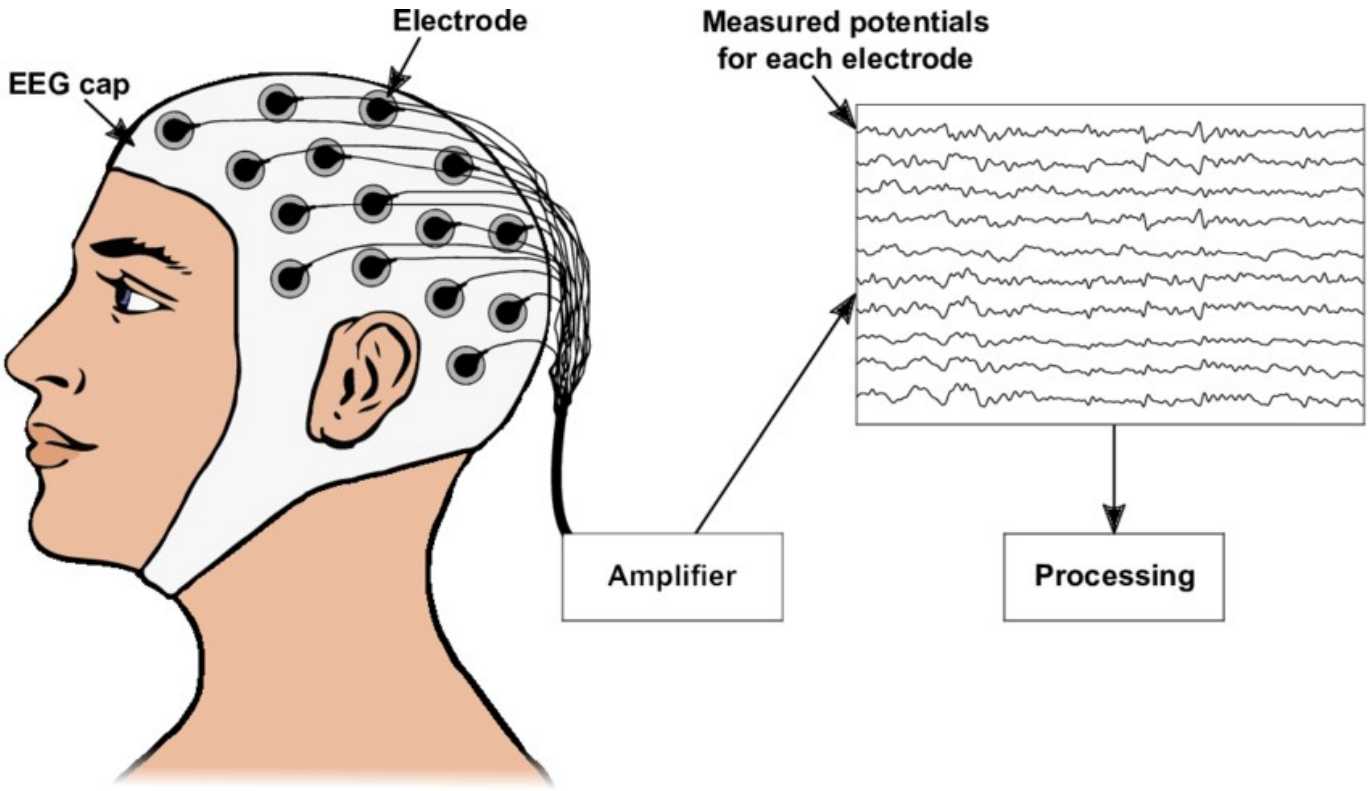
Aspiring Engineer: Catto Alex

Capstone Project:

Select and develop a project on your own design

Confused Person EGG Brainwave



Using encephalogram (EGG) brainwaves to classify a person state of confusion

Domain Background

Our field of research is the EEG (electroencephalography) analysis. This methodology measures the brain activity by detecting ‘”brainwaves “, or rather, electrical signals emitted continuously by the brain itself.

References

Ariely, Dan, and Gregory S. Berns. "Neuromarketing: the hope and hype of neuroimaging in business." *Nature reviews neuroscience* 11.4 (2010): 284.

Barnett, Samuel B., and Moran Cerf. "A ticket for your thoughts: Method for predicting content recall and sales using neural similarity of moviegoers." *Journal of Consumer Research* 44.1 (2017): 160-181.

Boksem, Maarten AS, and Ale Smidts. "Brain responses to movie trailers predict individual preferences for movies and their population-wide commercial success." *Journal of Marketing Research* 52.4 (2015): 482-492.

Christoforou, Christoforos, et al. "Your Brain on the Movies: A Computational Approach for Predicting Box-office Performance from Viewer’s Brain Responses to Movie Trailers." *Frontiers in neuroinformatics* 11 (2017): 72.

Dmochowski, Jacek P., et al. "Audience preferences are predicted by temporal reliability of neural processing." *Nature communications* 5 (2014): 4567.

Halchenko, Yaroslav O., and Michael Hanke. "Advancing Neuroimaging Research with Predictive Multivariate Pattern Analysis (MVPA)." (2010).

Harmon-Jones, Eddie, et al. "The role of asymmetric frontal cortical activity in emotion-related phenomena: A review and update." *Biological psychology* 84.3 (2010): 451-462.

Krugman, Herbert E. "Brain wave measures of media involvement." *Journal of Advertising Research* 11.1 (1971): 3-9.

Luck, Steven J., and Emily S. Kappenman, eds. *The Oxford handbook of event-related potential components*. Oxford university press, 2011.

Ma, Qingguo, et al. "P300 and categorization in brand extension." *Neuroscience letters* 431.1 (2008): 57-61.

Ramsøy, Thomas Z., et al. "Frontal Brain Asymmetry and Willingness to Pay." *Frontiers in neuroscience* 12 (2018): 138.

Ravaja, Niklas, et al. "Predicting purchase decision: The role of hemispheric asymmetry over the frontal cortex." *Journal of Neuroscience, Psychology, and Economics* 6.1 (2013): 1.

Telpaz, Ariel, et al. "Using EEG to predict consumers' future choices." *Journal of Marketing Research* 52.4 (2015): 511-529.

Neuromarketers scientists are investing a lot of time and resources in this field of research. This analysis is a way to understand the human behavior, and, what happens in our brain when it communicates and synchronizes activities across different anatomical regions. The cognitive process is defined by the variation of these signals, both in humans and animals. This knowledge may be applied for many technological solutions able to detect the will of a person and to process it in order to transmit that information. In recent years, EEG equipment has become more inexpensive, portable, and wireless, opening up new possibilities for mobile, in-store, and virtual reality studies. New statistical and machine-learning techniques are starting to be used to decode and interpret the EEG signal at the level of the full brain, portending many new and original findings.

Brainwaves can be measured with electrodes placed on the scalp and connected to a signal amplifier. There are three types of EGG measurement:

- Brainwave Frequency Analysis
- Hemispheric Asymmetry Analysis (an application of Frequency Analysis)
- Event-Related Potential Analysis.

Brainwave signals emitted by the brain have frequency characteristics. Electrical frequency is measured in hertz (cycles per second) . The most common frequency is 10 Hz (10 cycles per second). Frequencies change due to different mental states, and also vary over time and across different parts of the brain.

Brainwave frequencies have been classified in these groups:

- *"Delta"* less than 4 Hz, dominant frequency in dreamless sleep;
- *"Theta"* 4-8 Hz, associated with internally focused processing, such as memory activation and conscious concentration;
- *"Alpha"* 8-12 Hz, the brain's "default" frequency, dominant when the brain is in a relaxed state and suppressed under the influence of attention;
- *"Beta"* 13-30 Hz, associated with alertness, active attention, and reward expectation,
- *"Gamma"* greater than 30 Hz, associated with information processing, learning, and emotional processing.

In order to measure brainwave frequencies, two metrics are commonly used :

- *"power"* measures the activity at a particular frequency, over a period of time;
- *"coherence"* measures the correlation of frequencies across different parts of the brain.

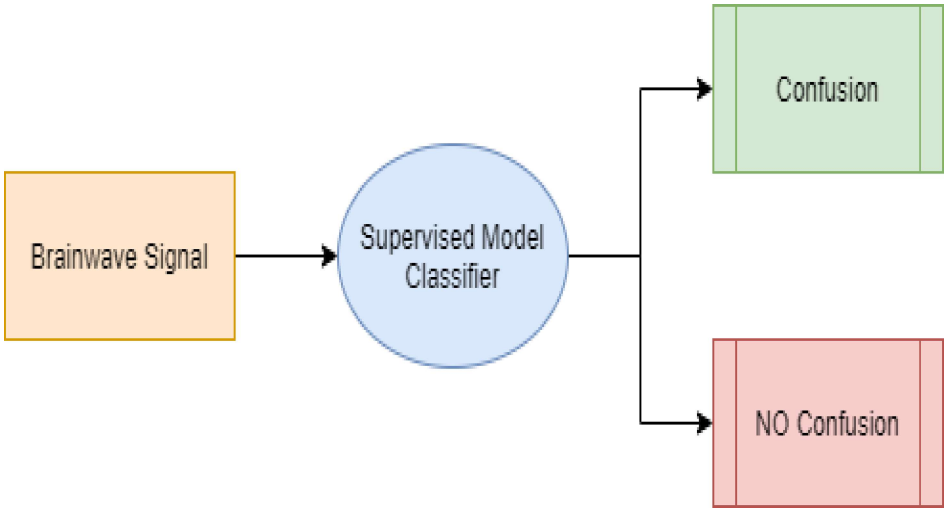
Problem Statement

The purpose of this project is the trial of developing some automation able to analyze a brainwave and to classify it with an accuracy of at least 65/70%, in order to understand if the person on which the signal has been extracted was in a mental state of confusion at the moment.

First, we want to understand if there are some patterns in the brainwaves signals able to correlate different frequencies to our specific mental state research. What are the frequencies that affect the confusion state the most? Which are the frequencies that affect the confusion state the most?

Furthermore, we want to build a classifier able to define if a brainwave signal sample owns to a confused person or not. Our question is "Is this signal a confusion mental state?"

In order to do this, we are going to use unsupervised and supervised Machine Learning techniques.



This application is able to detect if a person is in front of something new or difficult or unknown. This detection should be used as part of a modern "Truth Machine" or for computerized auxiliary systems able to understand when you need for more explanations (such as the fanta-scientific IronMan intelligent armor or StarWars protocol robot) or for user experience improvement automation.

Datasets and Inputs

In order to solve our problem, we need for samples of brainwaves labeled with a state indication of confusion. We will use a dataset of samples collected by a group of students and released for public access on Kaggle at the URL https://www.kaggle.com/wanghaohan/confused-eeg.

This data has been collected by submitting a test to 10 college students. The students was asked to wear the EGG signal detector while watching some MOOC video clips (Massive Open Online Courses). These videos have been selected according to students knowledge. Everyone had to watch for both a simple subject video and an unknown/difficult one. Furthermore, the videos expected to be muddler have been chopped with the purpose of showing the middle of the topic explanation: by this way, the subject should have been much more confusing.

The EGG cap was made of a single-channel wireless MindSet able to measure the activity over the frontal lobe: the voltage between an electrode resting on the forehead and two electrodes (one ground and one reference) each in contact with an ear. Every recording was two minutes long.

Every student, at the end of the video clip, rated his confusion level. According to this, a label (confused: yes or not) has been created and attached to the signal recording. Every signal data has been enriched with the indication of the student ownership (with his age and nationality data) and the video reference.

The data have been structured in this way:

- Every signal has been reduced to a one minute shot;
- Every minute of recording has been sampled every 0.5 seconds, thus every signal is represented in more than 120 rows.

The data representation is structured in different files:

- EEG_data.csv: Contains the EEG data recorded from 10 students, with features
    - SubjectID, the student identifier code
    - VideoID, the watched videoclip identifier code
    - 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
    - Alpha1, Lower 8-11 Hz of power spectrum
    - Alpha2, Higher 8-11 Hz of power spectrum
    - Beta1, Lower 12-29 Hz of power spectrum
    - Beta2, Higher 12-29 Hz of power spectrum)
    - Gamma1, Lower 30-100 Hz of power spectrum
    - Gamma2, Higher 30-100 Hz of power spectrum
    - Predefinedlabel, whether the subject is expected to be confused
    - user-definedlabeln, whether the subject is actually confused

  This file contains 12811 rows of samples for 15 features.

- demographic.csv: Contains demographic information for each student
  - SubjectID, the student identifier code
  - Age, the student Age
  - Ethnicity, the student Ethnicity (TYPE string)
  - Gender, the student gender (M/F)

  This file contains 10 rows, as the 10 students submitted to the test
- video data : Each video lasts roughly two-minute long, we remove the first 30 seconds and last 30 seconds, only collect the EEG data during the middle 1 minute.

Solution Statement

In order to solve this problem, it is necessary to apply some unsupervised and supervised Machine Learning techniques.

Exploration is needed both for data and models, thus a Jupyter notebook will be used. In this notebook data will be explored in order to find the best pre-processing solution. Moreover, different models will be tuned with different hyperparameters and will be compared one to each other. The best solution will be detected in the notebook.

After that, a Python application will be designed with the pre-processing data steps, the chosen model training process and the prediction module:

- DataManagement.py is a module containing data processing functions;
- AppParametersManagement.py is a module with command line parameters parsing helpers;
- Train.py is an application able to train the model and to save it on a dump file;
- Predict.py is an application able to classify brainwave samples recorded in a file.

Benchmark Model

The author of the dataset collection tried to apply Machine Learning binary classification on the dataset.

I attach here their Benchmark, as written on Kaggle:

"*This dataset is an extremely challenging data set to perform binary classification. Here are some recent classification results for reference:*

- *SVM: 67.2%*
- *Bi-LSTM (Ni et al 2017): 73.3%*
- *CF-Bi-LSTM (Wang et al 2018): 75.0%*"

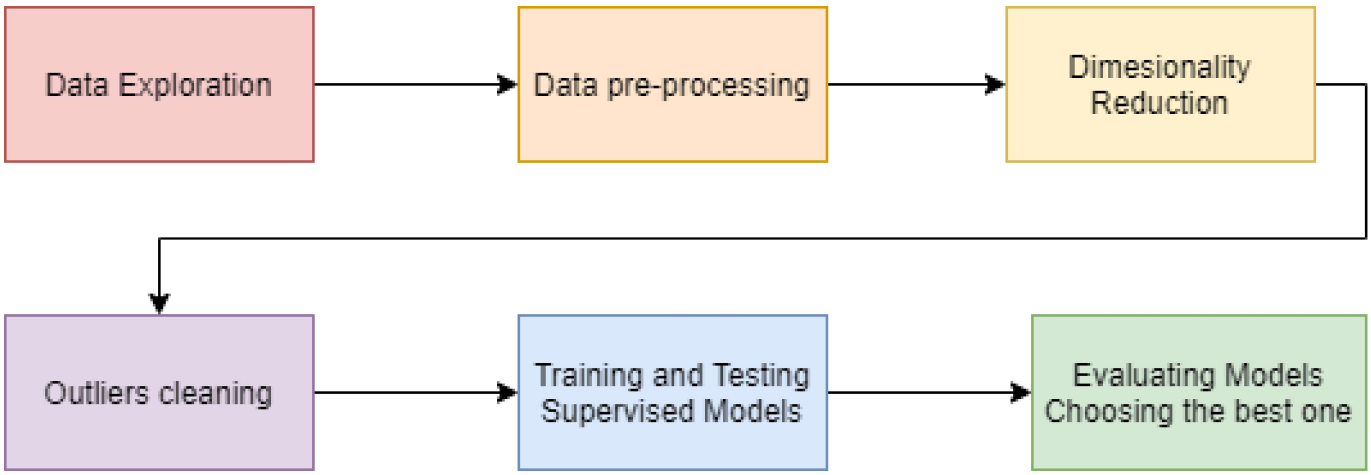
These values of model accuracy may be used for benchmark comparison between the model of this project and the already existing one. We expect to get a model with accuracy around 70%.

Evaluation Metrics

The Benchmark model is evaluated with accuracy score. For this reason, the model of this project will be evaluated in the same way. Moreover, Precision and Recall metrics will be evaluated, in order to understand the False Positives and False Negatives detection. It should be useful evaluate the model in its computation speed: time for the training process and for the prediction.

Project Design

In order to solve this problem, it is necessary to apply some unsupervised and supervised Machine Learning techniques.



First, data will be studied, in order to explore features variance and patterns on the data. By this way, it may be possible to identify useless or outliers features, and then, as a consequence, applying dimension reduction. While conducting this data exploration, it is possible to understand if features such as gender and age are relevant or not. Moreover, a samples grouping should be able to find eventual outlier students. If the clustering process detects a distance from one or a few students respect to the others: they should be considered as outliers, it is necessary to delete them. In order to do this, data will be scaled, and eventually imputed. Furthermore, categorical features will be one-hot encoded.

After data exploration and pre-processing, a supervised learning model has to be trained and tested. In order to do this, four models will be tuned with different hyperparameters (using grid search):

- **Decision Trees**

  Decision Trees model is used for classification problems. This algorithm is able to analyse data features and to understand how much information each of these fetaures can give to us, in order to make our prediction. The model can make really accurate predictions, because of each feature can be combined with every other feature according to its specific information gain. This type of model has also a bad behavior. It tends to overfit a lot, thus, we have to use it with high attention or use it with any ensemble method.

- Ensemble Methods

  Ensemble methods are able to solve classification problems combining different classification models. It can be used for real world applications, especially when data is not linearly separable. We can make a prediction using different models combined with an ensemble method, such as Ada Boost or Bagging or Random Forest. Ensemble methods perform very well when data is not linearly separable. Furthermore, they are able to create models fitting data well because of their ability of finding a good compromise between bias and variance.

  We will use both **AdaBoost** and **XGBoost** (Extreme Gradient Boosting).

- **Support Vector Machine**

  Support Vector Machine is able to make good classification predictions, defining models with a well-fitted boundary. This boundary is so good because of the use of the margin. This model has a good flexibility because of its parameters. This alorithm is able to work on different types of data, in fact, it has three kernels: linear, polynomial and Rbf. Moreover, an SVM model can give us the ability of tuning it in order to decide how much the model has to be precise: with the C parameter we can define how much weight we want to assign to the Classification error respect to the margin error.

The best model will be chosen, evaluating:

- Testing accuracy;
- Training and Testing Fscore visualized in a graph with "Learning Curves" technique (we don't want to overfit or underfit);
- Execution time (How much is the algorithm computationally expensive?).

LET'S START: Data Loading and Exploration

At first, it is necessary to **load the dataset**: EGG brainwaves samples and people demographic info.

Data is stored in the files "data/EEG_data.csv", "data/datasets_106_24522_demographic_info.csv". The format of these two files is CSV. Every file has an header line with feature names. Every value is divided by the comma (',') separator.

In the previous section, a detailed view of the data structure has been presented, but we want to visualize it. Which are the features? How data is structured? How large is the dataset? What types of features are there?

Here's a print of the **datasets shapes**:

```
EGG Brainwave samples shape: (12811, 15) People info: (10, 4)
```

We are able to display the first rows of each of the datasets:

EGG Brainwave samples:

	SubjectID	VideoID	Attention	Mediation	Raw	Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2	predefinedlabel	use definedlabe
0	0.0	0.0	56.0	43.0	278.0	301963.0	90612.0	33735.0	23991.0	27946.0	45097.0	33228.0	8293.0	0.0	0
1	0.0	0.0	40.0	35.0	-50.0	73787.0	28083.0	1439.0	2240.0	2746.0	3687.0	5293.0	2740.0	0.0	0
2	0.0	0.0	47.0	48.0	101.0	758353.0	383745.0	201999.0	62107.0	36293.0	130536.0	57243.0	25354.0	0.0	0
3	0.0	0.0	47.0	57.0	-5.0	2012240.0	129350.0	61236.0	17084.0	11488.0	62462.0	49960.0	33932.0	0.0	0
4	0.0	0.0	44.0	53.0	-8.0	1005145.0	354328.0	37102.0	88881.0	45307.0	99603.0	44790.0	29749.0	0.0	0

Students Demographic Info:

	subject ID	age	ethnicity	gender
0	0	25	Han Chinese	M
1	1	24	Han Chinese	M
2	2	31	English	M
3	3	28	Han Chinese	F
4	4	24	Bengali	M

Every sample is associated to the person from who it has been detected. People data presents age, ethnicity and Sex. These two datasets are correlated by a feature 'SubjectID', a unique identifier for the 10 students. Its values are the range between 0 and 9 (Student 1 – 10).

As we can see, there are 12811 EGG samples and 10 people. Every feature is a numerical value in the samples dataset. In the demographic one, instead, there are **two categorical features: Sex and Ethnicity**. Sex is a binary classification feature: 'M' stands for "Male" and 'F' for "Female". Ethnicity, instead, has more values. These values are strings referring to the student origin.

We don't want to study the two datasets separately, thus, we have to join them. In order to do this, we have to consider their relational correlation: the samples dataset contain an "External Key" related to the people dataset "Primary Key". The key feature (external/internal) is "SubjectID" for both, as seen above, yet. The joined df is displayed below:

Mediation	Raw	Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2	predefinedlabel	user-definedlabeln	subject ID	age	ethnicity	gender
43.0	278.0	301963.0	90612.0	33735.0	23991.0	27946.0	45097.0	33228.0	8293.0	0.0	0.0	0	25	Han Chinese	M
35.0	-50.0	73787.0	28083.0	1439.0	2240.0	2746.0	3687.0	5293.0	2740.0	0.0	0.0	0	25	Han Chinese	M
48.0	101.0	758353.0	383745.0	201999.0	62107.0	36293.0	130536.0	57243.0	25354.0	0.0	0.0	0	25	Han Chinese	M
57.0	-5.0	2012240.0	129350.0	61236.0	17084.0	11488.0	62462.0	49960.0	33932.0	0.0	0.0	0	25	Han Chinese	M
53.0	-8.0	1005145.0	354328.0	37102.0	88881.0	45307.0	99603.0	44790.0	29749.0	0.0	0.0	0	25	Han Chinese	M

The new complete dataset has 18 features: SubjectID, VideoID, Attention , Mediation, Raw, Delta, Theta, Alpha1, Alpha2, Beta1, Beta2, Gamma1, Gamma2, predefinedlabel, user-definedlabeln, age, ethnicity, gender. The number of rows in the dataset is 12811, exactly as the number of rows in the previous samples dataset. **New shape: (12811, 18).**

As we want to use this dataset for classification, we must be sure of **avoiding samples with missing values**. The algorithms requires that every data point has acceptable values. For this reason, now we look at missing values. If there are columns with a really high number of missing values, we should drop them. If there is an acceptable number of missing values, instead, **we can just impute the dataset** with some imputation algorithm. For imputation we can use the 'mean value' technique that sets to missing values the mean value for the feature. Or we could use 'K-Nearest Neighbours imputation': this algorithm is great because it assigns to the point a value based on the k-nearest points values in the n-dimensional space.

Let's display a count of the missing values for each feature: VideoID 0, Attention 0, Mediation 0, Raw 0, Delta 0, Theta 0, Alpha1 0, Alpha2 0, Beta1 0, Beta2 0, Gamma1 0, Gamma2 0, predefinedlabel 0, user-definedlabeln 0, subject ID 0, age 0, ethnicity 0, gender 0.

There are no missing values in the dataset: that's really great.

Feature Transformation

Feature transformation is a necessary step in the pre-processing work-flow. The algorithms require that data is in a specific format (for example missing values are not allowed, as we saw before). Moreover, some feature properties could affect the result of the algorithm (For example we should not do PCA without scaling features). The result of the supervised algorithm should be affected by features that are useless for the problem purpose. These useless features increase the computationally expense, too. For these reasons, we have to select and transform the features.

First of all, according to the algorithms requirements, we must be sure of holding **numerical features only**. Thus, **we have to encode categorical features**. As we noted above, there are two categorical features: Sex and Ethnicity. The first is binary (M/F) and as a consequence we can just map the two values to 0s and 1s. Ethnicity, instead, has more values. In order to solve this, we use the "**one-hot encoding**" technique. Every feature is splitted into more dummy columns. Every new column has values in the range of 0s and 1s.

gender	ethnicity_Bengali	ethnicity_English	ethnicity_Han Chinese
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1

Here's an example of a few rows after the encoding. Gender (Sex) has one column only, with values of 0 and 1 (Male or Female). Ethnicity, instead, produced more columns labeled with the rule 'ethnicity_value'. Every of these columns has the value 0 if the raw wasn't filled up with the correlated value. Instead, if the column name refers to the original value, the row field is '1'.
After the encoding , in the field of Feature Transformation, we can do **Feature Selection/Extraction**.

There are two features that we certainly don't need for. The first is the 'VideoID'. This column gives an information that is correlated to the context of the experiment: it is not a valuable data of the EGG brainwave. For this reason, we drop it.

'predefinedlabel' is another useless feature: it indicates which confusion state was supposed to be detected, by the experiment conductor, before doing the test.

We need for the 'user-definedlabel', instead, because that's the label indicating if a signal is correlated to a confusion state. Thus, we drop 'predefinedlabel'.

What about SubjectID? This feature indicates the Identifier of the people who generated the detected brainwave. This is clearly unuseful for our purpose, but for now, let's hold it. In order to find outliers, we want to look at the data and to see if some students samples are distant from the other ones. For this reason, we will drop the feature in a second moment: after outlier analysis.

What about Sex, Age and Ethnicity? Are these features useful for our purpose? Does a brain work in similar or different ways on Male and Females? Does Age impact the brain activity? Ethnicity?

We can't know this, without an important background. **However, we can study feature variance by applying PCA: after that, we 'll be able to do a better evaluation.**

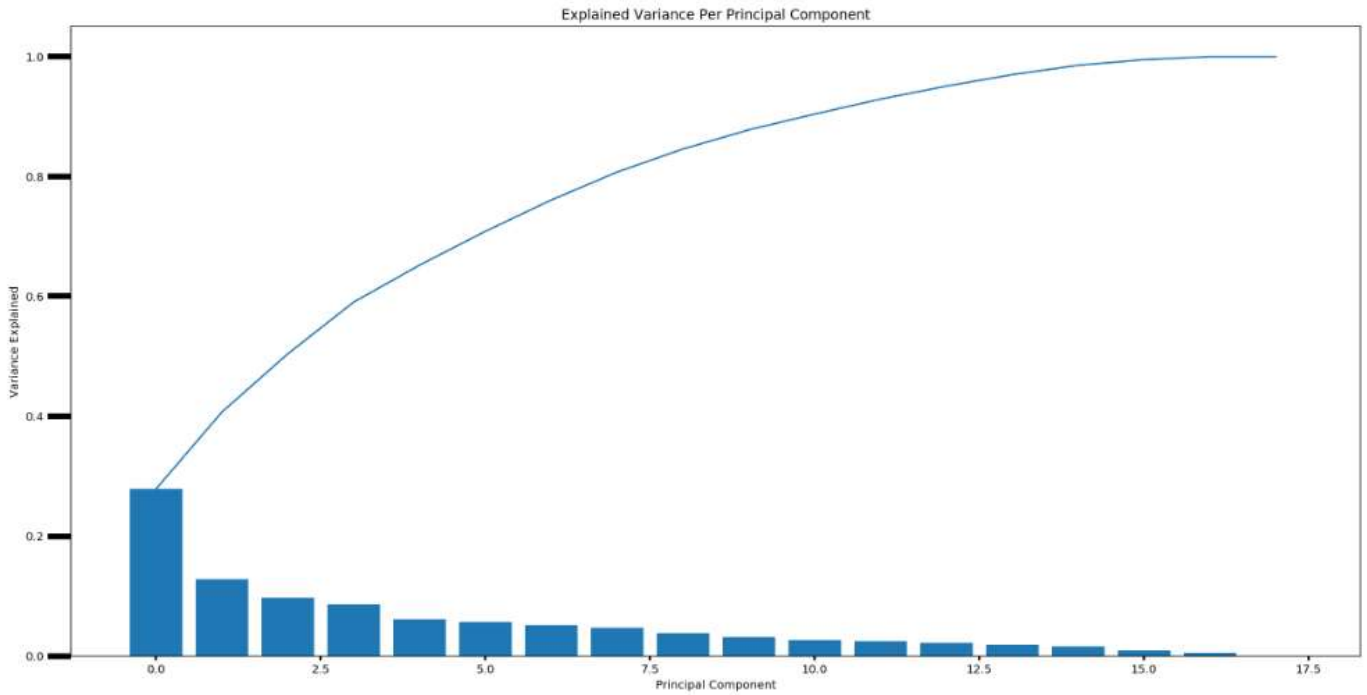
Before applying dimensionality reduction and unsupervised techniques to the data, **we need to perform feature scaling.** By this way, the principal component vectors are not influenced by the natural differences in scale for features. In order to do this, we use a Standard Scaling technique: features are standardized by removing the mean and scaling to the unit variance. The standard score of a sample x is calculated as

"z = (x - u) / s" where 'u' is the mean of the training samples , and 's' is the standard deviation of the training samples'.

Here we can look at a few rows after the StandardScaler transformation:

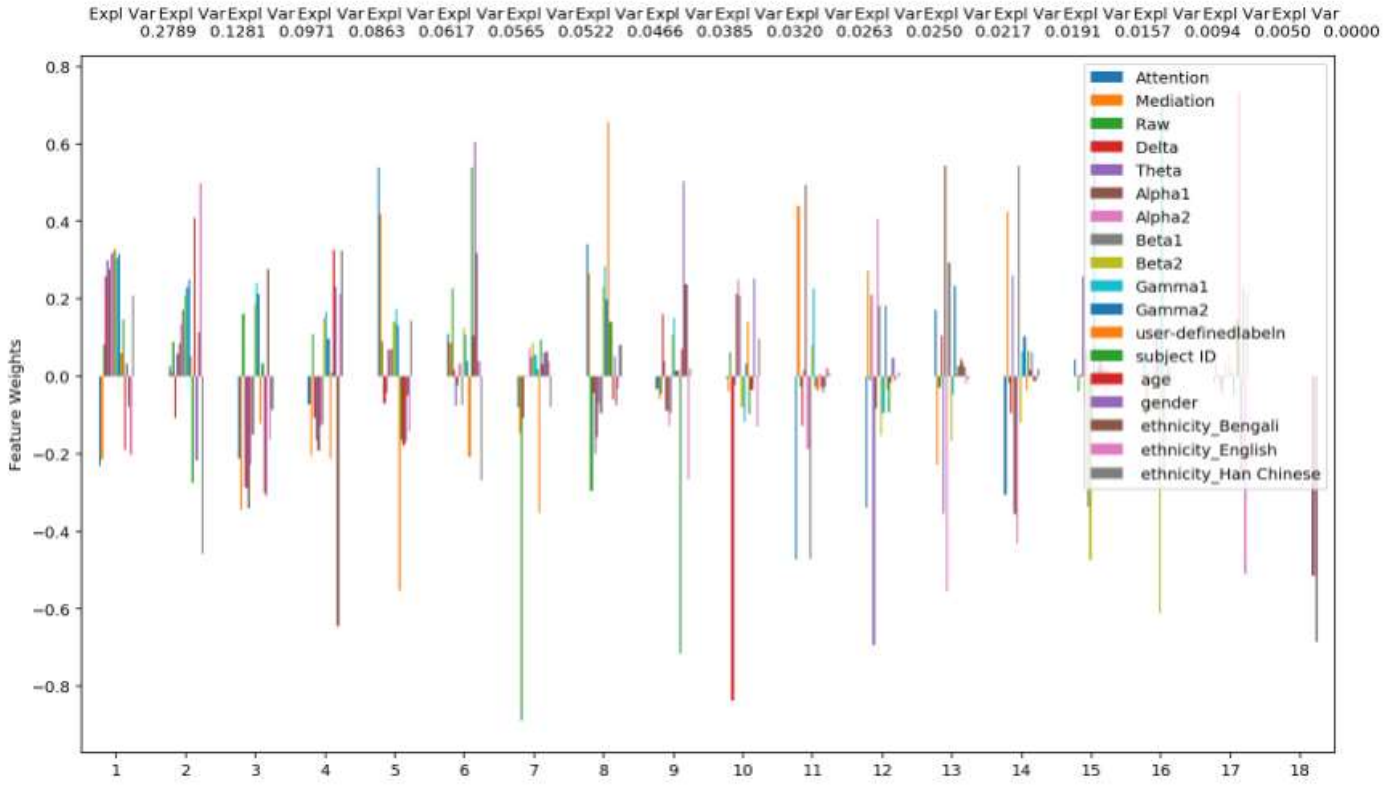
	Attention	Mediation	Raw	Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2	user-definedlabeln	subject ID	age
0	0.634334	-0.184623	0.355294	-0.476510	-0.317217	-0.105613	-0.157642	0.094523	0.087938	0.045544	-0.169923	-1.025539	-1.566138	-0.185918
1	-0.056750	-0.537745	-0.193295	-0.834378	-0.573352	-0.551518	-0.530654	-0.562100	-0.435821	-0.304417	-0.324028	-1.025539	-1.566138	-0.185918
2	0.245599	0.036078	0.059256	0.239285	0.883532	2.217577	0.496016	0.312017	1.168583	0.346396	0.303549	-1.025539	-1.566138	-0.185918
3	0.245599	0.433339	-0.118031	2.205862	-0.158536	0.274088	-0.276091	-0.334314	0.307573	0.255157	0.541603	-1.025539	-1.566138	-0.185918
4	0.116021	0.256779	-0.123049	0.626350	0.763033	-0.059126	0.955168	0.546890	0.777338	0.190389	0.425518	-1.025539	-1.566138	-0.185918

Once the dataset is scaled, it is possible to apply the PCA (principal component analysis) in order to detect feature variance. At the moment, we don't want to do a "feature extraction" by giving a number of combined features to the PCA transformation algorithm. We just want to explore the variance and the weights. For this reason, for now, we apply PCA without a 'n_components' parameter. We'll do it later, if the evaluation retains it necessary.



In the graph above you can see an histogram showing each of the 18 features. For each component there is a bar indicating the percentage of variance contribution. The graph y label (variance explained ratio) goes from 0 to 1 (0% to 100%). Furthermore, there is a continuous line able to show the increasing total amount of variance given by the sum of the components. This line increasing and this bar value decreasing is given by the order of the plots: the components are shown in descending order of variance weight, from the left to the right.

The first three components own almost the 60% of the total variance. The last 7 components own less than the 5% of the total variance. These last components are probably useless features. They could be the ones we were trying to understand: Age, Gender, Ethnicity. In order to solve this mistery: let's show another graph, with the visualization of the components weights.



In this graph above, we can see the feature variance weight for each of the components respect to the others. There are the 18 components. For every of them, there is a view of the other components variance correlation (the vectors). The legend positioned at the top-right of the graph shows the feature names in order of variance. Thus, the first at the top is the feature with the most variance and the one at the bottom is the one with the less.

We can note that the total amount of variance in the **data could be splitted in three groups**:

- The first three features: Attention, Mediation and Raw.

  These features are brainwaves parameters and they are really impactive on the dataset. As we discussed before, they own almost the 60% of the total amount.

- The Frequencies features: Delta, Theta, Alpha (1-2), Beta (1-2), Gamma (1-2).

  These features are the frequencies of the spectrum in the brainwaves: the real signal. They own a good amount of the percentage of variance.

- Demographic info and labels: user-definedlabeln, SubjectID, age, gender, ethnicity.

  These features are low impactive in the dataset. We had doubdts about their importance, and we can effectively see that our assumptions were founded.

According to this, we have to decide what to do with the features: Do we take them? Do we apply PCA with a restricted number of components? Do we just drop useless features?ì Well, the label is the feature we want to predict: we cannot drop it or combine it with other component vectors. The SubjectID is a feature that we want to hold, in order to do the outliers evaluation. The other components of the third group seem to be useless. For these reasons, we don't apply PCA again on the dataset but **we just drop the not relevant features: Age, Gender and Ethnicity.**

This analysis of the Feature Transformation step is highlighting one thing: In order to do the brainwave classification, we don't need for demographic info. For this reason, it will not be necessary for our application to load demographic data, merging it to the EGG one and finally encoding categorical features.

As a consequence, **the final pre-processing step will be really easy and cheap**.

Searching for outliers

The dataset might have some outlier students. Every student has generated many samples. It is possible that some students was in a mental state not good for the experiment. **This may have corrupted the dataset**. Now, we look at the data in order to find students with feature showing very distant median values from the other ones, both they're confused or not.

At first, we have to group data according to the Subject ID. After taht, we can display **the mean value of each feature** for each student.

subject ID	Attention	Mediation	Raw	Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2	user-definedlabeln
-1.566138	0.463526	0.137041	-0.049401	-0.079091	-0.147780	-0.164685	-0.200454	-0.167752	-0.006978	0.089025	0.010623	-1.025539
-1.217130	0.480852	0.634896	-0.056641	-0.430281	-0.309199	-0.229508	-0.230934	-0.135661	-0.340621	-0.293397	-0.275394	-1.025539
-0.868121	0.483125	0.591083	-0.056485	-0.916609	-0.625706	-0.482017	-0.450010	-0.481244	-0.408168	-0.335563	-0.337569	-1.025539
-0.519113	-0.053047	-0.297396	-0.053774	0.123350	0.007281	-0.005710	-0.082311	-0.158214	-0.138618	-0.211915	-0.288900	-1.025539
-0.170104	0.433945	0.490279	-0.037489	-0.487516	-0.301457	-0.251043	-0.251699	-0.215892	-0.310456	-0.246533	-0.182426	-1.025539
0.178904	0.308163	0.281390	-0.045177	-0.100387	-0.011215	-0.016723	-0.035443	-0.237937	-0.224155	-0.209239	-0.334614	-1.025539
0.527912	-1.784458	-2.082651	0.472901	0.126036	0.145062	0.114927	0.446730	0.679254	1.549016	1.560742	1.333831	-1.025539
0.876921	0.571452	0.183816	-0.049661	-0.373972	-0.343069	-0.295101	-0.294702	-0.167953	-0.123406	-0.119183	0.006332	-1.025539
1.225929	0.415664	0.230156	-0.047508	0.178774	0.086919	-0.050603	-0.128893	-0.125373	-0.250992	-0.211130	-0.158485	-1.025539
1.574937	-0.019229	0.087551	-0.067893	0.324043	-0.075628	0.096449	0.103728	-0.185403	0.058684	-0.134313	-0.312958	-1.025539

We can note that students 3rd (-0.868121 in scaled dataset) and 7th (0.527912 in scaled dataset) have very distant mean values for frequencies features, respect to the other students (indipendently of the confused or not confused state). Let's look at a few examples, just for understanding what I mean:

- the 3rd student has for 'Theta' a median of -0.6 when the others are between –0.0/-0.3
- the 3rd student has for 'Beta2' a median of -0.4 when the others are between –0.05/-0.3
- the 7th student has for 'Attention' median of 1.7 when the others are about 0.4
- the 7th student has for 'Mediation' a median of -2.08 when the others are about 0.1/0.5

We want to reject these samples. After that, we can drop the SubjectID column.

Here, the preprocessing flow has finished.

Shuffling and Splitting data

The data is pre-processed and ready for being used in order to achieve our purpose. We'll have to build a supervised model which is supposed to be trained. In order to train and evaluate the models: **we need for training and testing data**. In this case we don't use a Cross Validation subset: we'll just test different trained models. The data samples are shuffled and then splitted in training and testing data: 20% for testing and 80% for training. Now, Training set has 8201 samples, Testing set has 2051 samples.

Model Training, Tuning, Evaluation and Selection

In order to achieve our purpose, **we must build a supervised classifier**.

In order to do this, we will train four different models and then we'll evaluate them. The selected algorithms are the ones we discussed in a previous section: DecisionTrees, AdaBoost, XGBoost, SVM.

After their evaluation, we will be able to select the best suitable model for our problem, and finally use it in the python application.

Every model will be tuned using different hyperparameters, through the GridSearch tool.

Here's a view of the hyperparameters combinations:

- DecisionTreeClassifier
  - 'max_depth': list(range(4, 20, 2))
  - 'min_samples_split': list(range(2, 15, 2))
  - 'min_samples_leaf': list(range(1, 20, 2))
- AdaBoostClassifier
  - 'algorithm':['SAMME','SAMME.R']
  - 'n_estimators': [10, 40, 60, 100, 120, 130, 140]
- SVC (Support Vector Machine Classifier)
  - {'kernel': ['rbf']
  - C': list(np.arange(0.5, 1.5, 0.1))
  - 'gamma': ['scale', 'auto']
- XGBClassifier
  - 'base_score': list(np.arange(0.2, 0.5, 0.1))
  - 'n_estimators': [10, 40, 60, 100, 120, 130, 140]
  - 'objective': ['binary:logistic']}

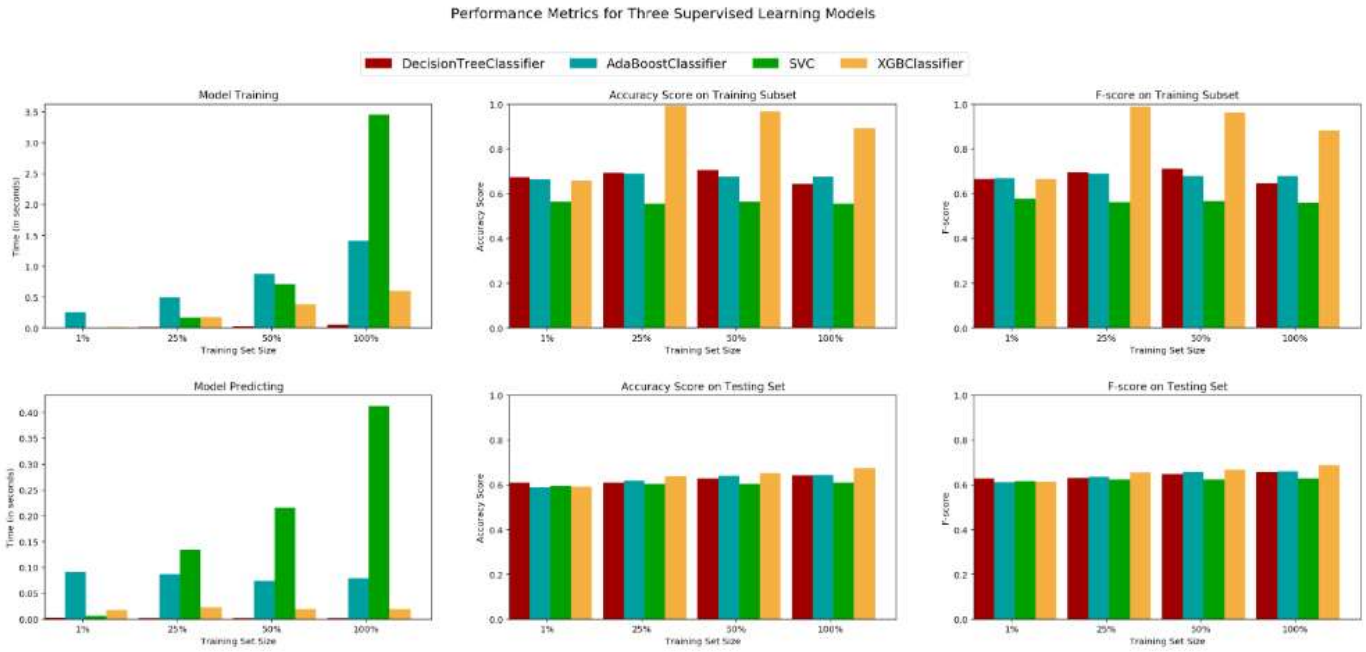
We create a training and tuning pipeline where **every model is trained and tuned**:

- The learner is fitted to the sampled training data, and the training time is recorded;
- The best estimator from grid search is detected;
- Predictions are performed on the test data, and also on the first 300 training points;
- The total prediction time is recorded;
- Accuracy score is calculated both for the training set and testing set.
- F-score is Calculated both for the training set and testing set.

Now, we have the best estimator for every algorithm we decided to use.

- **DecisionTrees BestParam**s: {'max_depth': 6,'min_samples_leaf': 7,'min_samples_split': 2}
- **AdaBoost BestParams**: {'algorithm': 'SAMME.R', 'n_estimators': 120}
- **SVM BestParams**: {'C': 0.7, 'gamma': 'scale', 'kernel': 'rbf'}
- **XGBoost BestParams**: {'base_score': 0.4000000000000001,'n_estimators': 60,'objective': 'binary:logistic'}

It is time to evaluate the models. It is a good idea to train the models on different percentage of data and calculate metrics both on testing (Cross-Validation) and training data. By this way, it is possible to plot the metrics on a graph of models comparison where we can see the metrics behavior according to the training set size. The trend of the values during the dataset size increasing, can give a lot of information about the model complexity and its tendency to underfitting/overfitting. Looking at these results, we'll be able to detect the best model, according to: Performance, Accuracy, Bias-Variance balance.



The graphs above shows exactly what we need. There are three columns of graphs: every column correspond to a metric we want to evaluate. The one in the left displays the training time. The one in the middle displays the accuracy score. The one on the right displays the F-score with beta=0.5. Every metric has been calculate both for training and testing set. For the training set prediction were used the first 300 points. Every graph is divided into four groups of bars. Every group is a representation of a subset of the data: 1%, 25%, 50%, 100% of the total amount. Every group of bars is composed by the scores of the four algorithm we're evaluating. There's a legend with the colors associated to each of the algorithms.

Looking at these graphs, we can clearly identify the models with the highest values in the scores:

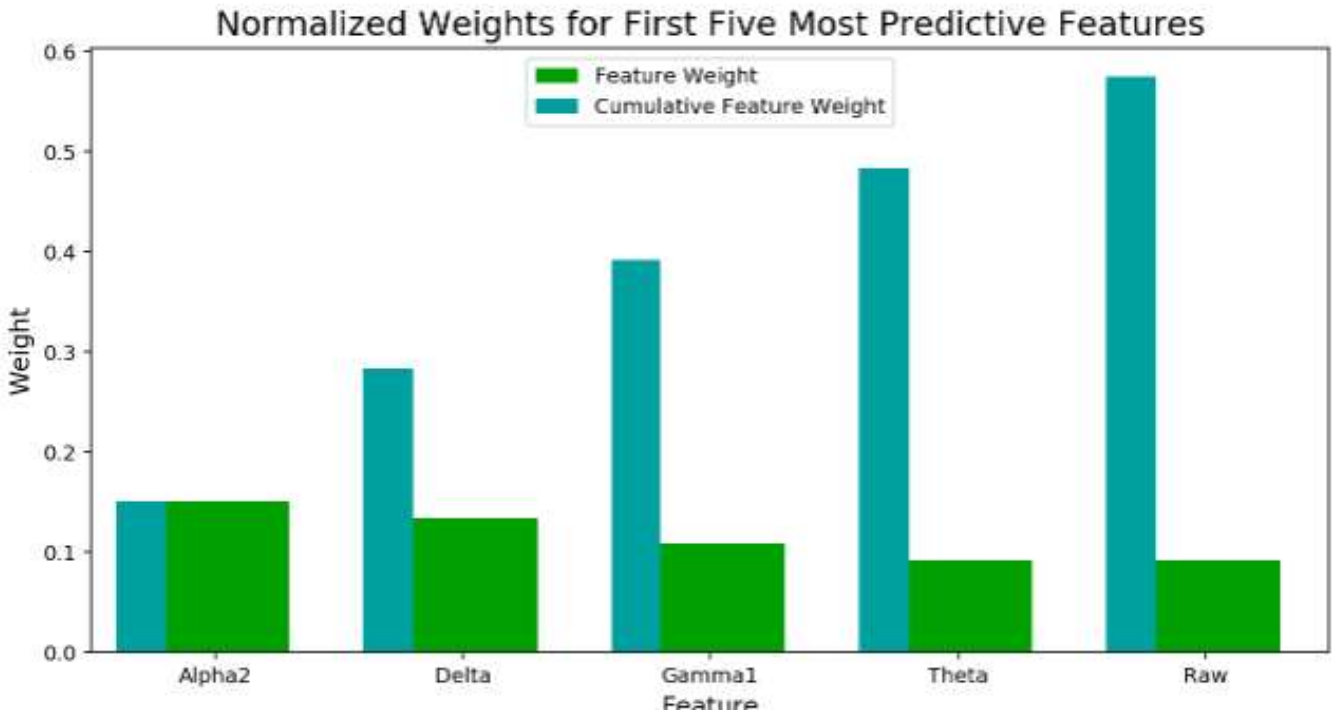
- The **execution-time** metric highlights a model that seems to be much more computationally expensive than the others: SVM. In the other hand, there is an algorithm resulting really fast: DecisionTrees. AdaBoost and XGBoost have anyway a good execution-time, thus, in case of their better performance in other metrics, we should not be much worry about the time comparison.
- If we want to look for models with a bad **bias-variance balance**, we have to look at the other metrics trends. An underfitting model shows the training and testing scores converging to a low value. **There aren't underfitting models here**. An overfitting model, instead, shows training and testing error diverging at the data size increasing. **There aren't overfitting models here**.
- **The Accuracy and FScore** metrics highlit that the XGBoost model is the one with the higher values. AdaBoost and DecisionTrees are really similar one to the other. SVM seems to be the worse model.

According to our metrics, we select the XGBoost model as the one we want to use, with an accuracy of about 70%. XGBoostClassifier: {'Accuracy': 0.674305216967333, 'f_score': 0.6878541076487251}

Feature Importance

One of the purposes of our project is understanding what features are most correlated to the brainwave confusion detection. When applying PCA to the data we could see what features had the higher variance. Now that we trained a supervised model, we can see how much any feature is relevant for the final classification and as a consequence which features are more relevant in the confusion state brainwave.

Here's a plot of the five most relevant features



Conclusions and Benchmark

The purpose of this project was the one of understanding what features affect the confusion mental state the most, and we wanted to build a model able to classify brainwaves at the scope of detecting confusion. We was able to display and explore both general feature variance in the dataset, and feature classification importance. In both cases we could note that the frequencies 'Theta' and 'Delta' are really high correlated to the state of confusion. 'Delta' is in the top 5 features necessary to detect the confusion state. **'Theta' is the first of the frequencies** for classification importance. When doing PCA, we saw that **both 'Theta' and 'Delta' have a lot of variance**.

The model we selected for classification is able to classify the brainwaves with an accuracy of about 70% (67.4%) . If we compare this result with the one of the data collectors, we can see how the model's accuracy is really near their SVM one (67.2%).

We wanted to understand if could be possible to find a better model without using NNs, in order to have a **really fast** prediction and a very **low training time** without the help of GPUs. We know that there are two better models: Bi-LSTM 73.3% and CF-Bi-LSTM 75.0%. These models have been created with RNNs architectures after a lot of experimentation by the data collectors.

Python Application

Now we have all we need for building a python application able to train a model and to predict different brainwaves. The application has two main access points:

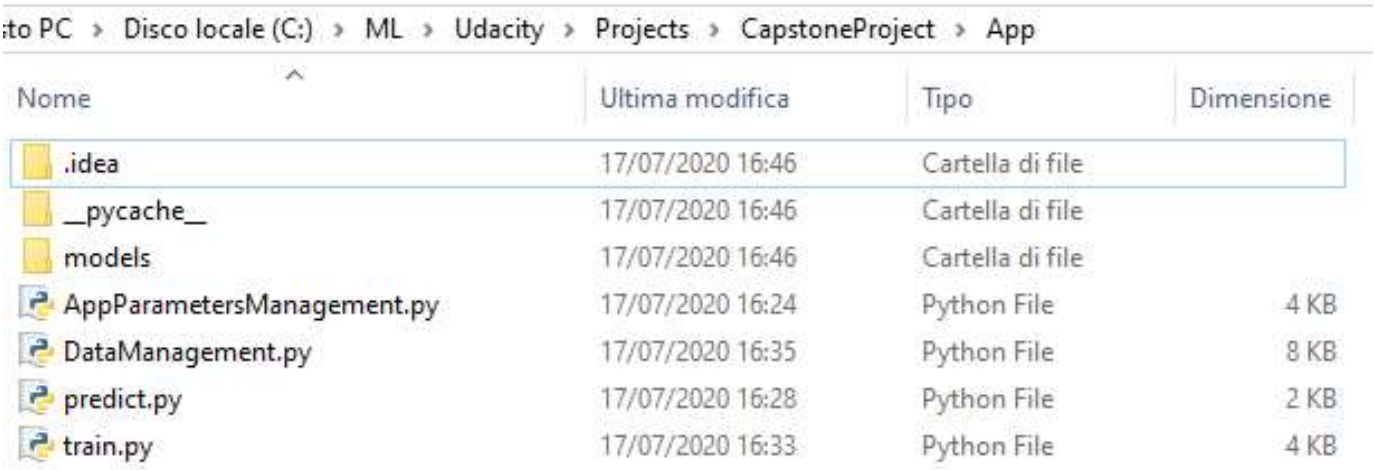
- Train.py

  This is a script able to train a model and to save the weights on a file. The script takes as an input the file path with the data to use for training and testing the model. At that point, it is able to train a model, according to the selected classifier. When the training process is done, the model weights are saved in a file, inside the 'models' folder. The file name is composed by the name of the classifier and by a string format of the file creation timestamp (unique name!). This file could be loaded in a second moment by other applications, in order to be used as a pre-trained model.

- Predict.py

  This is a runnable script able to classify a set of brainwaves. It takes as an input a file path where to find the pre-trained model to load. This model will be used for predictions. Furthermore, there are other inputs. It is necessary to define a file where to find the data to classify. This data has to be well formatted: every column in the training data must be present in this dataset. The irrelevant columns will be dropped. If this data has the label column, it will not be considered: the predictor will fill in the labels itself. At the end of given brainwaves classification, a new file is created by the application. This last file will contain the dataset to predict with the predicted labels. The name of the file is composed by the name of the original fil with the prefix 'predicted_' in addition.

Here's a view of the Application Folder:



As you can see, there are two additional files:

- AppParametersManagement.py

  This script is a module containing a class able to load application parameters and to parse them. Furthermore, every command line parameter has a pre-defined default value. Potentially, the applications could be run without specifying any parameter, using the default ones. This defaults have been set according to the current position of the scripts and the data files. Data, for example is

supposed to be in the '..\data' directory. The directory for saved models is supposed to be '\models' and so on.

Here'a list of the methods:

- get_by_line(param_name): This function gets a parameter by prompt line;
- models_dir(self): This method returns models_dir parameter;
- data_dir(self): This method returns data_dir parameter;
- outliers_students(self): This method returns outliers_students parameter;
- beta(self). This method returns F-score beta parameter;
- predictor_file_name(self): This method returns predictor_file_name parameter;
- topredict_file_name(self): This method returns topredict_file_name parameter;
- print_all(self).:This method prints parameters.

- DataManagement.py

  This script is a module containing a class able to load, save and pre-process data as required.

  Here'a list of the methods:

  - impute_dataset(df): This function imputes a dataset, replacing NaNs values with K-Nearest Neighbours median value;
  - remove_outliers(EEG_data_df, outlier_subjects): This function filters the dataframe removing samples with specific SubcjectIDs;
  - preprocess_data(EEG_data_df, outlier_subjects): This function pre-processes dataset: impute it, removes outliers and drops useless columns;
  - load_data(csv_file_path, preprocess: object = True, outlier_subjects=[]): This function loads a csv_file into a pandas dataframe, with optional data preprocessing;
  - save_df(df, csv_file_path): This function saves a df on a csv file;
  - split_X_y(self, EEG_brainwave_df): This method split the EEG brainwave dataset into features and labels;
  - format_topredict_df(self, topredict_brainwave_df): This method assures that in the df there is no the label column;
  - create_train_test_split(self, EEG_brainwave_df, train_file_path="data/train.csv", test_file_path="data/test.csv"): This method splits a dataframe into training and testing data. This split is saved in the class;
  - training_data(self, train_file_path="data/train.csv"): This property returns training data. If the class has no training data loaded, tries to load it from file;
  - testing_data(self, test_file_path="data/test.csv"): This property returns testing data. If the class has no testing data loaded, tries to load it from file;
  - save_predictions(self, topredict_brainwave_df, predictions, file): This method saves on a file the result of the prediction: creates the original preprocessed df with addictional label column.