EEG Brain Wave for Confusion

Eric Dagobert
CUNY Graduate Center

February 20, 2017

Abstract

The aim of this paper is to define a prediction model for the confusion mental state, in order to improve the didactic value of online educative video by detecting whether the student is confused or not watching the video. The data related to this study is available on Kaggle(`https://www.kaggle.com/wanghaohan/eeg-brain-wave-for-confusion`). We will show here how Moving Average can improve the accuracy of the prediction model and help determining the causal inference of the different variables available.

The data set

The data set consists in EEG signal data from 10 college students while they watched MOOC video clips. After watching the video, each student rated his confusion level. The confusion level is translated into 0 and 1 for not confused/confused.

Variables:

The data set comes to the form of a CSV file and has the following columns:

- Subject ID
- Video ID
- Attention: Proprietary measure of mental focus
- Mediation: Proprietary measure of calmness
- Raw: Raw EEG signal
- Delta: 1-3 Hz of power spectrum
- Theta: 4-7 Hz of power spectrum
- Alpha 1: Lower 8-11 Hz of power spectrum
- Alpha 2: Higher 8-11 Hz of power spectrum
- Beta 1: Lower 12-29 Hz of power spectrum
- Beta 2: Higher 12-29 Hz of power spectrum
- Gamma 1: Lower 30-100 Hz of power spectrum
- Gamma 2: Higher 30-100 Hz of power spectrum
- predefined label :whether the subject is expected to be confused
- user-defined label :whether the subject is actually confused

Delta, theta, beta and gamma are brain waves frequencies. Electromagnetic waves produced by the human brain are classified depending on their pattern and frequency range. 'Attention' and 'mediation' are supposed to represent the level of mental concentration. Also, 'raw' is supposed to represent the electromagnetic signal emitted by the brain.

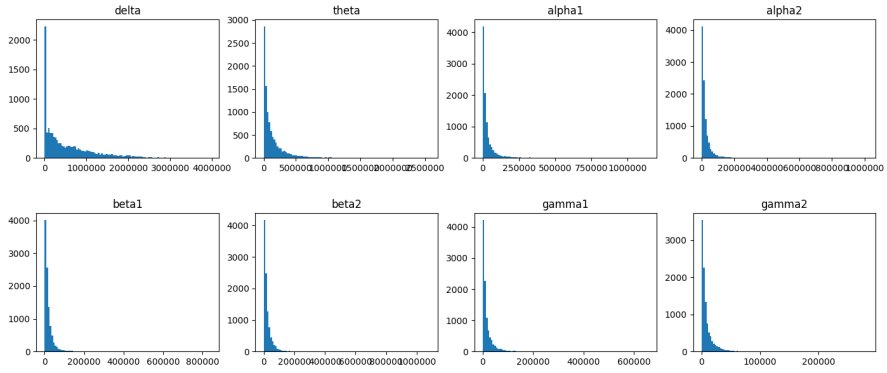
On top of that, a file of demographic data containing gender, age, and ethnicity of each subject can be joined to the data set. We choose to include the demographic data here but the results will show, as expected, that it has little impact on the prediction model. Another thing is in this study we will drop the variable 'predefined label' because it may induce a bias. We also dropped the Video Id and the Subject Id for obvious reasons.

Preliminary analysis

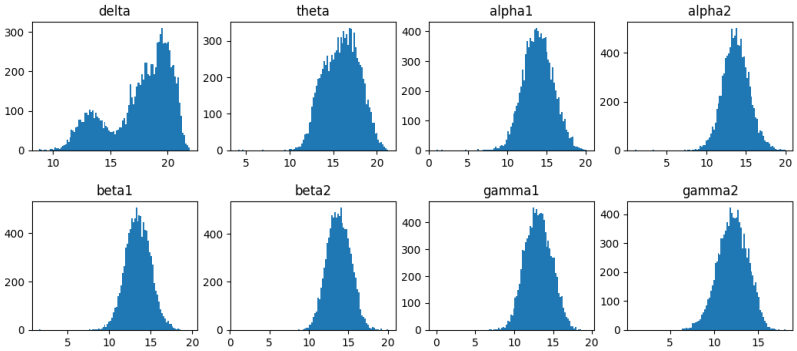
Before proceeding with data analysis, visualizing data may be very helpful to know the type of data preprocessing must be performed (such as normalization) and also to determine which type of machine learning may lead to the best predictions.
One of the first things to do with a complex data set is to visualize it in order to find a good statistical model that will fit the data. Knowing the type of distribution is fundamental in order to determine the causal relationship between variables. Furthermore, the shape of the distribution helps determining which type of noise reduction or quantization can be subject to prediction improvement. Last, the distribution chart can also show the existence of mixtures, which can be perfectly defined with specific algorithms such as EM.

Individual variables

The chart below shows that all the variables representing frequencies have some asymmetrical distributions, and look like log-normal distributions.



After applying logarithm on these variables we obtain distribution patterns as shown below. These patterns are very close to a Normal distribution:



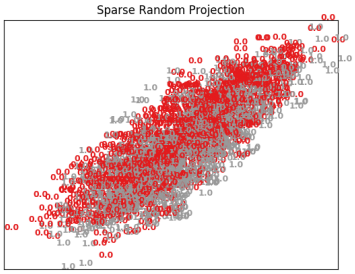
We remark that 'delta' seems to be sum of two variables. If necessary we can separate them using EM algorithm or Nearest Neighbor (and we will see later that the Nearest Neighbors algorithm performs quite well in some conditions).

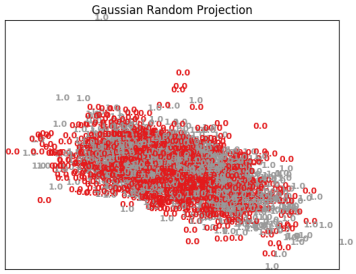
Variables inference

We have here a problem of classification; the idea is to separate two 14-dimension spaces: One corresponding to the 'confused' state, an the other 'non confused'.

We first used different 2-D projections, random and PCA, to check whether there is an obvious separation between the two target sets (confused/non confused).

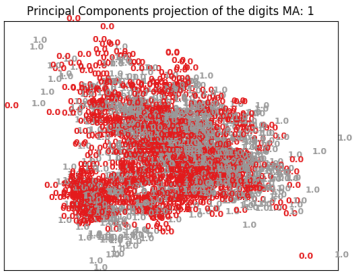
- Random Projection consists in finding a reduction of dimensionality by projecting the original dataspace using a random matrix multiplier. This is equivalent to finding a low rank approximation as described in [HMT]. The python library [sklearn] offers two types of projection: one using gaussian matrices, the other using sparse matrices:





  We tried many different random projections without noticing any obvious separation.

- PCA (Principal Component Analysis) is another projection based on covariance axis; we do a projection along the highest eigenvalues of the covariance matrix, obtained via SVD decomposition:



  Unfortunately the PCA decomposition didn't show any obvious data separation neither.

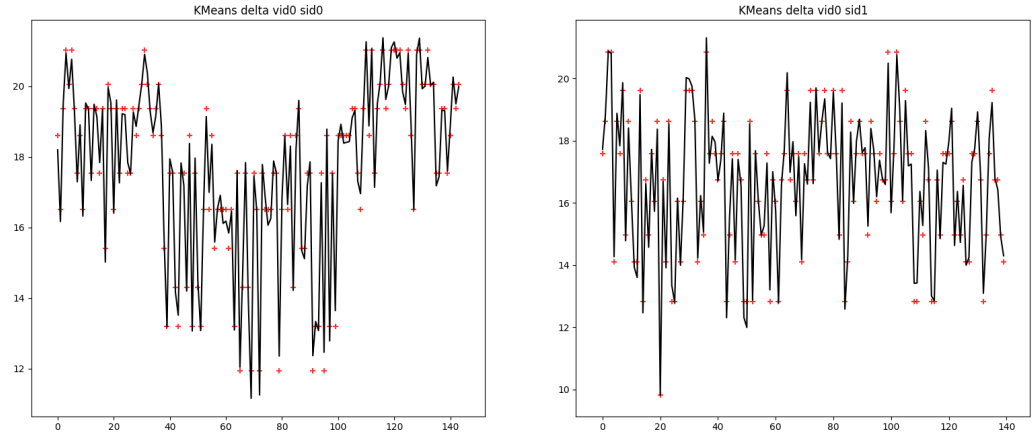
Not surprisingly the accuracy of different machine learning models applied to the data set is quite low: around 60% with Classification Trees and between 60% and 65% using SVM or Gaussian Mixture models.

Noise removal

By watching at the different projections and also at the time serie representation of the data we conclude there exists a significant amount of noise that could be removed. Different techniques are used to do so. We choose here to look at quantization and moving average.

KMeans

K-Means is a technique of quantization aiming to cluster data according to spatial affinities, such as Euclidean distance, or the variance for example. The idea is to remove noise by grouping data close together and substituting them with their center of gravity. K-Means separates data of same variance, but there exist other algorithms such as Nearest Neighbors, using a different distance function.
We performed the K-means algorithm using [sklearn] on every distinct pair (Subject,Video). An example of K-Means represented by red dots is shown next:



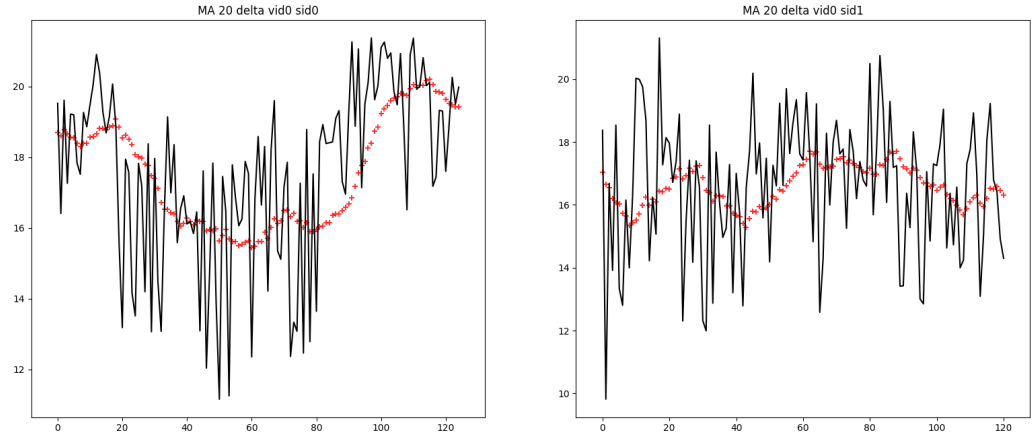
The data is then replaced by K-Means which are fed to a Tree Classifier and a Random Forest. Nevertheless, the accuracy of the prediction model is quasi unchanged.

Moving Average

Moving average is a technique used for Financial Time Series ([DMA]), often modelized as *Wiener Process*:

$$W_{t+u} - W_t \sim \mathcal{N}(\mu, \sigma_t)$$

Thus the moving average $\mathrm{MA}(t,u) = \frac{1}{u}\sum_{t-u}^{t} W_i$ becomes (almost) independent of $\mathcal{N}(0,\sigma_t)$. The idea beneath the Moving Average is a quick way to suppress the noise, thus highlight the *trend*.
The charts below show an example with Delta waves along with their moving average of 20 elements for two (subject,video):



Moving averages are extracted from the data using Python and [panda]:

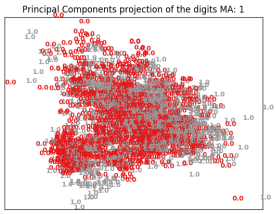
```
for i in np.unique(p.vid.values):
    for j in np.unique(p.sid.values):
        for col in ["med","att","raw","delta","theta","alpha1","alpha2","beta1"
            ,"beta2","gamma1","gamma2"]:
                d = p.ix[(p.vid==i)&(p.sid == j),col].rolling(window=wsize).
                    mean()
                p.ix[(p.vid==i)&(p.sid == j),col]=d

# ...
p = p.dropna()
```

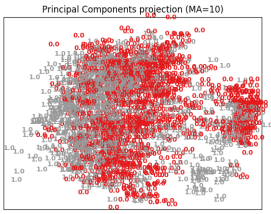
We can see that the variance of the red spots is much smaller than the variance of the signal, therefore we have significant noise reduction.
But if quantization successfully remove noise too, the moving average has a very important property: it contains information about the *sequence* of data; every MA point is computed from its past and represents a sort of short-term memory. This is very important because quantization clusters are based on two directions: backward and forward. In our case, data sequence is a factor we should consider because brain wave signals here are sequential. As machine learning algorithms don't care about the order in which the data is represented, keeping track of the sequence is important because it may corresponds to a physiological parameter such as the human short term memory for instance.
In fact, we will see in the next section that considering the sequence factor, on top of noise reduction significantly improve the accuracy of the prediction models.

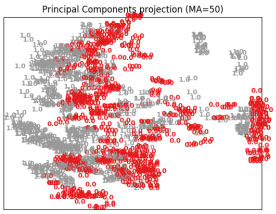
Results

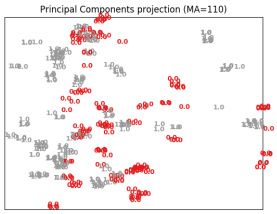
PCA and Moving Average

We can see below that increasing the window size of the moving average improves data separation:









Unfortunately, increasing the moving average window size also increases overfitting; for a window size of N, the $N-1$ first samples are not available anymore. So, roughly, the data set size available for training and testing is $s \sim 100 \times (112 - N)$, 112 being the smallest number of measures for a given video/subject. A rough estimate can be that the sample size is reduced by $N\%$ for a window size of N. So as the window size increases, the sample size decreases proportionally. That phenomenon can be seen on the PCA charts above, the most extreme case being MA size=100.

Cross validation and accuracy

We computed the cross validation 5-fold accuracy on half of the data and the F1 score on the other half for different window sizes, this for different [sklearn] machine learning processes, both supervised and unsupervised.
We noticed that every algorithm's accuracy increased linearly with window size (until a maximum is reached), which means MA really has a positive impact on the prediction model. Also, all the models reached a maximum accuracy between 1 and 0.8, which confirmed the improvement due to MA. The F1 score is consistent with accuracy and overall well-balanced between false negative and false positive. That means first the data sample is correctly balanced, and secondly the machine learning algorithms are not biased towards false positive or false negative.
We selected some of the best algorithm/MA and measured their ROC/AUC (Area Under the Curve)

Table 1: Cross Validation 5-Fold

MA size:	1	2	3	4	5	7	10	15	20	25	30	40	50	60	100
KNeighborsClassifier	0.66	0.68	0.73	0.78	0.82	0.87	0.91	0.95	0.97	0.98	0.99	1.00	1.00	1.00	1.00
SVC(kernel='linear', C=0.025)	0.62	0.63	0.64	0.66	0.67	0.68	0.70	0.72	0.73	0.74	0.75	0.76	0.78	0.79	0.79
SVC(gamma=2, C=1)	0.61	0.65	0.72	0.78	0.82	0.88	0.93	0.97	0.98	0.99	1.00	1.00	1.00	1.00	1.00
GaussianProcessClassifier(RBF)	0.69	0.72	0.76	0.80	0.83	0.87	0.91	0.95	0.97	0.99	1.00	1.00	1.00	1.00	1.00
DecisionTreeClassifier	0.61	0.62	0.67	0.67	0.72	0.75	0.80	0.85	0.88	0.92	0.94	0.94	0.96	0.98	0.99
RandomForestClassifier	0.70	0.71	0.75	0.77	0.80	0.84	0.88	0.93	0.94	0.97	0.98	0.99	0.99	0.99	1.00
MLPClassifier	0.68	0.70	0.73	0.74	0.77	0.79	0.82	0.84	0.86	0.87	0.89	0.89	0.93	0.95	0.97
AdaBoostClassifier	0.62	0.65	0.67	0.69	0.71	0.72	0.74	0.77	0.78	0.81	0.83	0.86	0.89	0.92	0.98
GaussianNB	0.60	0.60	0.61	0.62	0.64	0.64	0.65	0.68	0.68	0.69	0.69	0.70	0.71	0.70	0.72
QuadraticDiscriminantAnalysis	0.65	0.67	0.70	0.71	0.72	0.73	0.75	0.78	0.79	0.81	0.81	0.84	0.84	0.88	0.90

Table 2: F1 score(average on 10 cycles of training/test)

MA size:	1	2	3	4	5	7	10	15	20	25	30	40	50	60	100
KNeighborsClassifier	0.64	0.67	0.72	0.77	0.81	0.86	0.91	0.95	0.97	0.99	0.99	0.99	1.00	1.00	1.00
SVC(kernel='linear', C=0.025)	0.61	0.64	0.65	0.66	0.67	0.69	0.70	0.72	0.73	0.74	0.75	0.77	0.78	0.79	0.80
SVC(gamma=2, C=1)	0.64	0.65	0.71	0.77	0.80	0.89	0.87	0.93	0.97	0.99	0.99	1.00	1.00	1.00	1.00
GaussianProcessClassifier(RBF)	0.67	0.72	0.75	0.78	0.82	0.92	0.96	0.98	0.99	0.99	0.99	1.00	1.00	1.00	1.00
DecisionTreeClassifier	0.60	0.62	0.65	0.68	0.71	0.75	0.81	0.86	0.89	0.91	0.93	0.95	0.96	0.97	0.99
RandomForestClassifier	0.68	0.71	0.74	0.77	0.80	0.84	0.89	0.93	0.95	0.97	0.97	0.98	0.99	0.99	1.00
MLPClassifier	0.67	0.71	0.71	0.74	0.76	0.78	0.82	0.84	0.85	0.87	0.90	0.90	0.93	0.94	0.97
AdaBoostClassifier	0.61	0.65	0.67	0.68	0.70	0.71	0.74	0.76	0.78	0.81	0.82	0.86	0.90	0.93	0.97
GaussianNB	0.59	0.60	0.63	0.64	0.64	0.65	0.66	0.66	0.68	0.69	0.69	0.71	0.71	0.70	0.74
QuadraticDiscriminantAnalysis	0.65	0.68	0.68	0.70	0.71	0.75	0.75	0.77	0.79	0.81	0.82	0.83	0.84	0.86	0.91

scores to have a better idea of the false positive/false negative proportions. We also kept the Decision Tree Classifier because it was the one chosen by the authors of the Kaggle study (plus we believe that practically, a Decision Tree is very easy to implement into a small electronic portable device).

Table 3: ROC score(cross-validation 5-fold)

MA size:	1	5	10	15	20	25	30
KNeighborsClassifier	0.70	0.88	0.96	0.99	0.99	1.00	1.00
SVC(gamma=2, C=1)	0.70	0.90	0.98	1.00	1.00	1.00	1.00
DecisionTreeClassifier	0.60	0.72	0.80	0.85	0.88	0.92	0.94

Results are compatible with the F1 score and shows almost perfect classification starting from MA=10. That would mean the data set is almost completely explained with a machine learning model.

Feature Importance

To measure the feature's importance, we computed the relative ROC/AUC scores of every single feature through cross-validation by randomly shuffling the feature considered, and compute the score difference that occurs:

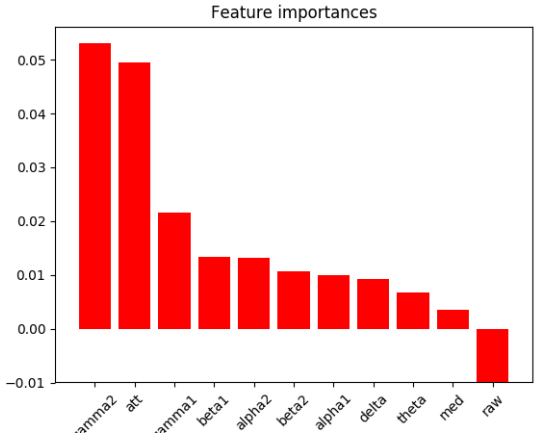
```
def fimportance(clf,X,y):
    score = np.mean(cross_val_score(clf,X,y,scoring='roc_auc',cv=5))
    importance = {}
    for i in range(X.shape[1]):
        X_feat = X.copy()
        X_feat[:,i] = random.sample(X[:,i].tolist(),X.shape[0])
        feat_score =  np.mean(cross_val_score(clf,X_feat,y,scoring='roc_auc',cv
            =5))
        importance[i]=score-feat_score
    return importance
```

We computed the feature importance for some models and for moving averages of sizes (1,5,10,15). We have extended the Decision Tree Classifier to MA sizes 20,25,30 and 35 to reach levels of accuracy close to 99%.

Table 4: 10 most important features / MA

Feature importance:	1	2	3	4	5	6	7	8	9	10
KNeighborsClassifier 1	att	gamma2	gamma1	beta2	alpha2	theta	delta	beta1	alpha1	gender
KNeighborsClassifier 5	att	gamma2	gamma1	beta1	delta	beta2	alpha1	theta	med	alpha2
KNeighborsClassifier 10	att	gamma2	gamma1	alpha2	alpha1	beta2	delta	gender	theta	age
KNeighborsClassifier 15	att	gamma2	gamma1	alpha2	gender	delta	theta	beta2	beta1	med
SVC 1	att	gamma1	gamma2	beta2	alpha2	delta	beta1	alpha1	theta	ethnicity
SVC 5	gamma2	att	gamma1	beta2	age	delta	alpha1	theta	beta1	ethnicity
SVC 10	gamma2	att	gamma1	beta2	alpha2	beta1	theta	delta	alpha1	age
SVC 15	gamma2	att	gamma1	alpha2	beta1	beta2	age	alpha1	med	delta
DecisionTreeClassifier 1	att	delta	beta2	theta	gamma1	beta1	raw	alpha1	gamma2	gender
DecisionTreeClassifier 5	att	theta	gamma1	beta1	delta	alpha1	raw	ethnicity	gender	med
DecisionTreeClassifier 10	alpha2	theta	beta2	att	beta1	gamma1	gender	raw	alpha1	delta
DecisionTreeClassifier 15	gamma1	alpha1	theta	gamma2	beta2	beta1	alpha2	delta	raw	gender
DecisionTreeClassifier 20	age	gamma1	alpha1	beta1	gamma2	alpha2	gender	raw	med	theta
DecisionTreeClassifier 25	delta	att	gamma2	raw	ethnicity	gender	med	alpha2	theta	beta1
DecisionTreeClassifier 30	gamma2	att	alpha2	theta	delta	beta1	alpha1	gender	ethnicity	med
DecisionTreeClassifier 35	beta2	gamma2	alpha2	theta	age	att	med	delta	raw	beta1

We can notice two things from Table 10: first, the Decision Tree classifier seems not very reliable as the feature's importance seems to change randomly. Secondly, the demographic features have a very low impact on the classification as expected (and we verified using ROC scores computed after the removal of demographic columns).
The histogram of feature's importance below has been computed with a Support Vector classifier and a MA of 15.



Most of machine learning algorithms agree on the same feature importance, namely the 'attention' level and Gamma waves levels.
Another interesting point is the fact that 'raw' plays such a little role, and sometimes has a negative score: as suspected, it probably represents all the brain waves and therefore is redundant with alpha, beta, delta, gamma and theta measurements. If we don't know precisely how the 'attention' is calculated, it is nevertheless obvious that attention given to the video is a key factor for its comprehension. Another interesting fact is the Gamma waves levels, which seem to be associated to the visual cortex activity, can possibly explain the confusion level. This can be representing the fact that, once in a confused state, the subject either increases its attention to catch up, or inversely gives up and stops taking an interest in the video. On the other hand a non-confused subject keeps the same level of brain activity because he doesn't have to produce extra energy to understand the video.

Conclusion

There are several conclusions to draw from this study. First, using the moving average is significant in terms of score improvement (F1 and AUC), for various machine learning algorithms, both supervised and unsupervised. It appears from our results using a MA with a window size of 15 seems to be a good compromise between accuracy and overfitting, and yet can predict confusion from the EEG data with a probability over 90%. Another point is that machine learning systems using the kernel trick and particularly a radial kernel have better results, for smaller window sizes. The Support Vector Machine we have used has a radial kernel $K(x, x') = \exp(\gamma\|x - x'\|)$ applied to $log(F_{Hz})$ with $\gamma = 2$. This would be equivalent to the energy $E = F_{Hz}^2$ carried by a wave; the MA therefore would represent such energy over a given period of time. So an explanation of the confusion state could be lying in the energy *levels* of gamma waves, in other words the intensity of activity in the visual cortex. Of course this hypothesis deserves further detailed analysis. We insist on the word *level* because of the good results of Nearest Neighbors which would indicate that not only noise reduction but also quantization plays a role here. Nevertheless, an explanation of why the moving average works in this particular case would necessitate further study involving more data, especially more subjects, as well as the biological significance of some concepts such as Attention and Mediation.

References

[sklearn] @articlescikit-learn, title=Scikit-learn: Machine Learning in Python, author=Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., journal=Journal of Machine Learning Research, volume=12, pages=2825–2830, year=2011

[panda] `https://pypi.python.org/pypi/panda`

[HMT] Halko, Martinsson, Tropp, *Finding Structure With Randomness: Probabilistic Algorithms For Constructing Approximate Matrix Decompositions*. SIAM review, Vol.53, No. 2, pp 217-288

[DMA] Johnson, Barry *Algorithmic Trading and DMA* 4Myeloma Press, 2010