to evaluate the obtained results. We split our data into 20 sets, and for each iteration one set is remained for testing in order to do the validation.

5.2 Results

Because the results obtained weren't so relevant with the use of *Pearson's Correlation*, it was opted to not show the results with its appliance. However the implementation of the *Pearson's Correlation* is written in the respective Notebooks so the programmer/analysts could be capable to see the results with our without its implementation.

In this following subsection it will be shown the obtained results obtained by the Neural Network model trained using all of the data features and using only the optimal Principal Components calculated by PCA.

5.2.1 Using All Features Data:

The optimal hyperparameters in the Neural Network are:

Hyperparameter	Value
Batch Size	8
Epochs	10
Layers	[11, 5]
Drops	0.5

Table 2: Optimal Hyperparameters of Neural Network model, whom feeds from data of all the available dataset features

The model was evaluated with a testing set, which belongs from the data-set. So by using Holdout strategy we obtained the following results:

Metric	Result
Tested Accuracy	0.7714
Tested Loss	0.5548

Table 3: Results of Neural Network Metrics using all data-set features.

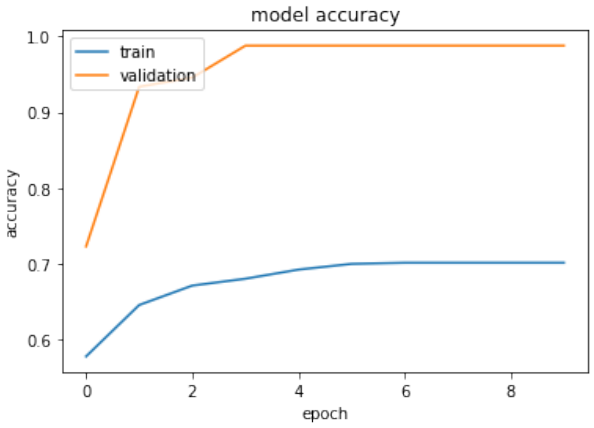

Figure 5: History of accuracy. This figure belongs to the Neural Network Training using Holdout strategy.

Using the K-fold strategy, the data-set is partitioned into 20 partitions, and in each iteration 1 of the sub-set is used as test set, and the rest as training sets.

Taking into account the accuracy and recall values obtained in each iteration, final the results were the following:

Metric	Result
Mean Accuracy	0.7616
Mean Recall	0.0219

Table 4: Results of Neural Network Metrics after using K-Fold Strategy of 20 partitions.

Hyperparameter	Value
Batch Size	16
Epochs	50
Layers	[6]
Drops	0.0

Table 5: Optimal Hyperparameters of Neural Network model, considering the Optimal Principal Components.

5.2.2 Using Principal Components:

The optimal hyperparameters in the Neural Network are:

The model was evaluated with a testing set, which belongs to the dataset. So by using Holdout strategy we obtained the following results:

Metric	Result
Tested Accuracy	0.6977
Tested Loss	0.6403

Table 6: Results of Neural Network Metrics using Optimal Principal Components.

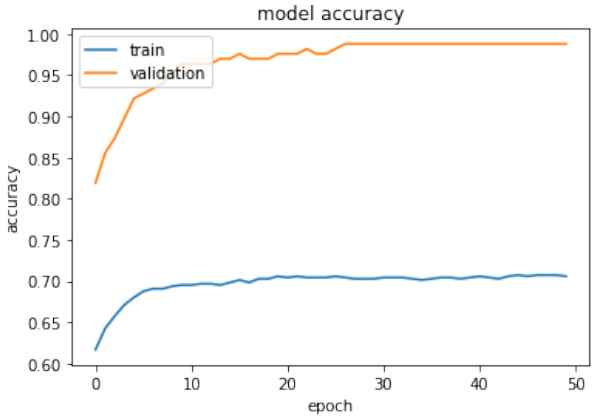

Figure 6: History of accuracy. This figure belongs to the Neural Network Training using Holdout strategy.

Using the K-fold strategy, the data-set is partitioned into 20 partitions, and in each iteration 1 of the sub-set is used as test set, and the rest as training sets.

Taking into account the accuracy and recall values obtained in each iteration, final the results were the following:

Metric	Result
Mean Accuracy	0.7304
Mean Recall	0.1078

Table 7: Results of Neural Network Metrics after using K-Fold Strategy of 20 partitions.

5.3 Observations

Having the Neural Network model set with the optimal Hyperparameters, it is noticeable that the best accuracy is achieved by training the model considering all the features of the dataset instead of using the optimal Principle Components calculated by PCA. This applies for both Holdout Strategy and K-Fold strategy.

Having a Neural Network model with the optimal Hyperparameters and trained with all the data-set features, it has been achieved an accuracy equal to 76%, approximately.

6 Conclusions

With the results we obtained from the different procedures we executed we can derive some conclusions relative to the hypothesis we proposed at the beginning of this assignment.

First we can affirm that, despite the accuracy of the predictions only being of approximately 75%, the two classes show differences in terms of their values, making them distinguishable. As we hypothesized at the beginning of this assignment the participants being told the solution seem to affect their mental state, if we look at the feature importance given by the Random Forest we see that factors such as immersion seem to have a great impact on the results.

7 Contribution

The contribution of both elements of the groups was equal, and as such we consider an attribution of 50% of work contribution to both elements fair.

*/