Motor Imagery (MI) for Left vs Right Hand Imaged Movement Prediction with Electroencephalography (EEG)

Wen Shi
shiwen0728@gmail.com

1. Abstract

This project finds a machine learning model to differentiate two motor imagery events detected by EEG signals. The subjects were prompted to imagine left vs right hand movement and the neural pattern of the subjects were recorded through the EEG and EOG data. Complex time-series data visualization and analysis in both time-domain and frequency-domain generated insights to extract physiological features. Four machine learning models were introduced to predict if the subject was thinking left vs right-hand movement. Then the best model with tuned parameters were selected through cross validation. The prediction accuracy of the best selected model is 74.86%, which shows advantages over a CNN model with an accuracy of 72.01%.

2. Introduction

Electroencephalography (EEG) is a non-invasive way to collect electrical signals from the scalp. Motor Imagery (MI) for left vs right hand imagined movement is one common neural pattern that is detectable through EEG signals. EEG signal is one-dimensional signal that records electrical activity of the brain across time, and similar to EEG, Electrooculography (EOG) records eye movements and blinks. In this project, I explored EEG and EOG signals to predict if a subject was thinking left vs right-hand movement.

Understanding how EEG and EOG signals correlate to the motor imagery and eye artifact is crucial in brain-computer interface (BCI) research. Analyze these imaginations in human brain and predict the motor imagery using only three EEG channels enables us to convert these neural signals into navigation commands. Ultimately it unravels the complexity of brain activity and benefits medical fields.

However, the goal is very challenging to achieve in the following ways.

1. It is inherently complicated and hard to abstract a physical model. Due to the nature of EEG and its time-varying properties (i.e. gel dries out, wires move around) and subject-varying properties (i.e. different headset setups, different head and brain folds), EEG signals are noisy and have high variance across samples within the same class.
2. Nonlinear behaviors. It is observed that many features such as statistical, time-domain, frequency-domain, wavelet, have been extracted. However, it still remains a question whether or not these extracted features are subject-specific optimal features. [1]
3. High dimensional feature space. 3 EEG electrodes and 3 EOG electrodes were used to collect the neural data, and each electrode was recorded at frequency of 250 Hz. Signals from each electrode are binned in 4 spectral channels.

Because of the large numbers of different types of data, the feature space has high dimensionality.

3 Implementation

3.1 Data Visualization

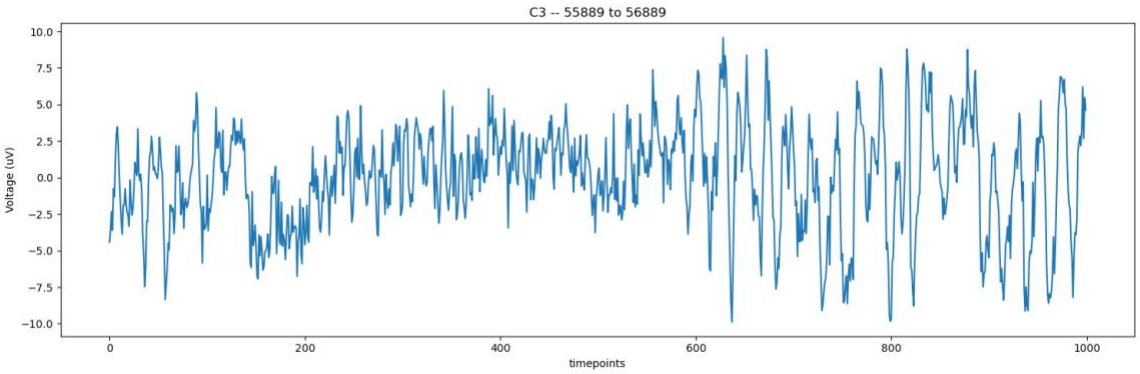
Raw data information

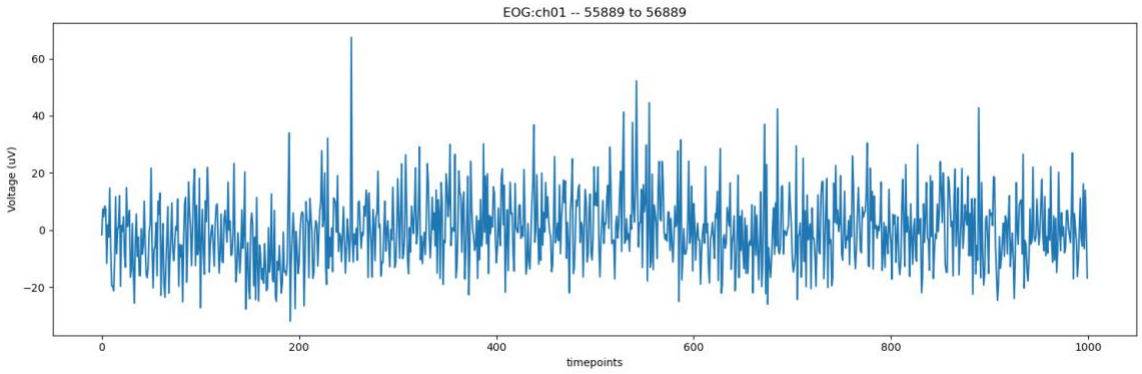
The time column is in ms and each recording comes in at 4ms intervals since the data is sampled at 250Hz. C3, Cz and C4 are electrode recordings in microVolts (uV) from the 10-20 EEG system. EOG:ch01, EOG:ch2, and EOG:ch03 are the EOG channels sampled at the same frequency. EventStart being 1 means the start of a trial. The labels of each trial correspond to EventType, meaning which event was started (left or right). The first EventStart ==1 will have the label in the first row of EventType. So the values in C3, Cz, C4, EOG: ch1, ch2, ch3 are numerical.

	time	C3	Cz	C4	EOG:ch01	EOG:ch02	EOG:ch03	EventStart
0	0	4.150454	0.192264	0.225834	-20.263981	18.738079	-5.889982	0
1	4	4.446479	1.913481	4.299992	-6.561379	41.260395	12.726024	0

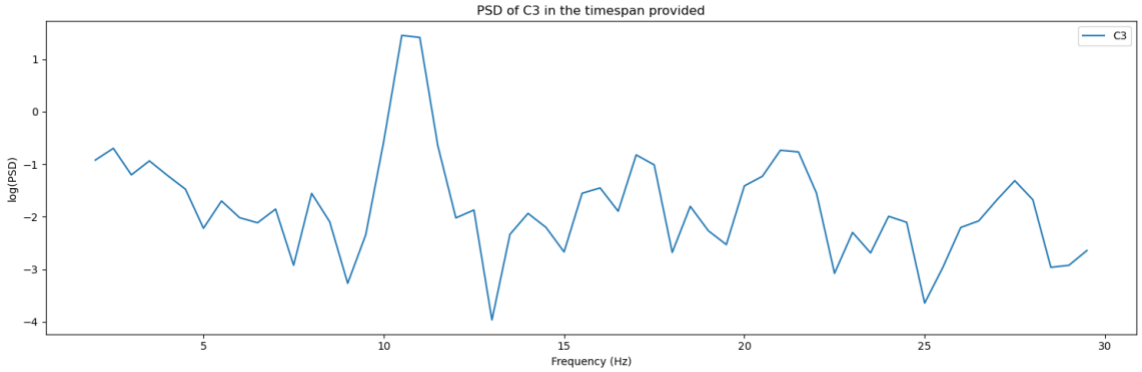
	EventType
0	0
1	1

Visualize continuous EEG and EOG data by timepoint

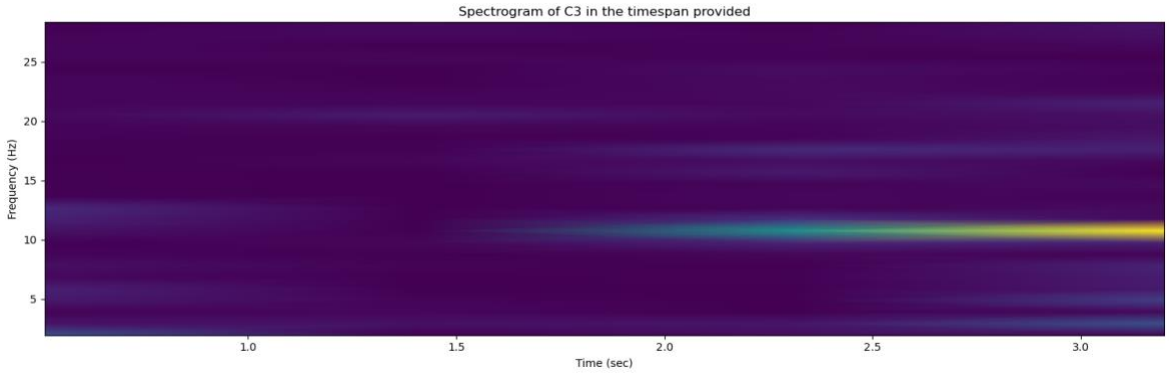




View 1000 time points of EEG channel and EOG channel. EOG channels fluctuate much more than EEG channels, in which the fluctuation would be blinks or muscle artifacts rather than brain signals.



Power Spectral Densities (PSDs) is a representation of the "amount" of each frequency a signal has.

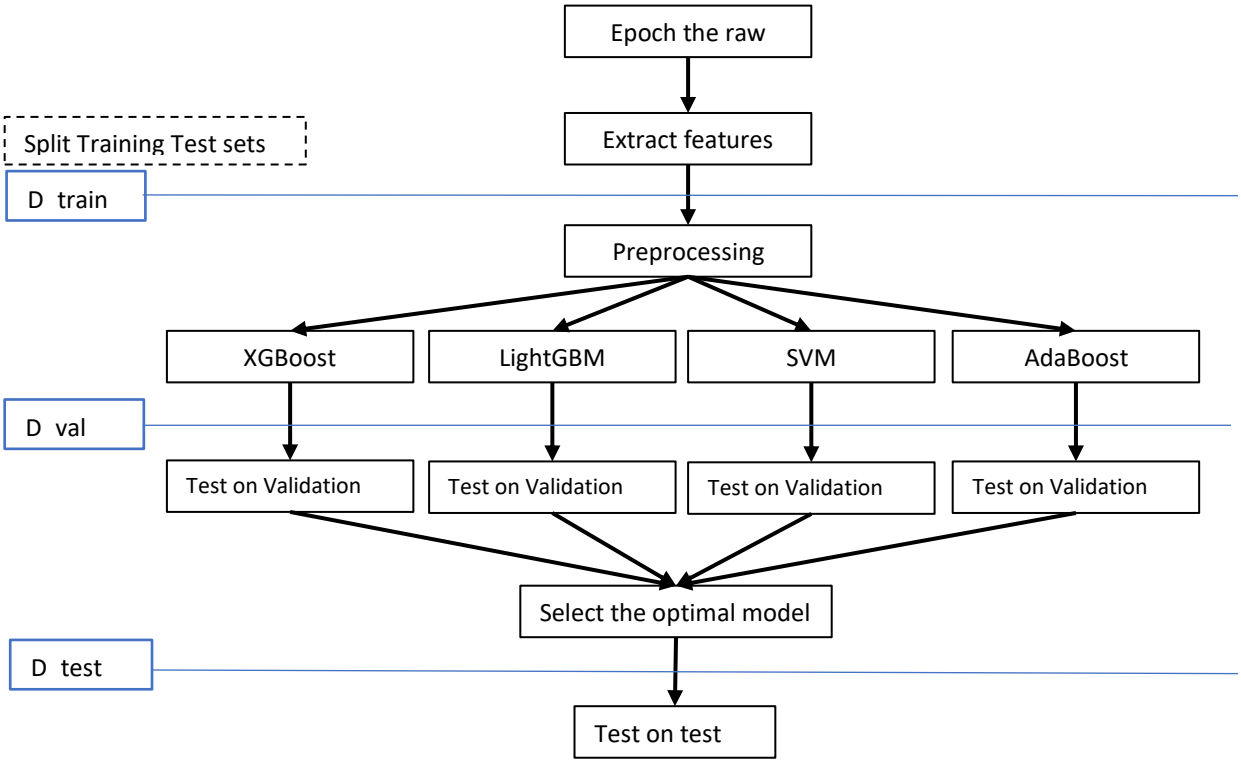


Spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time.

3.2 Dataset Methodology

There are 3960 trials in total with labels for this project. Each trial contains time series data in 3 EEG channels and 3 EOG channels. I developed features by analyzing PSDs, Power Bin with PyEEG, and Band ratios, calculating channel differences and time series statistics with NeuroDSP. In total, I extracted 330 features and selected 80 features for training and testing.

The 3960 data points are split with a ratio of 8: 2 into set D' (2944 data points) and Test set D_test (792 data points). Within D' set, I split into Training D_train (2355 points) and Validation set D_val (589 points) for cross validation. Cross validation with 5 folds and Grid Search are used to select the optimal parameters.



Preprocessing steps:

1. Feature selection and extraction
2. Normalization. The normalization factors were extracted from only training data and then passed to the test set.

Feature Extraction steps:

1. Based on prior knowledge, select features
2. Based on raw data exploration, select features. .

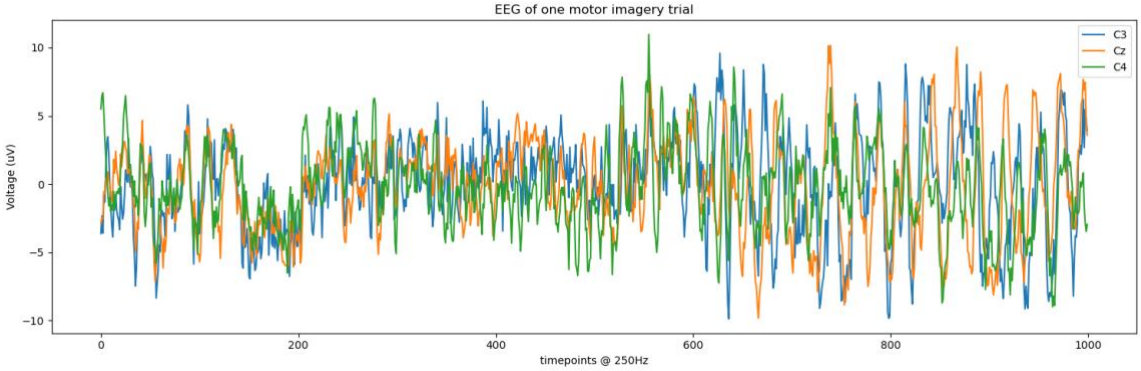
3. Put all features into a model first and weigh the importance of features. Select features with high importance.

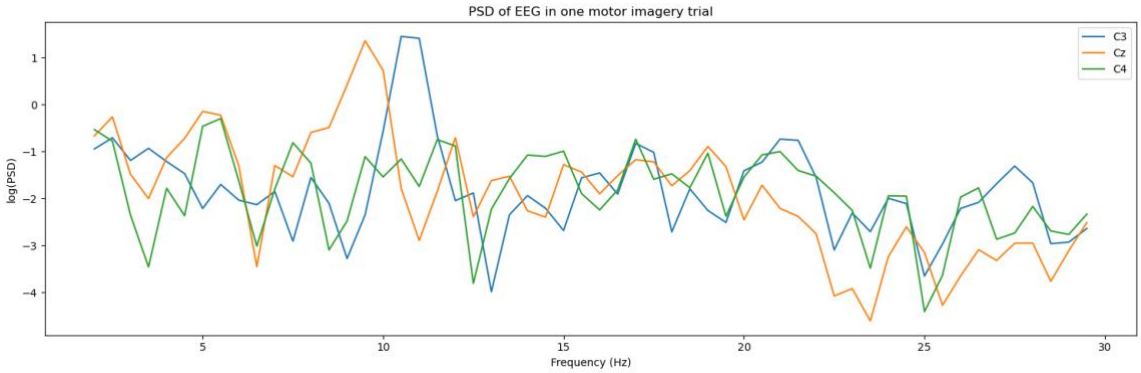
3.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

Epoch the data

Visualize EEG, EOG, and PSD data of Epoched data

The plots below show EEG and PSD for one trial. We can notice PSDs for a single 4 second trial is noisy.



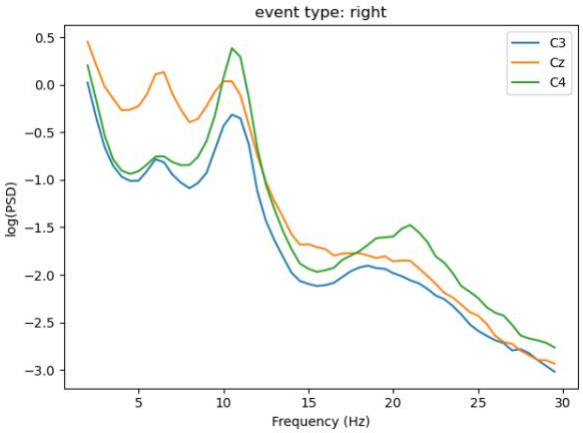


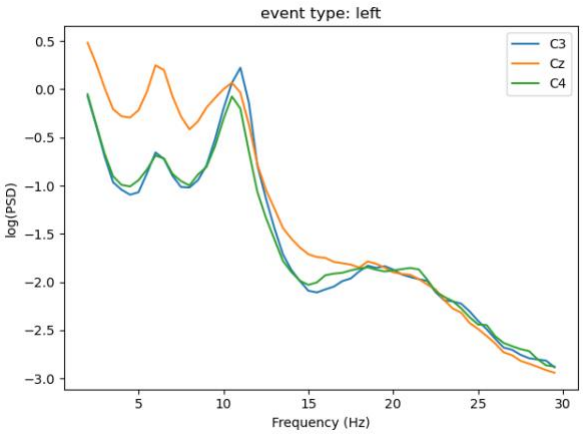
Feature extraction

Power Spectral Densities (PSD) analysis

Average PSD data

Taking the average of PSDs across trials to see “delta”, ”theta”, “alpha”, “beta” [(0.5, 4), (4, 7), (7, 12), (12, 30)] bands is commonly used in most EEG research. The following plots are the averaged PSD of a left label and a right label in 3 EEG channels.

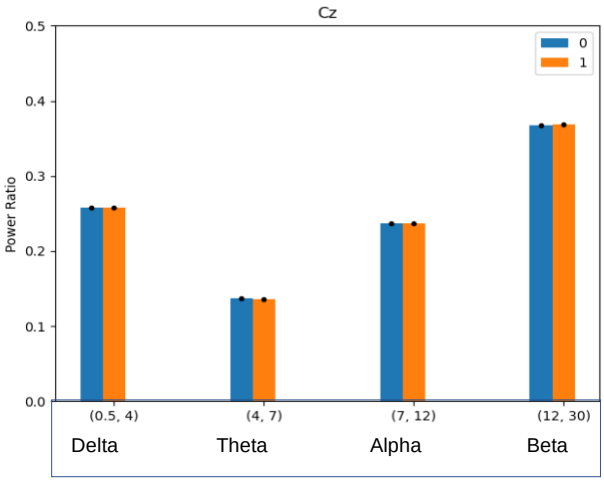




Integral of PSD in bin range

In order to extract features in EEG signals, it is common to bin the PSD within a range of frequency. Here are the plots of the spectral power in all the spectral bins corresponding to 2 labels in electrode Cz.

I extracted PSD data in C3,C4,Cz, EOG:ch1, ch2, ch3 electrode channel and PSD data in each channel is binned in delta, theta, alpha, beta bands. Overall, each electrode has 4 bands and there are 6 electrode channels in total.



Band Ratios

Research shows that ADHD patients have higher theta/beta ratio and memory impairment with higher theta/gamma ratio. However, there is not much literature on Motor Imagery, but the theta/beta ratios between frequency bands could be worth exploring.

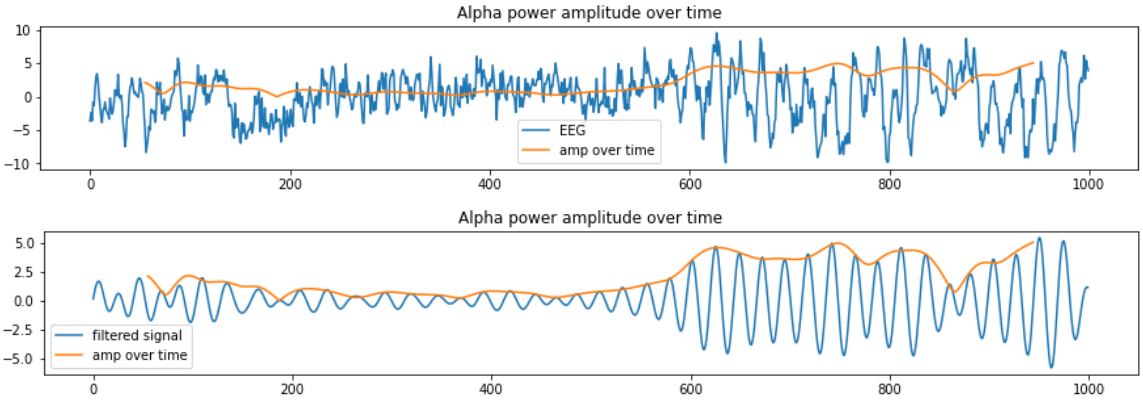
Channel differences

The plots of Average PSD data show the difference or ratios between channels could be more distinguishable than the absolute powers. So here I calculated the differences between C3 and C4 electrode channels in each band, the differences between C4 and Cz channels in EEG data.

Time series statistics

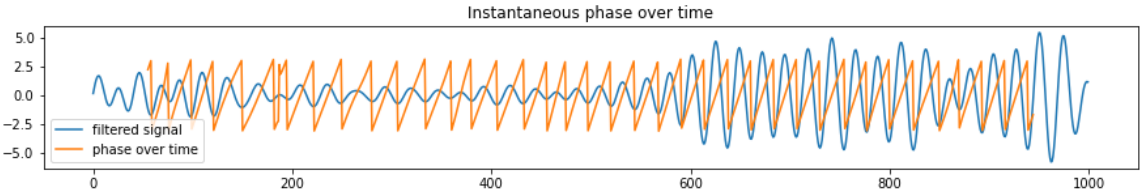
Instantaneous amplitude of the signal filtered to a band

The plots show a single trial EEG with alpha amplitude over time. I extracted statistical values from amplitude of each band in both EEG and EOG electrode channels. The statistics of amplitude include standard deviation, mean, median, max and min of first order differences.



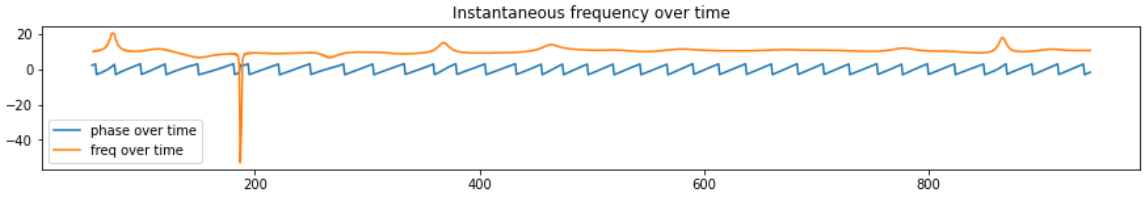
Instantaneous phase of the signal filtered to a band

Similar to statistics in amplitude, instantaneous phase calculated from the angle made from the analytical signal.



Frequency by time

Derivative of instantaneous phase over time is the frequency features. Research shows the frequency information is distinguishable in sleep stage detection, seizure detection and motor imagery.



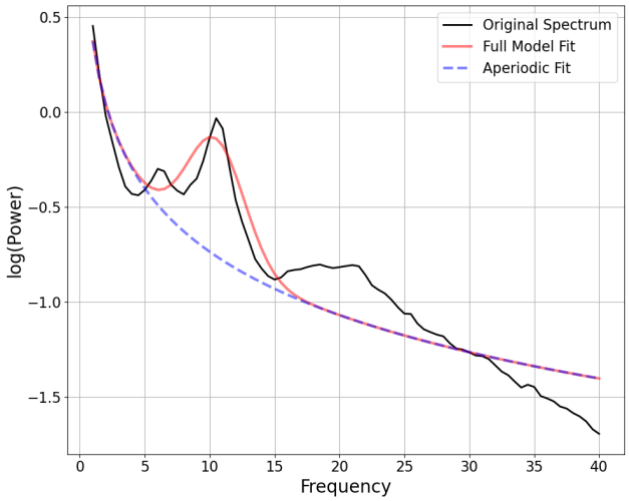
Parameterize PSDs

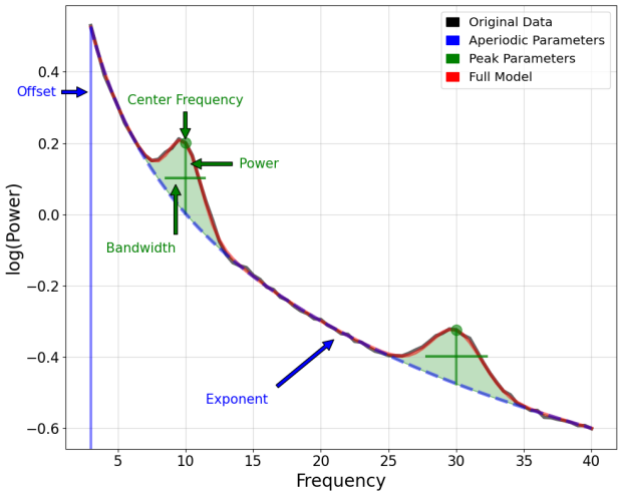
PSDs plots are visually helpful but hard to quantify, so it is necessary to parameterize PSDs. The 1/f noise is a common noise model seen in biology, as known as pink noise. FOOOF[3] parameterizes neural power spectra into periodic and aperiodic components. The parameters shown in the below figures are:

- Periodic
  - a is the height of the peak
  - c is the center frequency of the peak
  - F is the array of frequency values
  - w is the width of the peak

- Aperiodic
  - $L(F)=b-\log(F\chi)$
  - b = offset
  - χ = exponent (negative here)

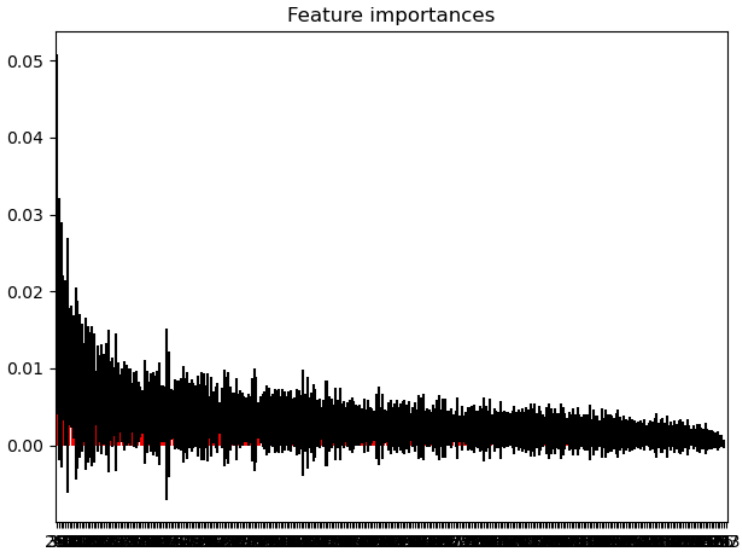
I used FOOOF package (https://fooof-tools.github.io/fooof/auto_tutorials/index.html) to extract such parameters in PSDs.





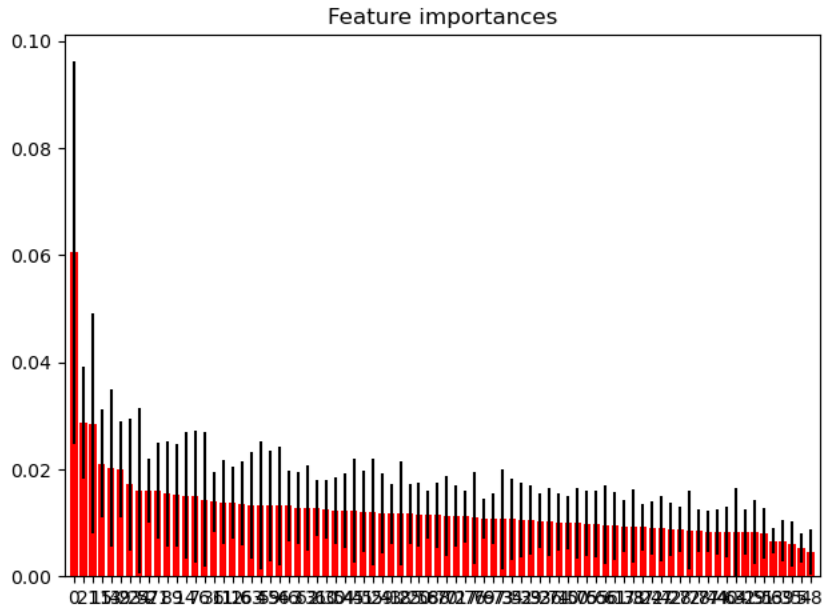
Feature ranking

I used random forest classifier from the open source library ScikitLearn to select the most important features that can classify the two events. I used function feature_importance_ to get the feature ranking, the higher the value shows the more important the feature is.



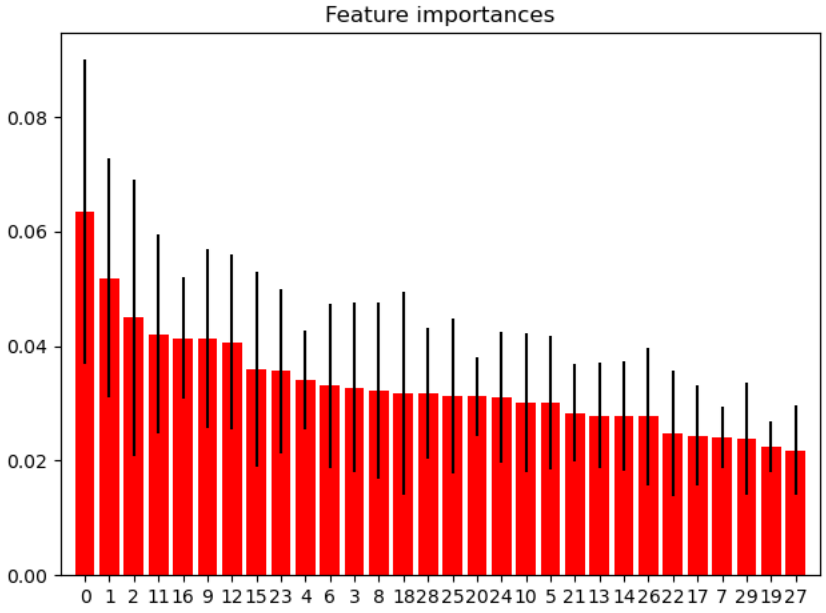

```
Random forest accuracy: 0.6213921901528013
```

I selected the first 80 features of the 320 features based on the ranking. Then I ran the random forest classifier again with the selected features, and the importance of every feature shows:




```
Random forest accuracy: 0.6994906621392191
```

In order to avoid overfitting if the feature dimension is too high, I selected 30 features and the results are:




```
Random forest accuracy: 0.6434634974533107
```

The evaluation of no feature selection and feature selection of 80 and 30 features shows that 30 features and 320 features do not have a significant difference in performance. It is likely that 320 features can overfit the model. However 80 features shows a significant improvement than 30 features, and it is not overfitting the model. The training features for the machine learning models are selected as these 80 features.

Standardization

Standardization is to scale the values of each column to be zero mean and unit standard deviation. Here min-max standardization is used to numerical features.

$$Column = \frac{column - column.min}{column.\max - column.min}$$

4 Machine Learning Models

4.1 Model Overview

The final model is chosen from a few machine learning models. They are Logistic regression, Support Vector Machine (SVM), random forest, AdaBoost, XGBoost, and LightGBM. And detailed illustration of each model is in this section below.

Due to the limited resources for result comparison, I implemented a Convolutional Neural Network (CNN) model as a comparison model in order to evaluate the performance of conventional machine learning models.

XGBoost

Extreme Gradient Boosting (XGBoost) is an ensemble learning algorithm of gradient boosted decision trees designed for improved speed and performance. XGBoost tends to perform well on structured or tabular datasets in classification and regression predictive modeling problems.

XGBoost Features

Regularization prevents overfitting. XGBoost penalizes complex models using both L1 and L2 regularization.

Continued training further boosts an already fitted model on new data.

Handling sparse data. Sparsity-aware finding algorithms are implemented in XGBoost to handle missing data values and data sparse problems.

Block structure for parallel learning. XGBoost makes use of multiple cores on the CPU using a block structure in its system design, which significantly speeds up the training process.

Parameters to tune in this implementation

The parameters can be divided into 3 categories, and here I selectively chose the important parameters among them to tune.

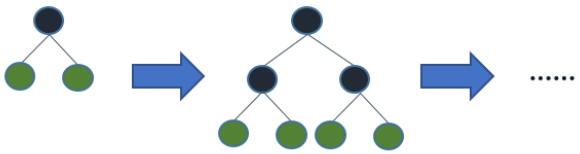
1. General parameters: booster

2. Booster parameter: eta, max_depth, min_child_weight, …
3. Learning task parameters: objective, eval_metric, …

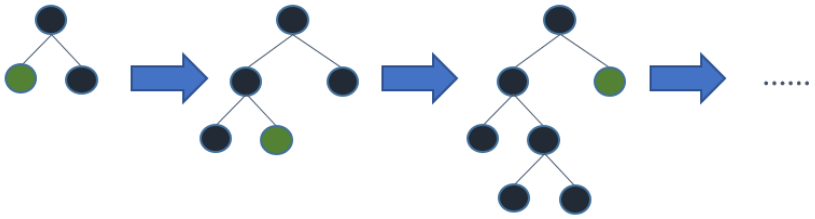
*Parameters are selected from XGBoost API:
https://xgboost.readthedocs.io/en/latest/parameter.html

LightGBM

Light Gradient Boosting Machine (LightGBM)[2] is a gradient boosting algorithm based on decision tree algorithms. Their development focus is on performance and scalability. LightGBM grows a tree leaf-wise instead of growing a tree level-wise. It does not use sorted-based decision tree learning algorithms that search the best split point on feature values, as XGBoost does. LightGBM uses histogram-based decision tree learning algorithms to achieve improvement in efficiency and memory usage.



Level-wise tree growth



Leaf-wise tree growth

Image courtesy of Analytics Vidhya
(https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/)

The primary advantage of LightGBM is its fast training speed. However, a leaf-wise tree is typically much deeper than a depth-wise tree for a fixed number of leaves. Leaf-wise splits tend to increase in complexity and result in overfitting. It can be overcome by specifying the parameter max_depth. So this parameter is more crucial than in XGBoost.

Parameters to tune

1. Overfitting and accuracy

   max_depth: limits the tree depth to avoid overfitting.

   num_leaves: is the main parameter to determine the model complexity. Due to the overfitting concern, num_leaves mush be smaller than 2 * max_depth.

   min_data_in_leave: is another important parameter to prevent over-fitting in a leaf-wise tree. Its optimal value depends on the number of training samples and num_leaves. The larger this value is, the less likely it is to grow too deep a tree.

2. Speed:

   bagging_fraction : uses bagging to achieve faster speed

   feature_fraction: uses feature sub-sampling for faster speed

In summary, the primary reason to choose the above two tree-based methods is that the model is nonlinear, which could fit the features better in comparison to a linear model.

SVM

Support Vector Machine (SVM) chooses a decision boundary that maximizes the margins from two classes. SVM model generates a binary linear classifier.

Parameters to tune

C: regularization parameter. The strength of the regularization is inversely proportional to C.

gamma: kernel coeficient.

kernel: specifies the kernel type in the algorithm, including 'linear', 'poly', 'rbf', 'sigmoid', …

Convolutional Neural Network (CNN)

CNN is not a candidate model in this project, due to lack of state-of-art results in this topic, I used a CNN model to generate results to evaluate the performance of this project.

Hypothesis sets for different models

XGBoost , LightGBM and AdaBoost:

F is space of regression tree (CART)

$$F = \{f(x) = w_q(x)\} \ (q = R^n \to T, w \in R^T)$$

SVM: mapping from R^{80} to R^1 (Trained with 80 features)

Select the best model

At last, I analyzed the predicted results of the above models to select the best model, and use that model to produce final test result. I used cross validation results as a reference to evaluate these models.

4.2 Training Process

4 different models were applied in training. CV results were also used to evaluate the performance of each model to find the best model. Here are the parameters of the model I applied:

XGBoost

```
params_xgBoost = {
    "eta": [0.05, 0.10, 0.15, 0.20, 0.25, 0.30],
    "max_depth": [3, 4, 5, 6, 8, 10, 12, 15],
    "min_child_weight": [1, 3, 5, 7],
    "gamma": [0.0, 0.1, 0.2, 0.3, 0.4],
    "colsample_bytree": [0.3, 0.4, 0.5, 0.7]
}
```

The complexity of the hypothesis set: 80 * 2355

The input matrix has 80 features as explicitly shown in feature extraction. Training set includes 2355 data points and the validation set has 589 data points.
The best parameters are:

```
{'colsample_bytree': 0.4, 'eta': 0.2, 'gamma': 0.0, 'max_depth': 12,
'min_child_weight': 1}
```

LightGBM

```
params_lightGBM = {"num_leaves":[26,31,36,41],
"learning_rate":[0.05,0.1],"boosting_type":['gbdt','dart','goss']}
```

The feature space and complexity of the hypothesis set of this model is the same as XGBoost.

AdaBoost

The feature space and complexity of the hypothesis set of this model is the same as other two boosting algorithms.

The parameters to be tuned are:

```
params_AdaBoost = {"n_estimators":range(30, 101, 10),
"learning_rate":[1, 0.1, 0.01], "algorithm": ['SAMME', 'SAMME.R']}
```

SVM

For a kernelized SVM, the feature function $\varphi: X \to H$ corresponds to the kernel function $k(x,y)=\langle\varphi(x),\varphi(y)\rangle_H$. The hypothesis set becomes:
$Hk=\{f(x)=\text{sign}(\langle w,\varphi(x)\rangle+b)|w\in b\in\mathbb{R}\}$
w optimizes SVM loss for training set $X=\{xi\}^n{}_{i=1}$ with the form of $w=\sum^n{}_i=\alpha i\varphi(xi)$.
So the hypothesis set is:
$HXk=\{f(x)=\text{sign}(\sum^n i=1\ \alpha ik(xi,x)+b)|\alpha\in\mathbb{R}n,b\in\mathbb{R}\},\ HXk\subset Hk$
The feature space is D+1 = 81, the complexity of the hypothesis set is:
(D+1) * N_train = 81 * 2355
The parameters to be tuned in this model are:

```
params_SVM = {'C': [1, 10, 100, 1000], 'gamma': [0.1, 0.01, 0.001,
0.0001], 'kernel': ['linear','rbf', 'sigmoid']}
```

4.3 Model Selection and Result Comparison

Cross Validation

In each parameter-specified model, cross validation was applied in 5 folds. Parameters for each model are different to avoid high variance caused by bad parameters rather than different model performances. Mean test score in cross validation was used as the criteria to select the best model.

The linear regression model performs the worst in the cv results due to non-linear behaviors of the problem. AdaBoost does not perform as well as XGBoost or LightGBM, since XGBoost and LightGBM are optimized boosting algorithms, which is explained thoroughly in the overview of different models.

XGBoost CV result

```
0.743(+ / -0.015) for {'colsample_bytree': 0.4, 'eta': 0.2, 'gamma':
0.0, 'max_depth': 12, 'min_child_weight': 1}
```

LightGBM CV result

```
0.727(+ / -0.029) for {'boosting_type': 'dart', 'learning_rate': 0.1,
'num_leaves': 41}
```

SVM CV result

```
0.732(+ / -0.042) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
```

AdaBoost CV result

```
0.697(+ / -0.028) for {'algorithm': 'SAMME', 'learning_rate': 1,
'n_estimators': 100}
```

CNN

The CNN model is composed of 2 conv layers, one MaxPooling layer, 3 dropout layers, and 3 dense layers. The architecture of this neural network is:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 998, 4, 32)	320
batch_normalization (BatchNo	(None, 998, 4, 32)	128
conv2d_1 (Conv2D)	(None, 996, 2, 64)	18496
batch_normalization_1 (Batch	(None, 996, 2, 64)	256
max_pooling2d (MaxPooling2D)	(None, 498, 1, 64)	0
dropout (Dropout)	(None, 498, 1, 64)	0
flatten (Flatten)	(None, 31872)	0
dense (Dense)	(None, 256)	8159488
batch_normalization_2 (Batch	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
batch_normalization_3 (Batch	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129

Total params: 8,213,249
Trainable params: 8,212,289
Non-trainable params: 960

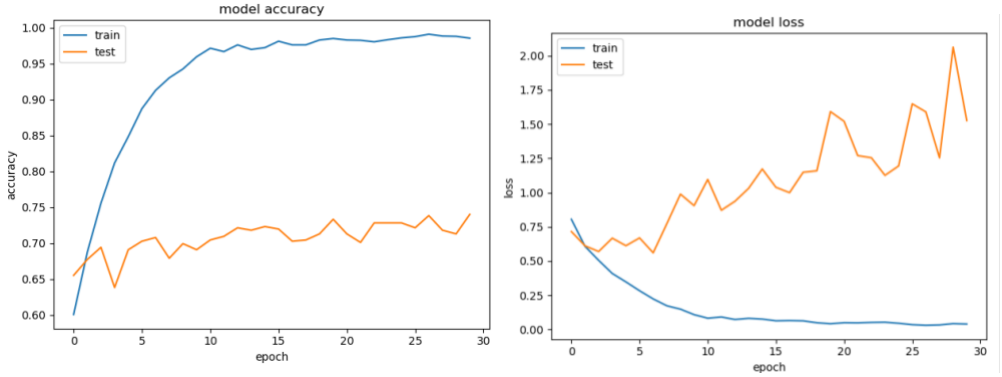
The results of CNN model are below:
CNN Test accuracy: 0.720108695652174

	Precision	Recall	F1-score	support
Class 1	0.69	0.82	0.75	376
Class 2	0.77	0.62	0.68	360
Accuracy			0.72	736
Macro avg	0.73	0.72	0.72	736
Weighted avg	0.73	0.72	0.72	736

Confusion matrix

	Positive	Negative
Predicted Positive	(TP) 308	(FP) 68
Predicted Negative	(FN) 138	(TN) 222

The validation results of the CNN model are shown in the plots below. In total, there are 30 epochs in training. At around the 10th epoch, the training accuracy reaches the plateau and validation loss starts to increase.



5 Final Results and Interpretation

After I selected the best model with tuned parameters from the cross validation results, I trained this model with Training set D' and test it on test set D_test. The classification accuracy of this model is:
xgBoost accuracy: 0.748641304347826

Confusion matrix:

	Positive	Negative

Predicted Positive	(TP) 272	(FP) 104
Predicted Negative	(FN) 81	(TN) 279

	Precision	Recall	F1-score	support
Class 1	0.77	0.72	0.75	376
Class 2	0.73	0.78	0.75	360
Accuracy			0.75	736
Macro avg	0.75	0.75	0.75	736
Weighted avg	0.75	0.75	0.75	736

Generalization bounds

$$E_{out} \leq E_{D_{test}} + \sqrt{\frac{1}{2N} \ln \left(\frac{2M}{\delta}\right)}$$

$$M_{test} = 1, N_{test} = 736$$

So the generalization bound $\varepsilon \leq 0.045$ with a $probability \geq 1 - \delta$, when tolerance $\delta = 0.1$.

XGBoost model performs better than the rest, because the algorithm is optimized for accuracy and speed. LightGBM is the fastest boosting algorithm among the 3 boosting algorithms, and it has a slightly lower performance than XGBoost. This is because it trades off its accuracy for speed, but this model is a very efficient algorithm for large datasets. AdaBoost is a boosting baseline model, and achieves an accuracy similar to the CNN model. SVM performs the worst due to the non-linearity in this problem. SVM is the slowest model but performs surprisingly good in comparison to AdaBoost. This is understandable because SVM shows its advantages when the features are sufficient.

The model performs slightly better than the CNN model, and there is room for improvement.

6 Summary and conclusions

This problem is a very challenging problem in an untapped field for EEG signals. Due to the unknown in neural data, scientists still do not understand if signals being detected by EEG are correlate to Motor Imagery, which makes feature extraction very challenging. The noisy nature of EEG and EOG data and subject-varying properties (same user can have distinct signals in 2 different trials) also make this problem very challenging.

I have extracted and selected features that represent EEG signal properties in time-domain and frequency-domain. And it achieves a better accuracy than a commonly used CNN model. However, due to the noisy data and unknown correlation between motor imagery and EEG signals, the discovery is very promising.

Feature selection is the key to a good performance. In order to improve the model, the future directions will be explore more features that can correlate to Motor Imagery in EEG signals. In addition, optimize the parameters in the model to improve the performance.

7 References

1. Sreeja, S. R. *et al.* Motor Imagery EEG Signal Processing and Classification Using Machine Learning Approach. in *Proceedings - 2017 International Conference on New Trends in Computing Sciences, ICTCS 2017* (2017). doi:10.1109/ICTCS.2017.15.
2. Ke, G. *et al.* LightGBM: A highly efficient gradient boosting decision tree. in *Advances in Neural Information Processing Systems* (2017).
3. Haller, M. *et al.* Parameterizing neural power spectra. *bioRxiv* (2018) doi:10.1101/299859.

8 Data Availability

The experiments used to produce EEG and EOG data by R. Leeb et al are accessible in BCI Competition 2008. (http://www.bbci.de/competition/iv/desc_2b.pdf)
The python packages neurodsp and pyeeg are used to process EEG signals. Their APIs also inspired features extraction in this project.