Seizure detection onset using Time, Frequency Correlation and Machine learning

Armand Hoxha

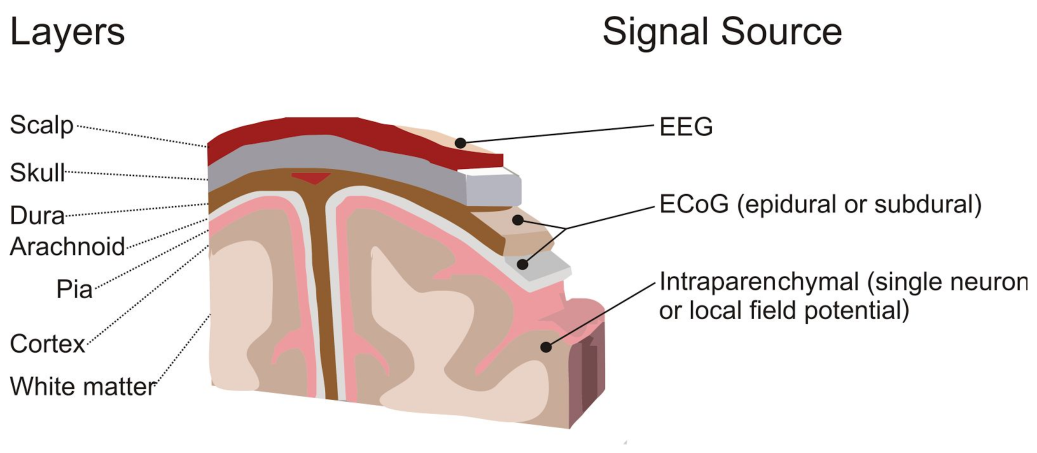
Definition

Project Overview

Chronic epileptic seizure affects over two million Americans, and approximately 50 million worldwide. The need of a warning sign, to give patients enough time to stop activities that would otherwise be dangerous due to an epileptic attack is essential and can be provided with the present technology.
In this project, I created a classifier capable of predicting seizure development and occurrence from pre recorded Intracranial Electrocorticographic data of human and dog patients. The classifier was trained using data provided by this Kaggle Competition. The current project is a precursing step for successful interventive/ therapeutic device to prevent or warn patients of an incoming seizure.

Problem Statement

Epilepsy is a common neurological disorder characterized by recurring seizures and abnormal brain activity in focal areas of the brain. Approximately 30% of patients suffering from focal seizure require surgical intervention. To determine where the seizures occur, surgeons use a combination of history, physical exam, as well as neuroimaging techniques such as EEG or intracranial EEG in cases where activity is hard to localize. Patient hospitalization can extend to weeks for enough seizures to be recorded using intracranial electrodes. In cases of a brain lesion 80% of surgeries are successful in rendering patients seizure free, however in case of a lack of lesions, only 50% of the patients make it through to live the rest of their lives without lesions.


Layers
Signal Source
Scalp
Skull
Dura
Arachnoid
Pia
Cortex
White matter
EEG
ECoG (epidural or subdural)
Intraparenchymal (single neuron
or local field potential)

One of the reasons suspected for failure of surgical removal of the seizure piece of the brain is that epilepsy could be a network wide degenerative disease; the removal of an intricate network of brains would leave the patient with a diminished quality of life.
Recent advancements in the Brain to Machine Interface (BMI or BCI) have shown great success in the ability to read real time data, and apply machine learning algorithms to estimate the state of the underlying neural mechanisms of the brain. Vaishali et al (2015) in his review of the successes of BCI in the field of biotechnology, shows that most of the work done in the field has either an investigative purpose or an enhancement end result. Machine learning algorithms have been used to investigate brain dynamics while participants execute a task wearing the Electroencephalography (EEG) cap, typically focused in the low and high brain frequencies (8-12 Hz and 12-28 Hz). This information can be used to drive a beneficial effect for patients, Chung et al (2011) demonstrates the possibility of to control the movements of an electric motor guided wheelchair using real time EEG data. The field is still in massive development and with the rise of publishing and public sharing.

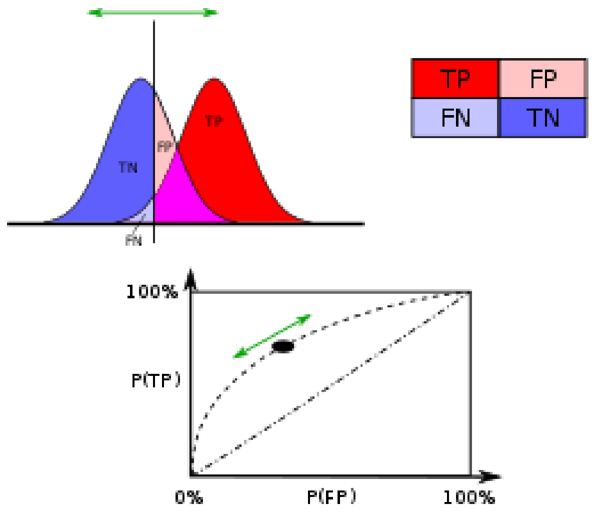
The goal is to create an Ecog (intracranial encephalography) data classifier to predict whether a one second segment of data pertains to the first fifteen seconds of a seizure episode, after fifteen seconds within the seizure, or a non-seizure episode.

Metrics

The competition metric is the mean area under the ROC curve (AUC) of two predictions. First prediction must predict the probability that a clip is a seizure. Second prediction is the probability that a clip is within the first fifteen seconds of a seizure. Early clips are double counted because early detection is critical for intervention purposes.

$$Score=1/2(AUC_{seizure}+AUC_{early})$$

The ROC curve is a plot of the True Positive Rate (on the y-axis) vs the False Positive Rate (on the x-axis), for every possible classification threshold of the model to separate a class; both True Positive Rate, and False Positive Rate range from 0 to 1. In this way an ROC curve visualizes all possible classification thresholds, whereas misclassification rate only represents your error rate for a single threshold. Thus a classifier with a ROC curve that hugs the upper left corner does a very good job in separating the classes; a classifier that approaches the diagonal line, does no better than random guessing. Area Under the Curve (AUC) is the percentage of the ROC line that covers the box from 0 to 1 for both axes.



The ROC curve is independent of the classifier prediction range; if the predictions are between 0 to 1 the plot is drawn the same even if they were between 0.9 and 1. The ROC curve is also independent of differences in classes; if there is more of one class than the other, the ROC curve jis still drawn the same way and is unaffected by such differences.

Design

Considering the size of the data (~50GB), a method to reduce the size is necessary to test different ideas effectively. A module called pickle, can be used to save extracted features from data, thus instead of loading 50GB of data to make predictions, one would have to load a much smaller amount of data. Features to be extracted will be considered using the information provided by Schindler et al 2007, and Muller 2005, indicates that an important feature to be examined could be the correlation of EEG and Ecog channels in both time and the frequency domain. Correlation measures the mutual relationship or connection between two things, in this case the authors suggest to measure the correlation between the data channels. The two types of correlation discussed are frequency correlation and time correlation.
Correlation is a measure between 1 (a=b) meaning completely identical relationship, and -1 meaning that there is an antagonistic behavior (b=-a). In time correlation we measure whether the channels are receiving similar data in the time series, where as frequency correlation measures "synchronous" brain activity in terms of frequency bands and their power. Thus an outline of the steps to follow would be:
1- Numpy will be used to normalize data across channels
2- Scipy to compute the Fourier Transform of the time series data
3- Use Numpy to compute the correlation matrix from he Fourier Transform data

4- Compute the Time series channel correlation
5- obtain only top triangle of the correlation matrix in a single dimension array NOTE about correlation
matrices: A correlation matrix is mirrored across the diagonal, thus only half is needed)
After the features are extracted, they will be saved using pickle, to enhance the speed of loading the
features for testing different learners, and debugging.

Analysis

Data Exploration

The competition data has 58,837 one second segments of data from 12 subjects (4 dogs, 8 human) who suffer from chronic seizures. Segments are denoted as ictal for seizure, interictal for non-seizure, and test for testing purpose. All data segments have the following fields:

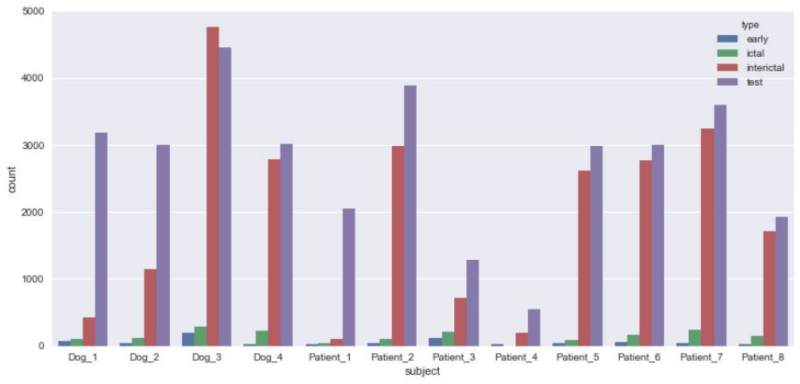
- **Latency:** how far into a seizure is the segment (as defined by experts clinicians)
- __header__:platform info that saved data into segments
- __globals__:empty
- Channels:name of the channels by location
- Freq:sampling frequency of the data
- __version__: always 1.0
- **Data**:all data in shape channels x sample

Most important field is Data, and the feature extraction is focused on this field only.
It must be noted that subjects have different sampling rates, as well as number of channels.

Subject	Sampling Frequency	Number of channels
Dog 1	399.6	16
Dog 2	399.6	16
Dog 3	399.6	16
Dog 4	399.6	16
Patient 1	500	68
Patient 2	5000	16
Patient 3	5000	55
Patient 4	5000	72

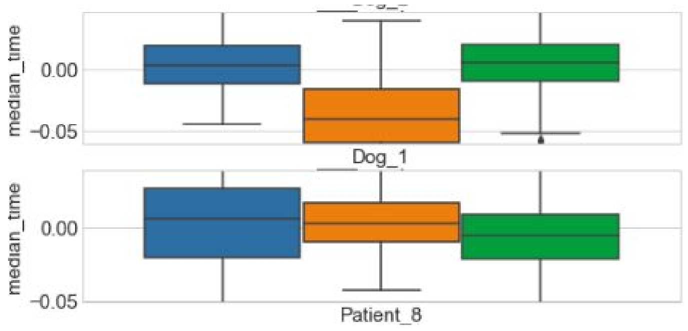
Patient 5	5000	64
Patient 6	5000	30
Patient 7	5000	36
Patient 8	5000	16

Glancing at the above table, clearly making a learner specific for each Subject is the best way approach the problem, as with different number of channels the number of features becomes hard to maintain to a single size, and little information is known about the condition of the seizure and their onset location.



Looking at the above countplot, majority of the data is interictal (non- seizure), and data distribution varies greatly; Patient_3 and Patient_4 have the least amount of data, with Patient_4 having no ictal (seizure) data.

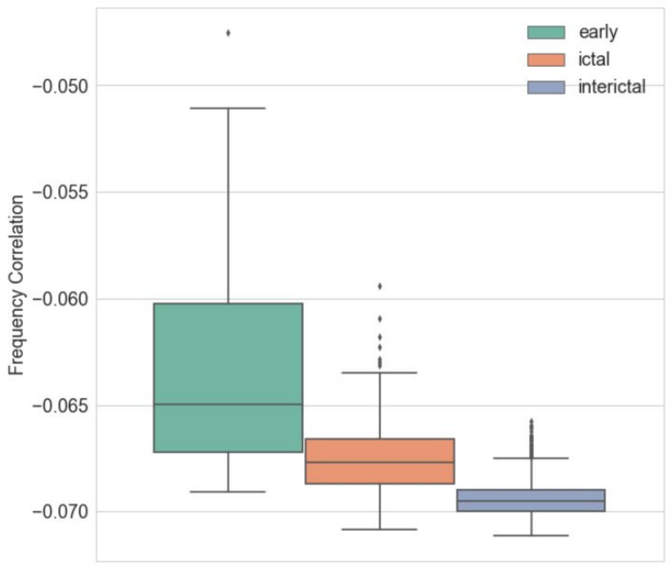
Distribution of time Data



These are two boxplots that represent the time series data data distributions for two subjects (blue - early, orange- ictal, green interictal). Muller et al (2005) suggest that data is average referenced across channels to remove the natural baseline of each data channel. For this numpy.scale can be useful to normalize data across the the columns of the data structure (data field is a CxP matrix, C for channels and P for data points).

Frequency Correlation

According to Schindler et al and Muller et al, one of the biological markers of seizures is synchronization of Electroencephalography (EEG) and Electrocorticography (Ecog) channels during an ictal episode (seizure). The authors have focused most of the analysis on the frequency spectrum of the data. During an epileptic seizure, the brain reaches a state of electrical synchrony, which can be recorded using the neuroimaging techniques of EEG or Ecog. In order to achieve similar results to the mentioned papers to use their results as features for the data, frequency correlation will be extracted from the time series data. To do this, the Fourier Transform of the data will be computed, then the cross channel correlation will be calculated which will yield a correlation matrix. The correlation matrix tells whether channels are acting alike in the frequency domain, or unlike each other. Thus, a value of 1 means that the two channels are in synchrony, 0 for unrelated activity, and -1 for antagonistic activity. In order to measure the strength of the overall correlation matrix, the eigenvalues will be computed as well.



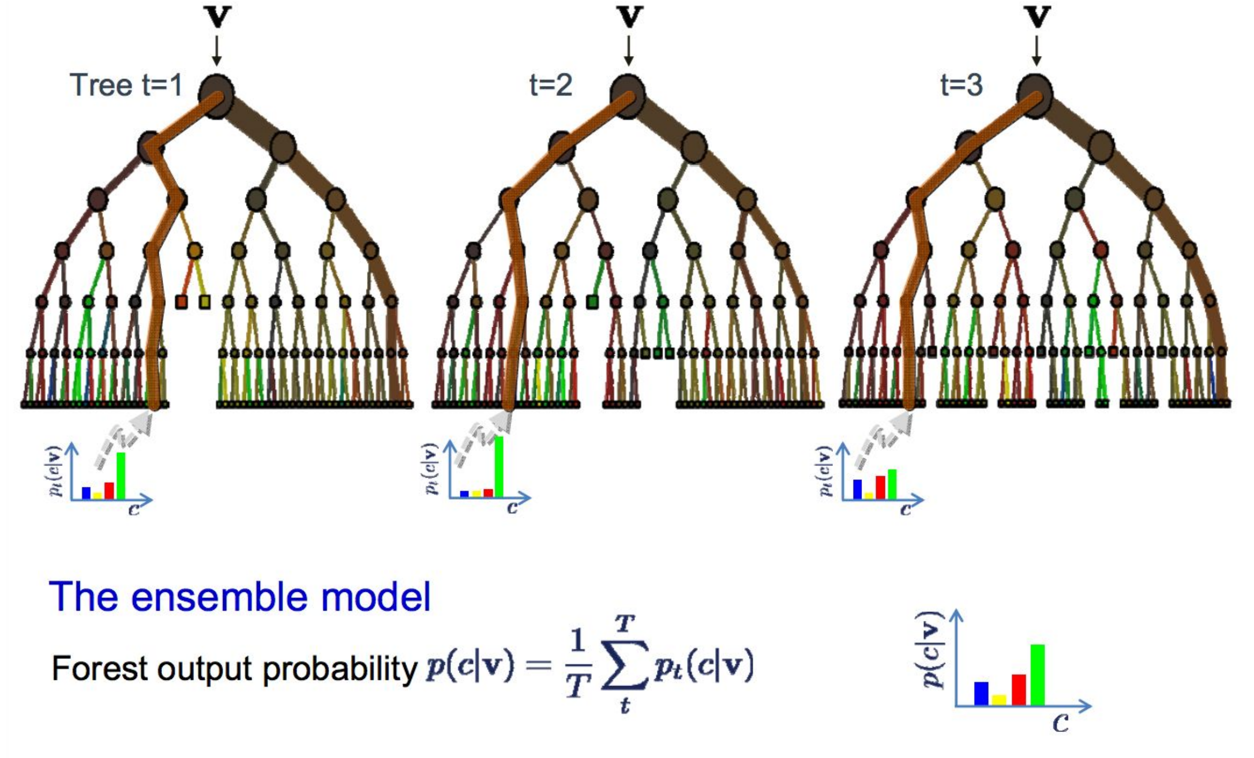
Time Correlation

Schindler et al and Muller et al, also report a common visual trend in the time series data; an increase in the slope of signals. The increase in the slope of the time series data in unison for multiple channels can be measured by computing the correlation of channel activity in the time domain. Following a similar approach to the frequency domain, eigenvalues will also be computed.

Algorithm and Techniques

There are many variables to adjust for in picking features, and also a lot of different methods to apply learning. This approach is made easy by utilizing scikit-learn which is a python machine learning library offering many different classifiers that can be implemented by easely substituting code. After testing different types of classifiers, the chosen model is a Random Forest Tree Classifier. A Random Forest Tree Classifier is an ensemble of classifiers (in this case Decision Tree Classifiers), that uses the output of each model to estimate a final answer. A Tree

Classifier predicts value of a target variable by simple decision rules, which works well with both categorical and numerical data. Decision Trees have one fall back which they tend to be unstable, in cases where the data changes between examples, as a new tree will need to be


v
Tree t=1
t=2
t=3
$p_t(c|\mathbf{v})$
c
The ensemble model
Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_t^T p_t(c|\mathbf{v})$
$p(c|\mathbf{v})$
c

generated for unique instances; this can lead to a common issue known as over-fitting. To mitigate the effect of over-fitting we use Random Forest (of) Tree Classifiers; each Tree Classifier is exposed only to part of the sample data, and another Tree Classifier is exposed to another part of the from the same sample. In the end, the Random Forest tallys the answers from each Tree Classifier, and by a majority rule decides the final answer. The algorithm outputs an assigned probability for each class, which is submitted as a .csv file to kaggle to check outcome. The parameters that required optimization are:
-n_estimators: number of estimators per data sample to average
- max_features: number of features to consider
-max_depth: maximum depth of tree
-min_samples_leaf: minimum number of samples required to be at a leaf

These parameters are used in a grid search for optimal parameters, for each subject. Since subjects' data varies in sampling frequency, number of channels, channel locations, seperate learners are implemented for each subject.

Benchmark Model

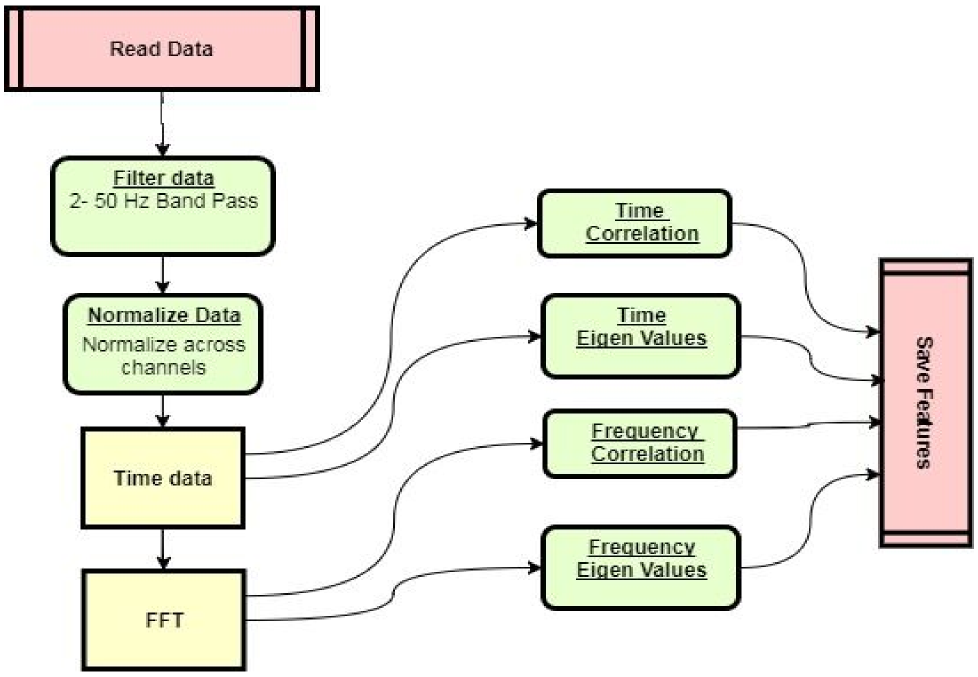
To objectively compare the progress of predicting models, the benchmark model learner will be a K-Neighbors classifier from sklearn:
sklearn.neighbors.KNeighborsClassifier(n_neighbors=10, weights='uniform', algorithm='auto,leaf_size=30)

According to Muller et al (2005) once onset of a seizure begins, channels start acting more alike, and during a complete seizure almost all channels act alike. Considering that our features measure the relationship between channels, it would be reasonable
to assume that KNN would perform well.

Using the above learner on all the data, and submitting it to the Kaggle Competition Submission, yields a score of 0.70128, using the above mentioned metric of area under the ROC curve.

Methodology

Data Preprocessing



Data preprocessing is executed by "feature_extractionv2.py" and consists of following steps:

1. Read in .mat file
   scipy.io.loadmat(file_name)

*file_name being the *.mat file*

2. Filter data 2 to 50 Hz using bandpass 4th order filter
   ```
   scipy.signal.butter(2,[1/200 50/200], btype='bandpass') #for each channel
   ```

3. Normalize data
   ```
   temp_mat['data']=sklearn.preprocessing.scale(temp_mat['data'],axis=0)
   ```
   *temp_mat is the dictionary after loading the *.mat file, and the ‘data’ field, is the actual Ecog data*

4. Run FFT of data
   ```
   for ch_num in  numpy.linspace(0,resampled.shape[0]-1,resampled.shape[0]):
   fft_matrix.append(numpy.real(numpy.fft.rfft(resampled[int(ch_num)]))[freqs[0]:freqs[1]]. tolist())
   ```
   obtain FFT transform of data, for each channel from the specified frequencies in the list ***freqs****, this is a tunable variable, in case one would want to switch the frequencies that are extracted; to extract the full frequency band, then freqs=[0 1990], but it is not reasonable*

5. Run frequency correlation among Ecog channels, get eigenvalues for frequency correlation
   ```
   corr_matrix=numpy.corrcoef(fft_matrix)
   eigenvals=numpy.absolute(numpy.linalg.eig(corr_matrix)[0])
   eigenvals.sort()
   corr_coefficients=upper_right_triangle(corr_matrix)
   return numpy.concatenate((corr_coefficients,eigenvals))
   ```
   Compute the correlation matrix; compute the frequency correlation between each channel, and compute the eigenvalues from the correlation matrix
   Upper_right_triangle *simply takes the top triangle from the correlation matrix and reshapes it to one dimensional list*

6. Get time correlation among channel, get eigenvalues for time data correlation
   ```
   scaled=preprocessing.scale(time_data,axis=1)
   corr_matrix=numpy.corrcoef(scaled)
   eigenvals=numpy.absolute(numpy.linalg.eig(corr_matrix)[0])
   eigenvals.sort()
   corr_coefficients=upper_right_triangle(corr_matrix)
   return numpy.concatenate((corr_coefficients,eigenvals))
   ```
   Compute the correlation matrix; compute the time correlation between each channel, and compute the eigenvalues from the correlation matrix
   Upper_right_triangle *simply takes the top triangle from the correlation matrix and reshapes it to one dimensional list*

7. Save extracted features to a pickle file; the name of the file includes subject, segment number, and type of data (early, ictal, interictal or test).
   pickle_out=open(file_name+'.pickle',"wb")
   pickle.dump(features,pickle_out)
   pickle_out.close()
   File_name contains the keyword (either ictal, interictal, early or test)
   Pickle is the module being used to save the features
   Features is a list including, frequency correlation with eigenvalues and time correlation with eigenvalues

After preprocessing the data needed to be analyzed is reduced drastically in size (from ~50GB to 1GB). The preprocessing step also has adjustable parameters such as:

- Resample to 400 Hz
- Filtering options (change the bandwidth of the band-pass filter)

Implementation

Considering the size of the data (~50GB), a method to reduce the size is necessary to test different ideas effectively. A module called pickle, can be used to save extracted features from data, thus instead of loading 50GB of data to make predictions, one would have to load a much smaller amount of data.

After the data is preprocessed and stored using the pickle module, the features for each subject's data segment is loaded for each learner. Data is split into train and test sets (90% train 10% test), and then we use a GridSearchCV() to find optimal parameters for the Random Forest Tree Classifier. The Learner with the optimized parameters is then saved using pickle (in case of retesting), and fits the whole amount of training data available, then predicts probability for each test segment whether it is early, or ictal (thus (0,0) refers to no seizure, (1,0) early seizure and (0,1) mid seizure).

It is worth to be noted, that careful attention must be paid to Patient_4, as there is no **ictal** data available, thus even the testing data has no ictal data, thus there the prediction for Patient_4, ictal is **always 0.**

All predictions, and filenames are stored in dictionary, which is then formed into a pandas DataFrame to be finally saved as a .CSV file, which is the accepted format for submitting results.

Results

After testing different types of classifiers, these are the results from submitting the outputs into kaggle.

Classifier	**Score**
Random Forest (predict prob)	0.92487
Ada Boost n_estimators=300	0.81397
KNN (predict prob)	0.78317
KNN(**Benchmark)**	0.70128
Random Forest	0.65262
Random Forest (no optimization)	0.65262
SVC	0.69005

It is clear that due to the measuring metric, providing a soft prediction (predicting probability) will provide a better score. It is worth noting that although Random Forest gives the highest score, it is by far the slowest learner, with KNN being the fastest of all. However, this competition is heavily dependent on the accuracy of the classifier and the scoring ignores computation time, thus any further discussion on execution times is omitted.

The objective of the project is to create a learner that would perform with a score of 0.8 or better; the score was almost achieved by the benchmark model with optimization using GridSearch and switching to soft predictions using predict_proba method. However, in order to exceed the the goal set, other learners have been explored, with Random Forest being the best model.

Best Model
learner_opt=RandomForestClassifier(n_estimators=best_parameters['n_estimators'], min_samples_split=best_parameters['min_samples_split'],random_state=1 ,bootstrap=best_parameters['bootstrap'])

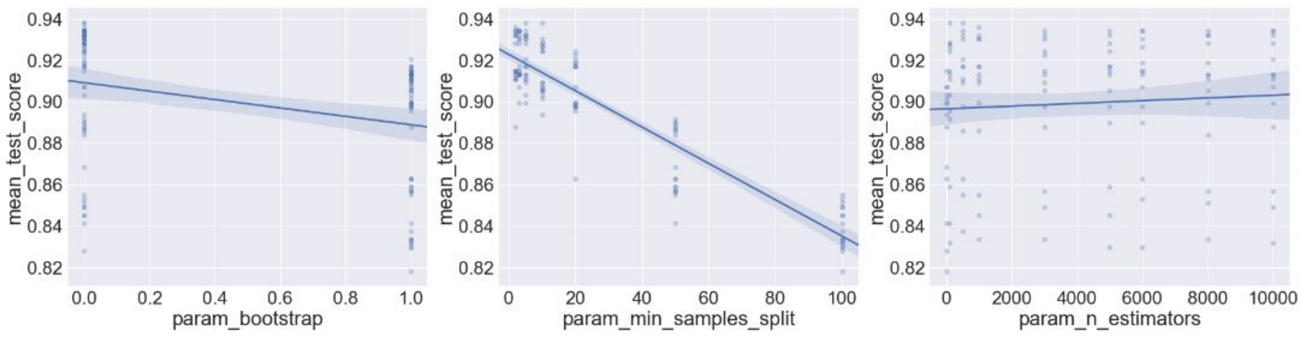
The best_parameters dictionary is the output of the hyper parametrization executed by using GridSearchCV. Each subject has their own learner, thus each subject a different set of parameters for the RandomForestClassifier. The results of the data have been tested using different random states, and with some of the pre-processing steps changed. Changing the sampling rate to a 400Hz (since the lowest sampling rate was at 400Hz), does not improve the quality of the predictions; it rather lowers the score from the Kaggle Competition website.

Discussion

In this project, I developed an Ecog data classifier for seizure onset prediction. To further understand how the classifier works, and its robustness it is important to include the results from **GridSearch() ;** how does each parameter affect the classifier?

parameters={'n_estimators':[10,100,500,1000,3000,5000,6000,8000,10000],'min_samples_split':[2,3,5,10,20,50,100], 'bootstrap':[False,True]}
clf=GridSearchCV(learner,parameters,n_jobs=1,verbose=2,return_train_score=True)

After GridSearch, correlation between parameters and mean_test_score reveals which parameters affects the score in a good or bad way.



The figure above is a regression plot of the mean_test_score vs, Bootstrap (True or False), minimum sample split (2,3,5,10,20,50,100), and number of estimators used in the ensemble (10,100,500,1000,3000,5000,6000,8000,10000). Some of the effects of the parameters are fairly clear from the plot; Bootstrap should not be used, minimum samples split should be kept under 10, and the number of estimators seems to have an overall slightly good effect. To further explore the relationship that the parameters have on the score, it would help to compute the Pearson Correlation between the test scores and the parameters:

```
import scipy.signal as sig
bootstrap_corr=np.corrcoef(gsdf.mean_test_score.as_matrix(),gsdf.param_bootstrap.as_matrix())[0][1]
sample_split_corr=np.corrcoef(gsdf.mean_test_score.astype(float),gsdf.param_min_samples_split.astype(float))[0][1]
n_estimators_corr=np.corrcoef(gsdf.mean_test_score.astype(float),gsdf.param_n_estimators.astype(float))[0][1]
```

```
Bootstrap Correlation: -0.314288279915
Sample Split Correlation: -0.912616656875
N_Estimators Correlation: 0.0711277022803
```

This helps putting a number on the effect that each parameter has on the final score. The algorithm performs very well with all types of data, and the score is always above 0.8 regardless of parameter choice.

Something else to consider in the discussion is feature importances; are the frequency features more important than time features? The feature importances are stored in the Random Forest Classifier object.

```
print sum(clf.feature_importances_[:135])
print sum(clf.feature_importances_[136:])
```

```
Sum of Frequency importances: 0.541973339566
Sum of Time importances: 0.421039101145
```

As reported by the Müller et al (2005), frequency correlation serves the main features to make predictions, however time correlation importance is still relevant to making accurate predictions for the learner, as verified by Tzallas et al (2009) whom stresses on the important features in the time series data.

Improvements

The opportunity of using channel locations was never explored. Thus advanced tools used in the field of neuroscience are not an option (Source estimation/localization, Source Information Flow, Granger Causality). Having source information, essentially shifts feature extraction and analysis to the source rather than channels and channel locations.
Other signal analysis tools were not explored, such as wavelet transforms, cross channel frequency magnitude squared coherence (if channels behave similar in the frequency domain), analysis of phases and phase locking values (essentially if channels lock into a specific phase at a frequency during a seizure).

References:

Link to competition: **https://www.kaggle.com/c/seizure-detection**

1. Rizzo G. (2013) Design and Evaluation of an Affective BCI-Based Adaptive User Application: A Preliminary Case Study. In: Carberry S., Weibelzahl S., Micarelli A., Semeraro G. (eds) User Modeling, Adaptation, and Personalization. UMAP 2013. Lecture Notes in Computer Science, vol 7899. Springer, Berlin, Heidelberg

2. Kaspar Schindler, Howan Leung, Christian E. Elger, Klaus Lehnertz; Assessing seizure dynamics by analysing the correlation structure of multichannel intracranial EEG, Brain, Volume 130, Issue 1, 1 January 2007, Pages 65–77,
https://doi.org/10.1093/brain/awl304

3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

4. Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

5. John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

6. Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

7.A. T. Tzallas, M. G. Tsipouras and D. I. Fotiadis, "Epileptic Seizure Detection in EEGs Using Time–Frequency Analysis," in IEEE Transactions on Information Technology in Biomedicine, vol. 13, no. 5, pp. 703-710, Sept. 2009.

8. Markus Müller, Gerold Baier, Andreas Galka, Ulrich Stephani, and Hiltrud Muhle, " Detection and characterization of changes of the correlation structure in multivariate time series", Phys. Rev. E **71**, 046116 – Published 14 April 2005

9.Vaishali P. Kadam, Ratnadeep R. Deshmukh" Advancements of EEG signal pattern recognition in BCI-based Applications", International Journal of Latest Trends in Engineering and Technology (IJLTET),Vol. 6 September 2015.

10.Yoon Gi Chung, Sung-Phil Kim, Min-Ki Kim,"Inter channel connectivity of Motor Imagery EEG signals for a Noninvasive BCI application", IEEE International workshop on Pattern Recognition in NeuroImaging 978-0-7695-4399-4/11 , IEEE DOI 10.1109/PRNI, 2011.

11. Introduction to Random forest by Raghav Aggiwal,Feb 28, 2017.
https://dimensionless.in/introduction-to-random-forest/