Dedole Tommy
https://github.com/D-Tommy/IBM-Course

Pair-reviewed project :
Emotion classification based EGG signals

Dataset Used :

- Aston university :
  - EEG Brainwave Dataset: Feeling Emotions
  - **J. J. Bird, L. J. Manso, E. P. Ribiero, A. Ekart, and D. R. Faria, "A study on mental state classification using eeg-based brain-machine interface,"in 9th International Conference on Intelligent Systems, IEEE, 2018.**
  - **J. J. Bird, A. Ekart, C. D. Buckingham, and D. R. Faria, "Mental emotional sentiment classification with an eeg-based brain-machine interface," in The International Conference on Digital Image and Signal Processing (DISP'19), Springer, 2019.**
  - (PDF) Mental Emotional Sentiment Classification with an EEG-based Brain-machine Interface

API : kaggle datasets download -d birdy654/eeg-brainwave-dataset-feeling-emotions

direct link : EEG Brainwave Dataset: Feeling Emotions

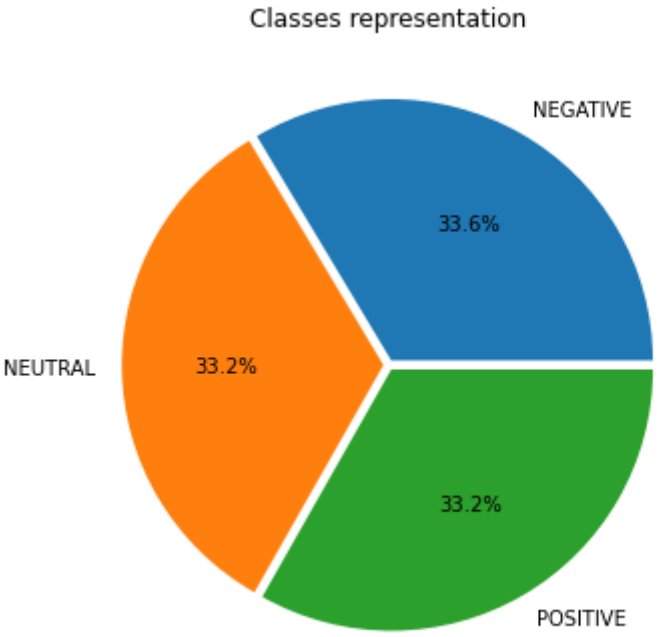
Content

This is a dataset of EEG brainwave data that has been processed with the original strategy of statistical extraction (paper above). It contains 2131 entries each having 2539 features.

The data was collected from two people (1 male, 1 female) for 3 minutes per state - positive, neutral, negative. We used a Muse EEG headband which recorded the TP9, AF7, AF8 and TP10 EEG placements via dry electrodes. Six minutes of resting neutral data is also recorded, the stimuli used to evoke the emotions are below

Stimulus	Valence	Studio	Year
Marley and Me	Neg	Twentieth Century Fox, etc.	2008
Up	Neg	Walt Disney Pictures, etc.	2009
My Girl	Neg	Imagine Entertainment, etc.	1991
La La Land	Pos	Summit Entertainment, etc.	2016
Slow Life	Pos	BioQuest Studios	2014
Funny Dogs	Pos	MashupZone	2015

Target feature distribution :



Objective :

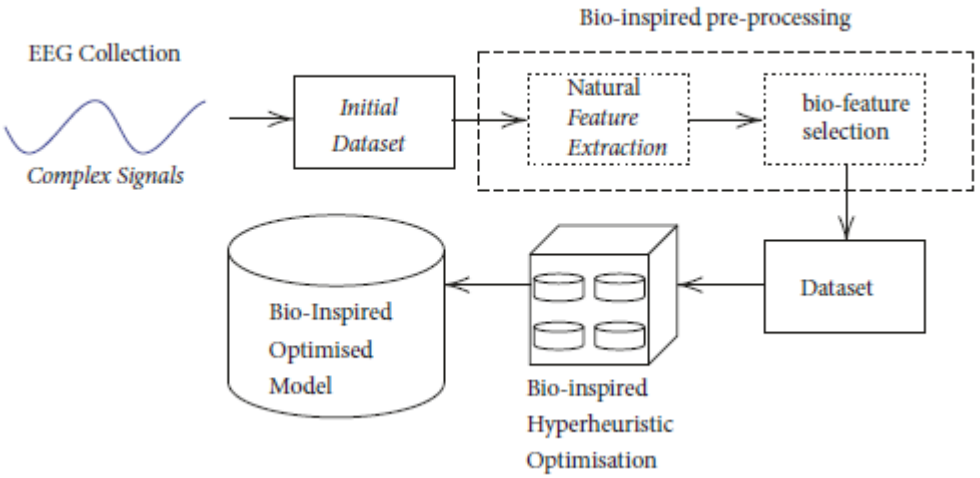
Use classifications models in order to classify EEG as Positive/Negative/Neutral.

- In this case, the focus will be made on prediction while not giving that much importance to interpretation since the dataset has already been processed and presents many engineered features.

Code used for this project is available on my github

1) Data cleaning and exploration

Processing workflow followed by the research team on this dataset.

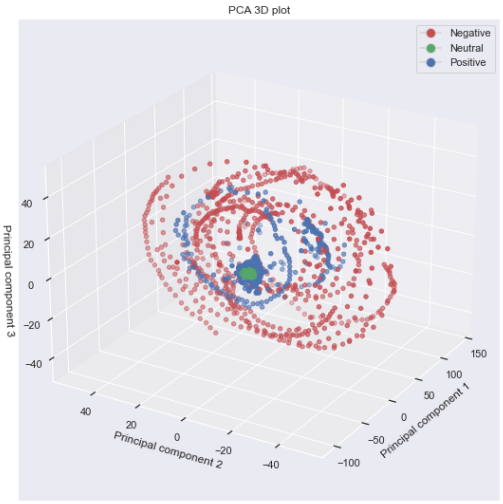


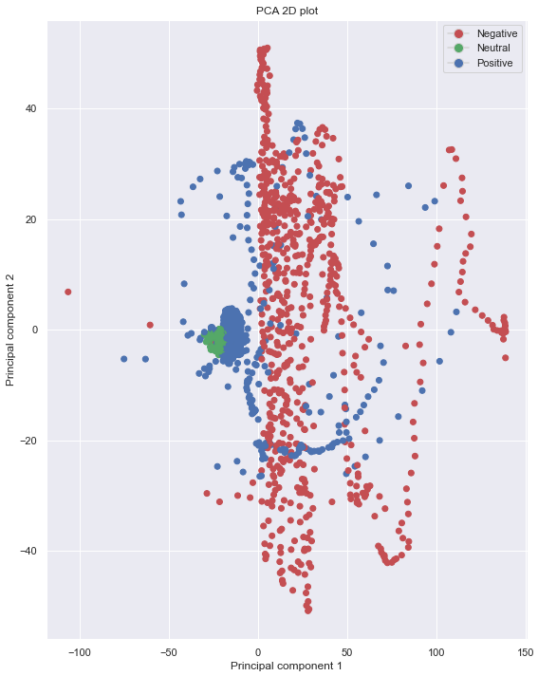
ref : A Deep Evolutionary Approach to Bioinspired Classifier Optimisation for Brain-Machine Interaction,
Jordan J. Bird , Diego R. Faria, Luis J. Manso,
Anikó Ekárt, and Christopher D. Buckingham, Published in March 2019 in ResearchGate

Since the dataset has already been precisely processed, it does not require any cleaning or encoding. However, for curiosity purposes PCA and t-SNE will be performed in order to reduce the dimensionality and models will be trained on these newly formed dataset to see how the training time can be reduced and at which cost.

1.1) Principal Component Analysis.

PCA uses eigenvectors and matrix diagonalization in order to extract linear combinations of original features that maximise variance between elements.
It is mainly used to reduce dimensionality of a dataset to have better visualization.





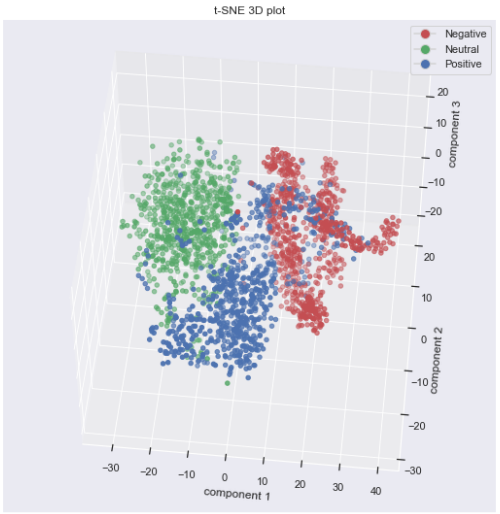

```
Entrée [79]:  1 print(f'Explained variance per component for 2D PCA : {pca2.explained_variance_ratio_}')
              2 print(f'Explained variance per component for 3D PCA : {pca3.explained_variance_ratio_}')

Explained variance per component for 2D PCA : [0.36772564 0.09553894]
Explained variance per component for 3D PCA : [0.36772564 0.09553894 0.08436069]
```

The 3 first components PCA explains 54.8% of the total variance of our dataset and offer us interesting information : neutral EEG should almost never be mistaken as negative EEG since the two populations are already well spread. However, other false-classification cases might be more frequent.

1.2) T-distributed stochastic neighbor embedding

t-SNE is another dimensionality reduction algorithm that is supposed to spread apart different classes and group together elements of the same class.



As expected, t-SNE provides one with more spreaded clusters, especially in 2D. It visually confirms what PCA showed : Neutral should rarely be mistaken as another class.

2) Classifications

2.1) Simple models

Score displayed for simple models are average unweighted scores.

2.1.1) Using full dataset.

	accuracy	precision	recall	f1
Logistic Regression	0.9609	0.9614	0.9615	0.9614
KNN	0.9406	0.9413	0.9445	0.9408
SVC	0.9422	0.9427	0.9452	0.9426
TreeClassifier	0.9688	0.9691	0.9690	0.9690

Simple models perform rather well on the whole dataset, let's try using the dimension-reduced one.

2.1.2) Using 3D dataset from tSNE

	accuracy	precision	recall	f1
Logistic Regression	0.8724	0.8717	0.8742	0.8712
KNN	0.9306	0.9301	0.9318	0.9300
SVC	0.9118	0.9110	0.9178	0.9101
TreeClassifier	0.8931	0.8933	0.8939	0.8932

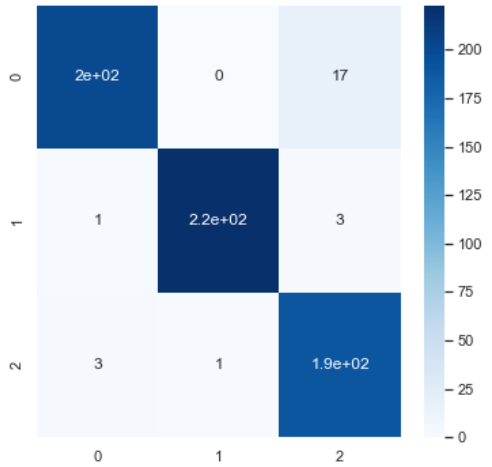
Results are obviously less qualitative than before but they remain acceptable given the fact we dropped from 2539 features to 3 only.

2.2) Combining simple models

Now that 4 simples have been trained, they might provide better results together :

```
estimators = [('lr_l2', lr), ('KNN', knn), ('SVC', svc), ('Tree Classifier', dtc)]
vc = VotingClassifier(estimators = estimators, voting = 'hard')
vc.fit(X_train, y_train)
y_pred = vc.predict(X_test)
print(classification_report(y_pred, y_test))
```

	precision	recall	f1-score	support
0	0.99	0.92	0.95	227
1	1.00	0.97	0.98	223
2	0.88	0.98	0.93	190
accuracy			0.95	640
macro avg	0.95	0.96	0.95	640
weighted avg	0.96	0.95	0.96	640



Results are similar to those obtained by simple models alone. Let's see if it can provide better results on the 3D dataset:

	precision	recall	f1-score	support
0	0.99	0.88	0.93	192
1	0.98	0.96	0.97	206
2	0.80	0.96	0.87	135
accuracy			0.93	533
macro avg	0.92	0.93	0.92	533
weighted avg	0.94	0.93	0.93	533

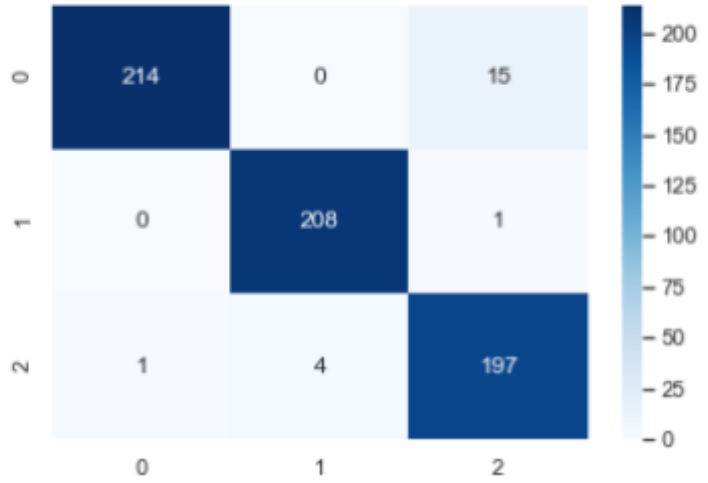
When used on a low number of features, the voting classifier provides significantly better results than simple models alone.

2.3) Ensemble model and boosting

Since there's a number of features, a random forest might be interesting to train on this dataset, while pruning trees and applying a maximum feature to consider at each split.

2.3.1) Random Forest

	precision	recall	f1-score	support
0	1.00	0.93	0.96	229
1	0.98	1.00	0.99	209
2	0.92	0.98	0.95	202
accuracy			0.97	640
macro avg	0.97	0.97	0.97	640
weighted avg	0.97	0.97	0.97	640

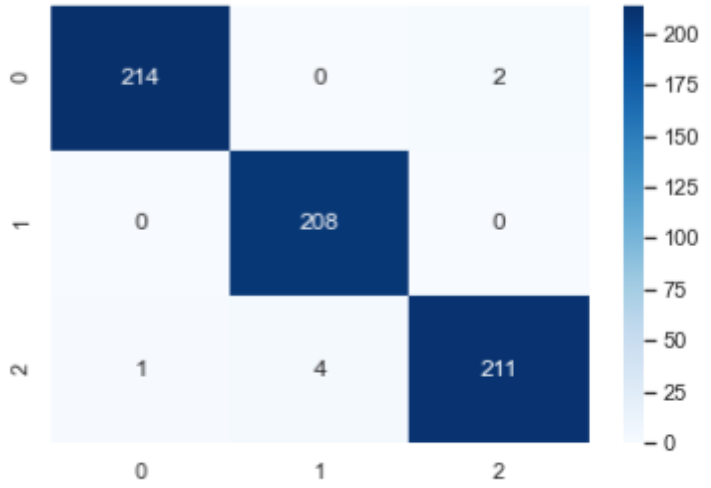


The random forest using only 10 features max for each split provided better results than the 4 simple models combined together.

2.3.2) XGBoost

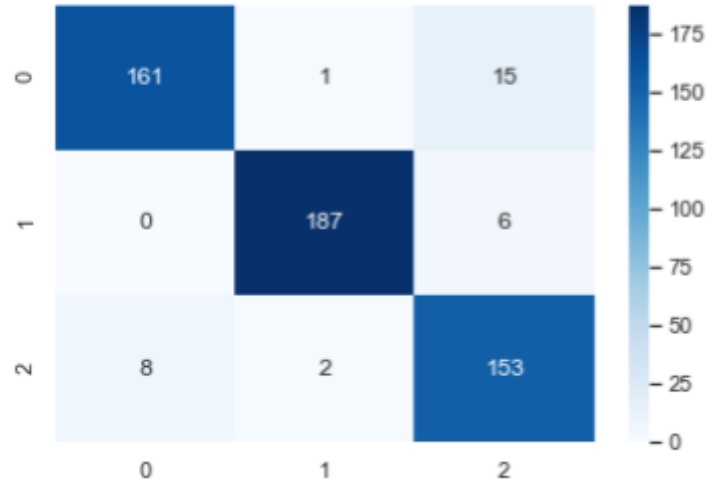
Gradient boosting and especially XGBoost has been known for quite some time now as a very efficient model.
Let's see the results obtained here with this model :

	precision	recall	f1-score	support
0	1.00	0.99	0.99	216
1	0.98	1.00	0.99	208
2	0.99	0.98	0.98	216
accuracy			0.99	640
macro avg	0.99	0.99	0.99	640
weighted avg	0.99	0.99	0.99	640



The results are close to perfect, making XGBoost by far the best model in term of prediction for this study.
Using only 3D dataset :

	precision	recall	f1-score	support
0	0.95	0.91	0.93	177
1	0.98	0.97	0.98	193
2	0.88	0.94	0.91	163
accuracy			0.94	533
macro avg	0.94	0.94	0.94	533
weighted avg	0.94	0.94	0.94	533



Even when dramatically reducing the number of features, the XGboost model performed surprisingly well, better than the voting classifier made of 4 different models.

Conclusion

The dataset used here had been super qualitatively processed previously, allowing one to obtain sky-high results even with simple algorithms. Moreover, the dataset has few rows which allow the user to train complex algorithms without having to think about training time, which should be taken into account when dealing with a huge dataset.

In order to tackle the dataset size, t-SNE and PCA offer dimensionality reduction that can be used in order to pick the best model.

In this precise paper, XGBoost alone performed the best, and its results were so high that nothing more was needed. However it is still possible to train several complex algorithms and combine them through a voting classifier. It would have probably been needed if the dataset had not been processed that well before.