Capstone Project, Seizure Prediction Report

Kyler Connelly
January 21st, 2017

I. Definition

Project Overview

People with epilepsy often experience frequent and spontaneous seizures. There are medications epilepsy patients can take to reduce the probability of a seizure occurring if an individual knows a seizure is likely to occur. There are also safety precautions an individual can take if they know an unpreventable seizure is likely to occur. Because of this it would be very valuable to be able to predict if a seizure is likely to occur in the near future.

For this project I will be using Intracranial Electroencephalography (iEEG) data collected by Melbourne University in collaboration with Mathworks and the National Institute of Health to attempt to predict seizures for individuals with epilepsy. The data was provided to Kaggle for a competition and I got the data from Kaggle (link below). iEEG data represents electrical activity occurring in the brain and is collected off electrodes which have been surgically implanted.
https://www.kaggle.com/c/melbourne-university-seizure-prediction

Problem Statement

The data consists of iEEG data for one hour segments. Each segment either preceded a seizure (preictal) or did not precede a seizure (interictal).

The problem to be solved is to be able to use iEEG data from three patients to be able to as accurately as possible determine if for a given one hour segment a seizure is likely to occur or not. To be relevant in this medical environment the model needs to make the prediction between 5 and 65 minutes before the onset of the seizure. The data provided from Kaggle was collected in this time window according to the website.

The iEEG data is provided with a label which indicates whether or not this data preceded a seizure. Features will be extracted for the entire dataset. For each of the three patients the data will be split into a training set and a testing set. The training portion of the data will be fed into a supervised learning model with the class labels included to train the model. The fitted models will then be used to predict the labels of the test set.

Since this data is part of a Kaggle competition there is a second set of data which is provided without labels, therefore the models will separately be trained using the entire labeled dataset. These trained models will then be used to predict the labels of the unlabeled

test set provided by Kaggle and the predictions will then be submitted to get a score. The Kaggle competition score is not the primary objective of this project but is an interesting test of the solution.

Metrics

The model performance will be measured using Receiver Operator Characteristic (ROC) Area Under the Curve (AUC), which is the metric being used to measure the results of the Kaggle competition. Sklearn provides a function for calculating the ROC AUC score which is what is used for measuring the labeled data set's performance. For the unlabeled dataset the score is determined by submitting the predictions into Kaggle where ROC AUC is applied.

The ROC is the curve on a graph where the x-axis represents the false positive rate and the y-axis is the true positive rate, of the predictions. With ROC the classifier returns the probability of each instance belong to one class or another. To get the ROC curve the threshold which determines the classification (probabilities below this threshold are one class, above the threshold are another) is varied between 0 and 1. When the threshold is varied between the 0 and 1, the false positive rate and true positive rate will also vary between 0 and 1. This creates the ROC curve.

The ROC AUC is simply taking the area under the ROC curve where the higher the number the better. The maximum ROC AUC score is 1, and the minimum is 0.

Calculating ROC is beneficial because one can optimize the model without knowing ahead of time the cost of false positives. In our application a false positive may result in an individual taking medication unnecessarily. Once the cost of a false positive is determined, which may involve a complex set of inputs, the threshold can be chosen using the ROC curve. By maximizing the area under the ROC curve we are solving for the most optimal set of solutions without needing to choose the threshold ahead of time.

II. Analysis

Data Exploration

For each patient two folders are provided from Kaggle, a training folder and a test folder. The training folder contains labeled iEEG data intended to be the training data for the Kaggle competition. The test folder contains iEEG data without labels which is intended to be used to test the fitted model for the Kaggle competition. For now I will be focusing on just the labeled training data.

From the labeled data, each patient has the same number of preictal instances, 25, but a varying number of interictal instances is provided for each patient. In all cases a single 1 hour instance is divided into 6 files, each representing 10 minutes of iEEG data in sequential order as indicated by filename and a sequence number inside the file structure. Therefore the 25

instances of preictal data for each patient, for example, is provided in the form of 150 separate files.

Each file of 10 minute iEEG data contains data for 16 separate electrodes, or channels. Each of the 16 channels is a voltage signal measured at 400 Hz. Therefore each file of 10 minute iEEG data contains 3,840,000 (16 channels x 10 min x 60 sec/min x 400 Hz) samples of data. The data will be preprocessed into fewer features before being fed into the learners.

The following table is the total file count for the 3 patients' training data:

	Patient 1	Patient 2	Patient 3
Preictal Files	150	150	150
Interictal Files	570	1836	1908

Table 1, Labeled File Set

Many of the files contain signal dropout where for some period of time all channels read out as zero. How these are handled is discussed in later sections when going over the feature extraction process.

The total size of all the labeled training data is quite large, approximately 29GB. More information about the dataset and the format can be found here: https://www.kaggle.com/c/melbourne-university-seizure-prediction/data

Exploratory Visualization

The following figure is an example of a typical 10 minute segment of data found in a single file:

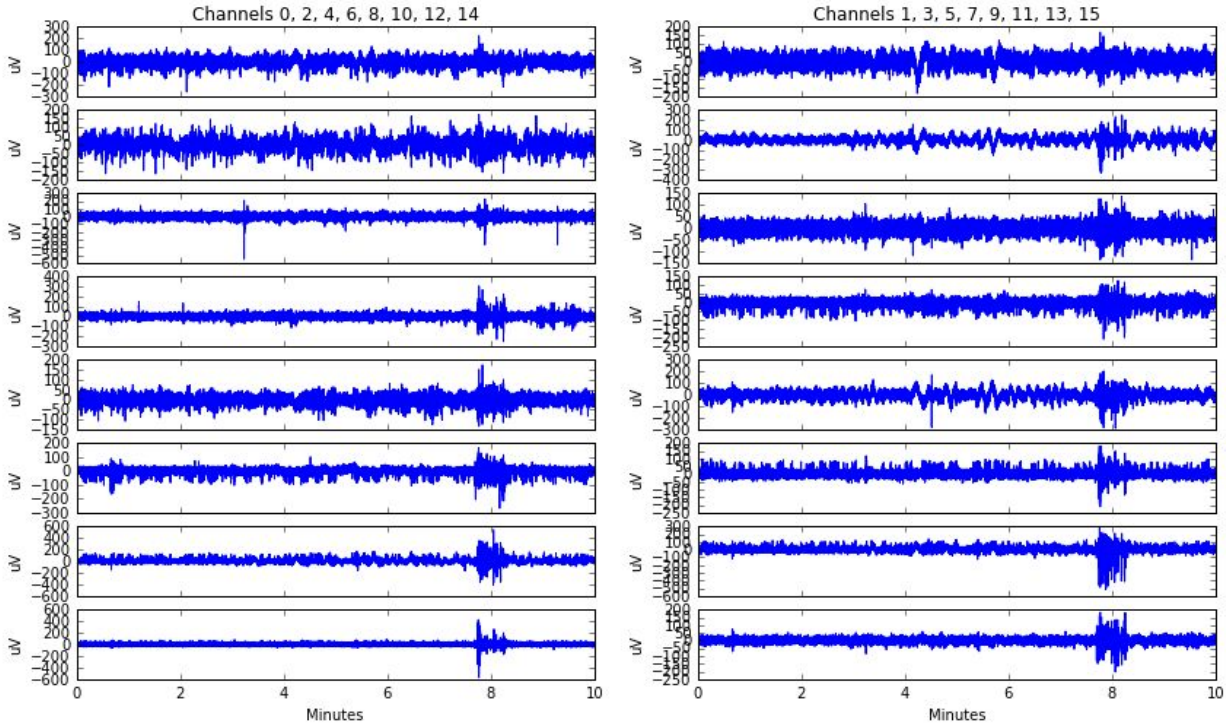

Figure 1, Raw Data Example

All of the files look something like the figure above with the exception of the ones which contain some signal dropout. Most of the literature one will find on seizure prediction suggests frequency domain analyses which is the direction I went with this problem. For example: http://www.sciencedirect.com/science/article/pii/S1388245709005264

The following figure is the Fast Fourier Transform (FFT) of the same dataset with two additional steps, first log base 10 was taken, then all channels were individually normalized:

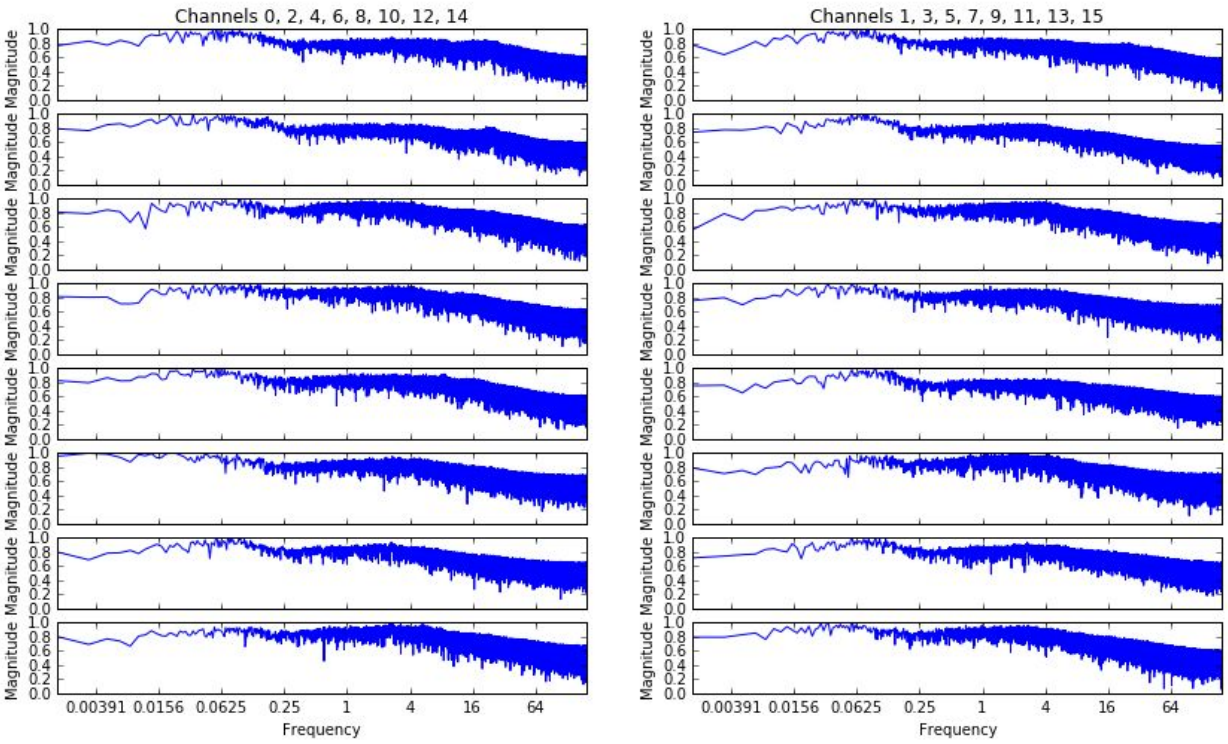

Figure 2, FFT of Raw Data Example

The x-axes on the previous figure are scaled with log base 2. This is done because the features to be extracted will provide more resolution at the lower frequencies than at the higher frequencies. Taking the FFT converts the signal from a time domain representation to a frequency domain representation, which is what we need if we are going to extract features from the data based on frequency. The reason for using features derived from the frequency domain will be explored in later sections.

The FFT of the channels is still way too much data to be used as the feature set. The last processing step is to create a number of frequency bins where each bin represents the sum of all the magnitudes between two frequencies. Many different combinations were tried but it was found that simply starting will a small bin size for the lower frequencies and doubling the bin width for each higher frequency bin above it worked the best. I started with the first bin starting from 0 Hz to 0.1 Hz, the next from 0.1 Hz to 0.2 Hz, followed by 0.2 Hz to 0.4 Hz, and so on with the last bin representing from 51.2 Hz to 102.4 Hz. This method resulted in 11 bins. The following figure is the same file as before except the frequency bins have been extracted:

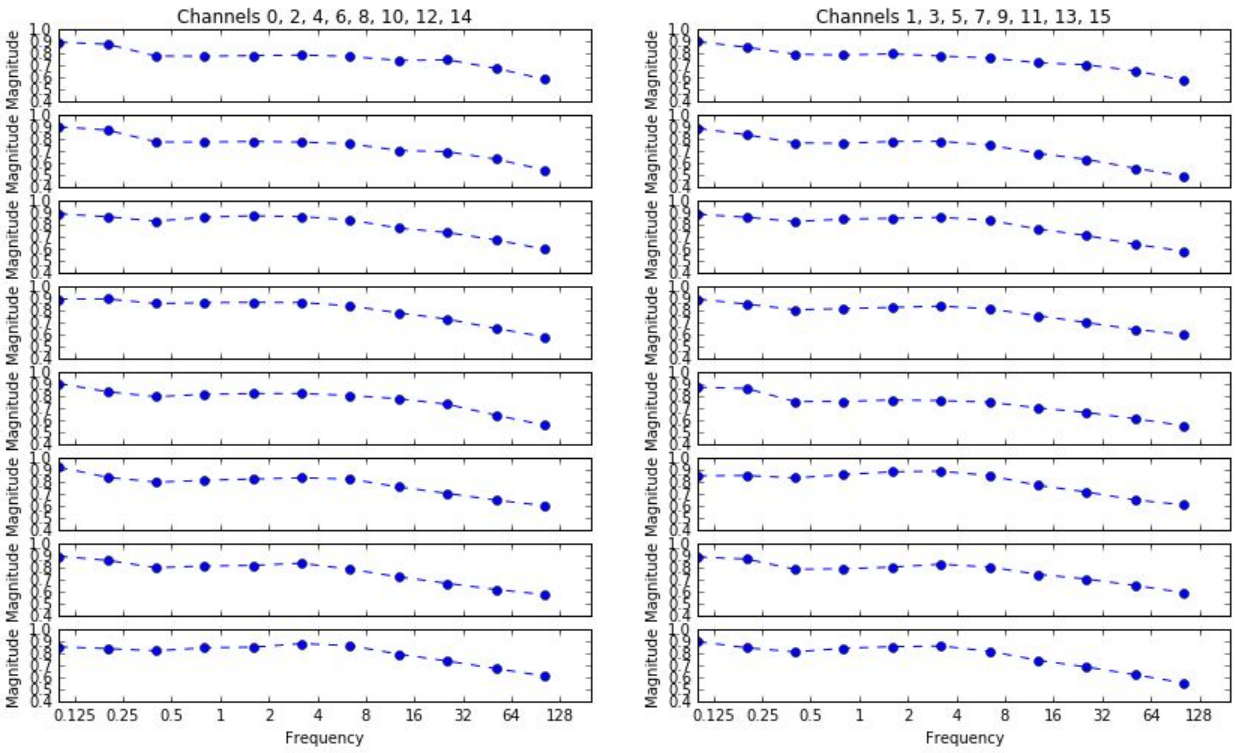

Figure 3, Summed Bins of FFT Example

In the previous figure the markers were placed on the upper frequency range of the bin, and as with the previous figure, the x-axes are scaled to log base 2. In addition to summing over a frequency range for each bin, the total sum was also divided by the width of the bin. For example the highest frequency bin representing frequencies from 51.2 Hz to 102.4 Hz was divided by 51.2 (102.4 - 51.2). This was done because if not then the width of the bin would overwhelmingly determine the final bin magnitude, which is not the point. The bin magnitude should be determined by the presence/magnitude of the frequencies inside the bin.

The frequency bin sums is the final form of the features which was used to train the models. With 11 bins and 16 channels, this results in 176 features per file, which is reasonable for some learners.

This processing was done to all files used for training and testing.

Algorithms and Techniques

Since I decided I was going to derive the features from the frequency bins, it was necessary to convert the raw data from time domain to frequency domain. The raw data is in time domain, meaning it is defined by values with respect to time. In frequency domain the values are defined with respect to frequency. When a signal is represented in the frequency domain it is represented as a collection of sine waves.

The figure below illustrates how a signal can be represented as sum of sine waves:

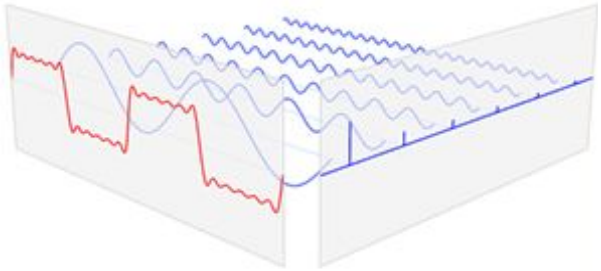

Figure 4, Example of Time Domain and Frequency Domain Representation

In the above figure the red signal on the left is the time domain signal. The set of sine waves in the middle/background is equal to the time domain presentation (red) if all summed together. The blue signal on the right (with the one straight line and a set of perpendicular lines of varying length) is the frequency domain representation of the original signal. The frequency domain representation is simply the magnitudes of the sine waves. These magnitudes is all that is necessary to represent the signal in frequency domain because we know that the shape of each one of those frequencies is a sine wave, as illustrated. Therefore when the FFT of the data is taken, the transformation illustrated in the previous figure is being done to all the iEEG data.

Once all the data is processed into features a variety of learners were explored. Of the explored learners only one type of learner was chosen to use on all three patients. The idea behind only using one type of learner is the solution may be more likely to generalise more effectively the less customization needs to be done to it for each patient. Additionally when the learners are explored they will not be tuned so one learner performing well for only one patient could be a fluke. Therefore only one type of learner was chosen based on what generally performed decently and consistently for all three patients.

Support Vector Machines (SVM) with a linear Kernel were chosen. SVM was chosen because they often perform well in complicated domains and high dimensional space. In this case there are 176 features and 288 instances for training and testing each learner. When exploring learners SVM performed consistently well for all three patients. This will be discussed more in later sections.

The SVM works by representing the training data as points in space. In training, the SVM will draw a straight line between the two classes of points while creating the largest gap possible between both sides of the boundary and the adjacent data points.

Below is an example of how an SVM would separate the two classes (red squares and blue circles) where the solid black line is the actual boundary and the solid red and blue shapes closest to the boundary are the ones who actually influence the placement of the boundary because they represent the minimum margin between the two classes.

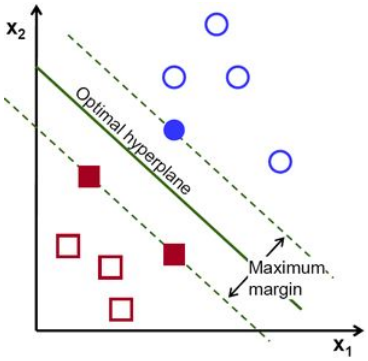

Figure 5, Example Simple SVM Fitting

However one can imagine a scenario where the two groups are not reasonably separable by a straight line. In these cases the SVM can lift the data into a higher dimension by applying some function, or Kernel, where it then can reasonably separate the data by a flat plane. Below is an example of some data being brought into a higher dimension:

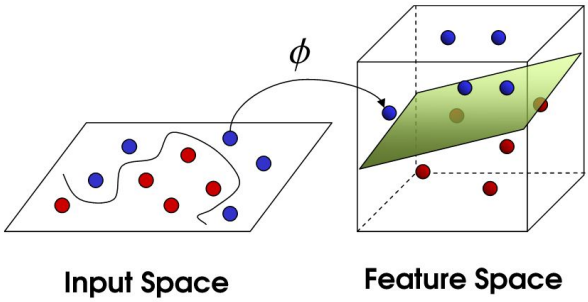

Figure 6, Example HIgher Dimensional SVM Fitting

The SVM does not have to actually calculate the datapoints' locations in the higher dimension space. If the SVM applies a similarity function between the points it is effectively doing the same thing without actually determining coordinates, which is computationally expensive.

Once the SVM has reasonably separated the data with some model, the model can then be used to predict future outcomes. Each SVM will also be individually tuned for each patient. This process will be discussed in more detail in later sections.

Benchmark

The unlabeled test set provided from Kaggle is 50/50 preictal and interictal according to Kaggle website. Therefore the subset of data pulled from the entire labeled dataset was also pulled at the same ratio.

If all the labels were predicted to be either all preictal or all interictal, the score would be 0.5 using ROC AUC as the scoring method. Likewise if a random label was predicted for each test file one would expect to score about 0.5. Therefore the benchmark I am setting for my model is to score above 0.5.

III. Methodology

Data Preprocessing

Since I wanted to be able to use my trained models to make submissions into the Kaggle competition, I needed to keep the testing data in mind when training the models. The testing data is not given with a sequence number, meaning I am unable to relate the 10 minute segment files into sequences of 6 files representing one hour of data. Because of this I chose to treat each 10 minute file independently from the rest of the 5 files in the same sequence, with one exception where the files were not treated as completely independent.

During this project the training file set from Kaggle in ***Table 1*** was split into training and testing sets at various times to test different models. When it is split, if part of a six files sequence goes into the training set and the other part of the six file sequence goes into the testing set, the model will easily learn the similarities between the set of 6 files and be able to predict the labels in the testing set very easily and the model will score nearly perfectly on the test data. However, when this model is used to predict the other unlabeled test set from Kaggle it will perform very poorly because the model overfit to the specific characteristics for each set of 6 files and did not effectively generalize to finding the differences between preictal and interictal.

Because of this the files are always kept in order and in sequence throughout the processing. When it came time to split the files into training and testing it was easier to cleanly split along sets of 6 files when they are in order. After the split the files can be shuffled in place as independent train and test sets. Many time throughout this project the models would score near perfect and in almost all cases this was because I had made a mistake with keeping the sequences of 6 together.

Once the features have been extracted for the entire dataset and split into training and test sets, a variety of supervised learners from sklearn were experimented with to try to find the most appropriate learner for this application. These will be discussed more in later sections.

The most common issue with the data provided is the signal dropout instances. One approach would be to remove all the instances with significant signal dropout. However, there are files in the unlabeled test set from Kaggle that also have signal drop out and I would like my models to be able to attempt to predict those.

Since the features are derived from the FFT of the signal, having sections of zero in the signal do not greatly affect the FFT other than the obvious problem that some potentially important data is missing. This is demonstrated by the following example.

The following is an example of a single channel of iEEG data which had a large section of drop out:

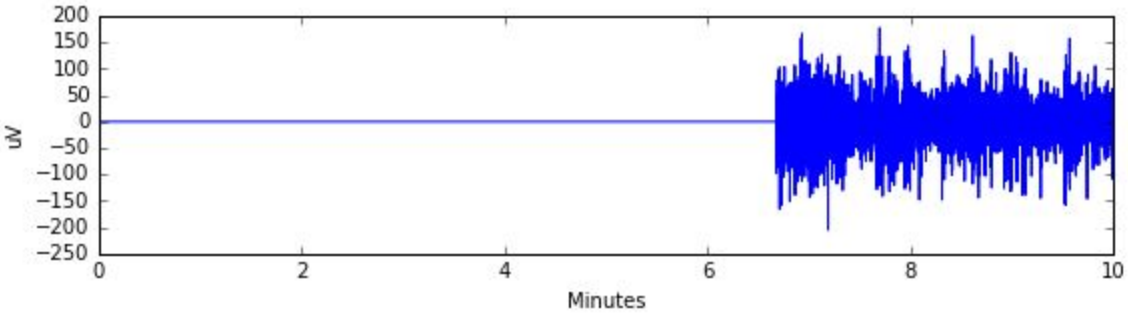

Figure 7, Signal Dropout Example

The previous figure shows the data was read as all zero for a little over 6 minutes.

The following figure graphs together the FFT of the original data plus the FFT of the data with all zero's removed:

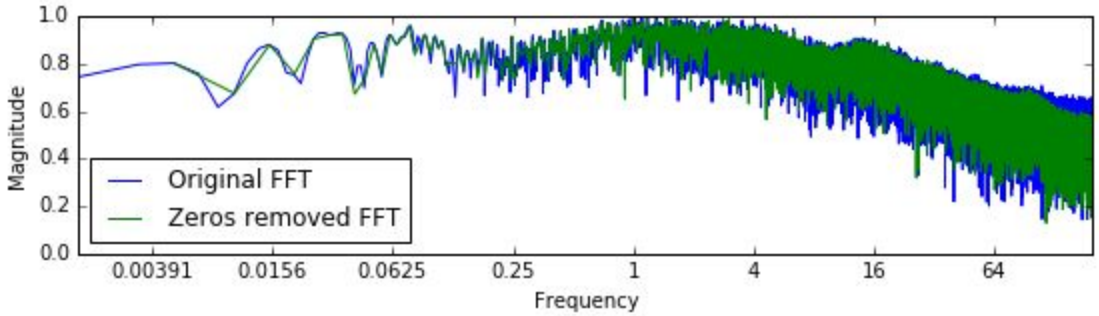

Figure 8, FFT of Original vs Trimmed

As shown in the previous figure there is not a significant change in the FFT resulting from the zeros being removed from the data. This is because when the FFT of the section of zeros is taken the result will simply be more zeros in the frequency domain and will add nothing to the rest of the spectrum. The differences will be even less significant once the FFT bins are calculated. Because of this I chose to not remove the zeros from the instances with signal dropout and simply process their FFTs like normal.

There were also files where the entire 10 minute segment of data for all channels is zeros. For these cases I simply removed the files from any training set since they will not be helpful at all.

In addition to addressing the instance with signal dropout, there was one file which for some reason would always failed to load into the workbook. This file was one of the preictal files for Patient 1. Because of this file I removed all 6 of the files from that sequence, which left 144 preictal files for Patient 1. To keep the file count the same between patients I chose to use 144 out of the 150 preictal files for each of the other two patients as well.

The final piece of data preprocessing which needed to be done had to do with the Kaggle competition. At some point during the competition Kaggle sent out a list of files which were determined to be "unsafe" for use in the competition. For the labeled set provided by Kaggle, only a subset of interictal files for each patient were deemed unsafe. This information

was provided in a .csv file. I wrote a small piece of code which filtered these files out of the main directory I used so they would not be chosen when selecting files for training and testing.

Implementation

The most complicated parts of the project involved properly extracting and handling the features, especially the splitting of the training and testing sets which was necessary at various points in the project. These were discussed previously.

As shown in ***Table 1***, there are many more interictal files than there are preictal files. However for the Kaggle competition the ratio of preictal files and interictal files is 50/50 for the unlabeled test data. I wanted my models to perform optimally with this ratio so I used the same ratio for training the models. Therefore 144 preictal files and 144 interictal files were used for model training, per patient, or 864 files for all three patients.

Once the features were extracted and any outliers removed a variety of models were tried without any tuning to get an idea of what models might work. A Stratified KFold cross validation was used to evaluate the various models using 75% of the selected labeled data for the KFold. Stratified KFold was used to keep the preictal/interictal ratio to 50/50 for reasons previously discussed. The other 25% was held out to use only for testing the final tuned models.

With all the results of the models, one was chosen from the list to tune to each of the three patients. Below is a list of all the models tested with KFold cross validation:

- SVC, Linear
- SVC, RBF
- SVC, Poly
- SVC, Sigmoid
- Logistic Regression
- KNN Classifier
- Decision Tree Classifier
- ADABoost Classifier
- Gaussian Naive Bayes
- Random Forest Classifier

Ten classifiers were tested in all. As previously discussed, only one type of model was used for all three patients. In testing the Support Vector Machine Classifier with a Linear Kernel generally performed the best for all three patients. Below is a table of all the results for all the untuned models tested:

	SVC Linear	SVC RBF	SVC Poly	SVC Sig	Log Reg	KNN	Dec Tree	ADA-Boost	GNB	Rand Forest
Pat 1	**0.685**	0.523	0.519	0.500	0.634	0.676	0.648	0.579	0.486	0.648
Pat 2	0.704	0.519	0.519	0.500	0.667	0.676	0.620	**0.736**	0.551	0.657
Pat 3	0.694	0.519	0.546	0.500	0.681	**0.736**	0.644	0.690	0.542	0.685

Table 2, Untuned Model Comparison

The previous table shows the Support Vector Machine with the Linear Kernel was consistent across all three patients and performed well.

Refinement

Once it was determined that the Support Vector Machine is likely the best fit for this dataset the last step is tuning each model for the corresponding patient. I reused some code from a previous Udacity project, with some modifications, to visualize the tuning process.
In the case of the SVC with a Linear Kernel, there is really only one parameter that makes any difference and that is the Penalty Parameter, C.

Below are the graphs for the 3 patients where C is varied from 20 to 780. The green and red regions represent 1 standard deviation away from the mean. In all cases 75% of the data was used and that was folded 8 times using stratified K Fold to preserve segment grouping as previously discussed. The folds were then individually shuffled.

Below is the chart for Patient 1:

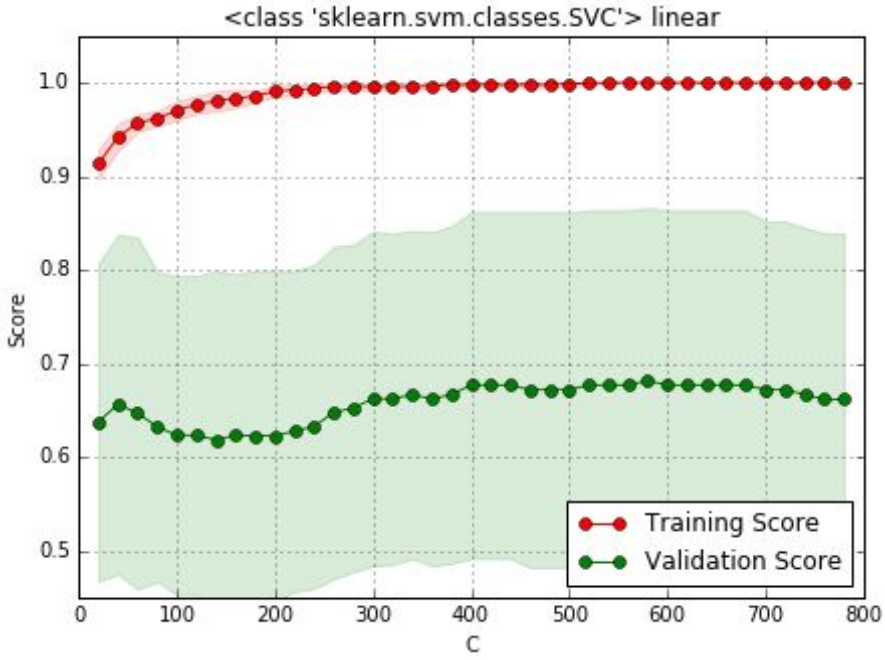

Figure 9, Patient 1 Model Performance, Varying C

The previous chart shows the validation score reaches maximum around 580.

Below is the chart for Patient 2:

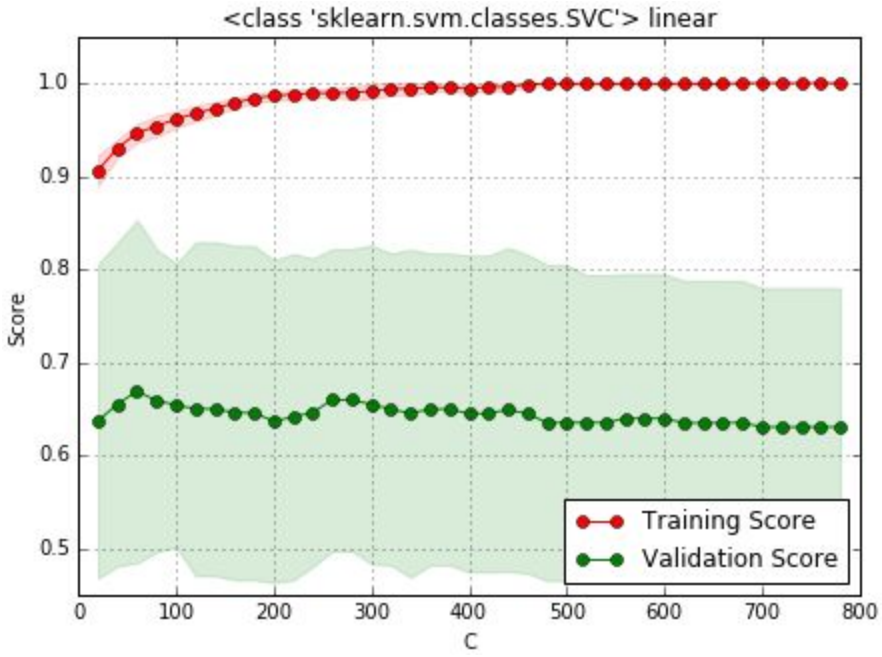

Figure 10, Patient 2 Model Performance, Varying C

The previous chart shows the validation score reaches maximum around 60.

Below is the chart for Patient 3:

Figure 11, Patient 3 Model Performance, Varying C

The previous chart shows the validation score reaches maximum around 420. For the Kaggle competition, since the additional unlabeled data is then the test set, this step was repeated

except instead of just 75% of the labeled data used, 100% of the data was used. In that case the optimum C values were found to be 180, 130, and 20, for the Patients 1, 2, and 3, respectively. This will be discussed more later.

Once it was decided to use a Support Vector Machine with a Linear Kernel, the C value optimization was the only optimization step.

IV. Results

Model Evaluation and Validation

The three SVM's with their corresponding C values were fitted to the same 75% of the data which was used to explore and tune the models. The fitted models were used to predict the labels of the other 25% of the labeled data. Below are the results:

	Patient 1	Patient 2	Patient 3
ROC AUC Score	0.745	0.655	0.619

Table 3, Final Model Scores, Labeled Data

As previously discussed, the models were refitted with C values which were derived using 100% of the labeled data. These models were used to predict the unlabeled test data provided by Kaggle. The unlabeled dataset was much larger than the set I used to fit the models. The following table shows the number of unlabeled test files provided by Kaggle for each patient:

	Patient 1	Patient 2	Patient 3
Unlabeled Files	216	1002	690

Table 4, Unlabeled Test Data

Using the fitted models as described, the score using the unlabeled test data from Kaggle was 0.741. The screenshot below from a submission, post deadline:

Figure 12, Kaggle Score/Rank, Post-Deadline

The final leader board consisted of 478 teams.

Justification

I aimed to be able to score greater than 0.5 and I have achieved that. When I originally submitted my project proposal the benchmark I was aiming for was 0.7. At the time I did not realize how challenging this project would be but I am glad I was achieve that with the Kaggle score. The final high score for all competitors on the Kaggle competition ended up being 0.807.

Additionally the average score for the labeled data was 0.673, which reached my goal.

Conclusion

Visualization

The below graph shows the actual ROC curves for each patient:

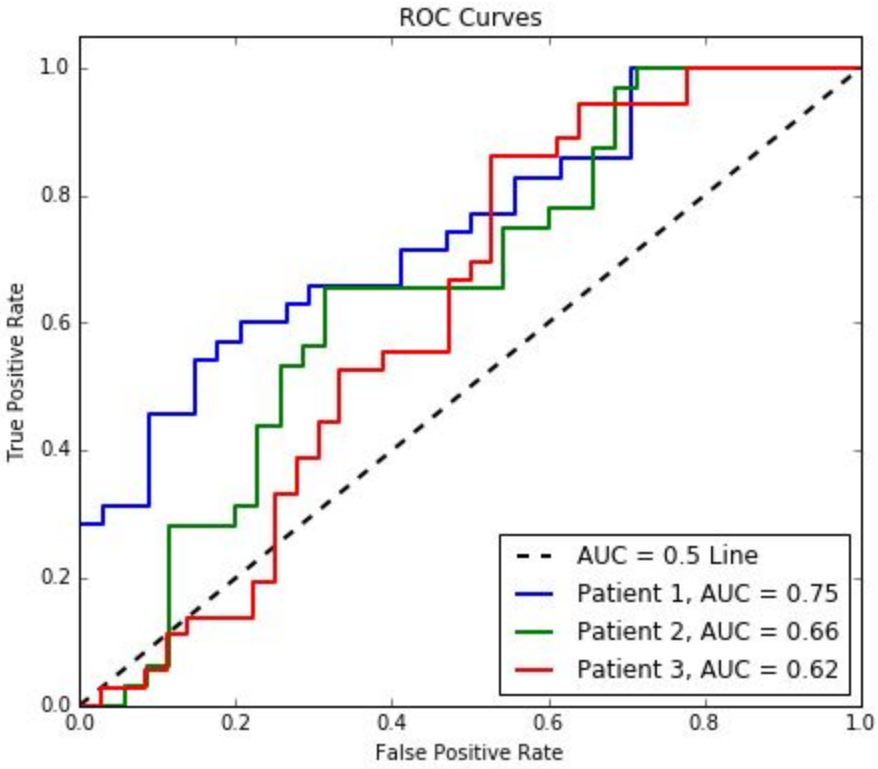

Figure 13, Patient ROC Curves

The previous graphs were generated using the results from testing on the 25% held out labeled data. The curves provide a good visual of the trade off between false positives and true positives. It is particularly meaningful in an application where true positives means a patient can prepare/prevent a seizure and a false positive means the patients prepared for a seizure unnecessarily or took medication to prevent a seizure unnecessarily.

Reflection

I found this project very challenging but I am happy with the end result and feel I have gotten a good grasp of how to solve a problem like this using machine learning techniques. I realized that I was obsessing about the Kaggle competition score too much while the competition was still open. After it ended I was able to become more focused and go back over everything and make sure I understood each step and ensure it was done as intended and I was able to make big improvements.

Since this project is focused on seizure prediction I am happy that the solution I came up with ended up being something that could actually be implemented in the real world on a wearable or something in that field.

I did not realize until I was researching ROC AUC to improve my explanation in this report that I had been calculating the final model score and submitting the results to Kaggle incorrectly. This entire project I have been submitting the results to Kaggle with the final classifications (0 or 1), not the probability (any number between 0 and 1). Once I corrected this mistake all results increased dramatically. I am glad the report reviewer insisted on a better explanation of ROC AUC otherwise I would have not realized this.

Improvement

One of the challenging parts of this project was the small dataset size of preictal data. Looking through the forums on Kaggle for this project you can see that many people split the 10 minute segments into smaller pieces as a way of creating more instances. I did not try this but it may help to improve the models.

Another common technique people tried was to calculate time series correlation data between the channels and adding that to the feature set. I did try this, however when combining time series correlation data with my FFT bin data the scores only went down slightly. This may only be useful when primarily depending on additional time series features which I did not use.

Another approach I tried was to do some research about seizure prediction and in many cases there were papers claiming that particular frequencies were good indicator. I then tried increasing the FFT bin resolution (reducing bin size) around these frequencies of interest. This also resulted in the scores going down slightly. I believe the papers I had found might have been referring to a particular patient and was not a generalizations.

Obviously there is a better solution that exists, since there are higher scores on the Kaggle competition. However it is also possible the higher scores are overfitted to those three patients and in a real world setting those models may not perform well.