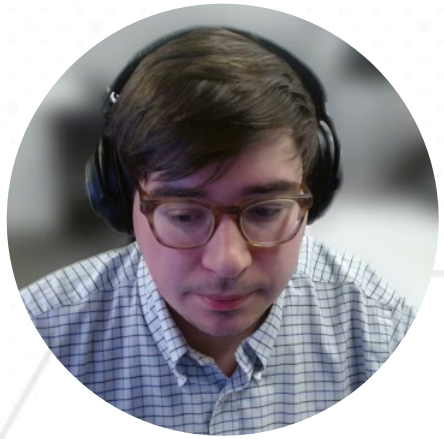
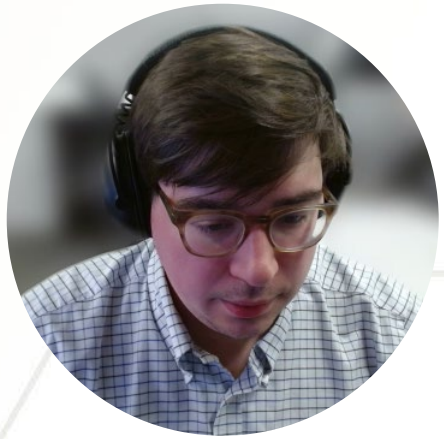
Alzheimer's & Frontotemporal Dementia (FTD) Classification from EEG Data

Sandia National Laboratories
Brian Keith (bkeith9@gatech.edu)
2024-04-15



Project Background & Motivation



Project Background

The Sandia National Laboratories' (SNL) project prompt revolved around using machine learning models to detect degradation in hardware components as they aged. The motivation for this is that physically testing components is time consuming and expensive.

This project was immediately appealing to me due to my background signal anomaly detection where at a previous employer my team was responsible for remote monitoring of vibration sensors installed on Pulp and Paper manufacturing equipment.

While it would have been easy for me to take my existing domain knowledge and directly apply it to this project, I decided challenged myself and take the concept in a different direction.

My idea was simple, the human brain is one big, complex electrical circuit, so what if we could model patterns of electrical signals in the brain to detect neurodegenerative diseases?

As it turns out, this is a burgeoning area of research in the medical field and there are researchers hard at work trying to answer exactly this question. Following discussion and approval from the SNL project lead, Stephen Smith, I moved forward with this idea.

Georgia Tech

Sandia National Laboratories

Motivation

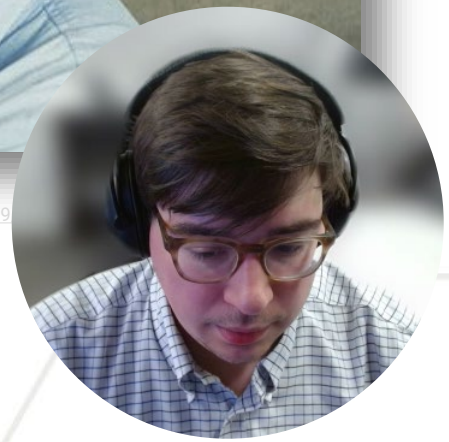
There is a growing prevalence Alzheimer's as well as other neurodegenerative diseases, and this trend is expected to continue over the next 30 years[1]. I as well as many others reading this may have a loved one or friend who they have seen affected by these diseases.

Detecting and distinguishing between different types of neurodegenerative diseases with qualitative testing can be inaccurate in many instances[2]. Due to this, there have been a significant move to qualitative machine testing for these diseases, such as electroencephalograms (EEGs) to diagnose these diseases more accurately and earlier[3].

When it comes to treatment of these diseases, while there are no current treatments that can reverse Alzheimer's disease, accurate and early diagnosis can help prepare families of the affected and extend the good quality of life years for individuals with the disease[4].

While any medical diagnosis should not be determined definitively without additional checks from medical professionals, models could be a good aid for physicians to offer them a second opinion.

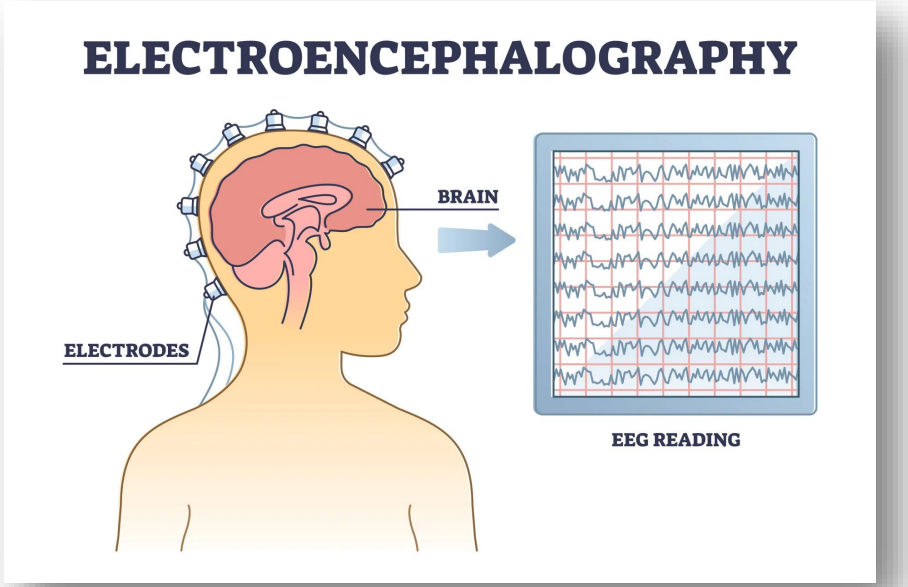
https://www.jax.org/news-and-insights/2019



EEG Overview

What are EEGs?

- An electroencephalogram (EEG) is a medical test which utilizes electrodes attached to a participant's scalp to measure electrical activity in the brain[5]. An EEG does not detect the activity of individual neurons, but instead measures the activity of small areas of the brain which can be used to indicate the level of activity in that region of the brain[5].
- These tests are used to detect a variety of neurological phenomena such as epilepsy, strokes, dementia, and other brain disfunction[5].
- While the information related to the magnitude of the waves is important, one of the key features used to analyze EEG data is signal decomposition of the data into the 5 widely recognized brain waves[3]:

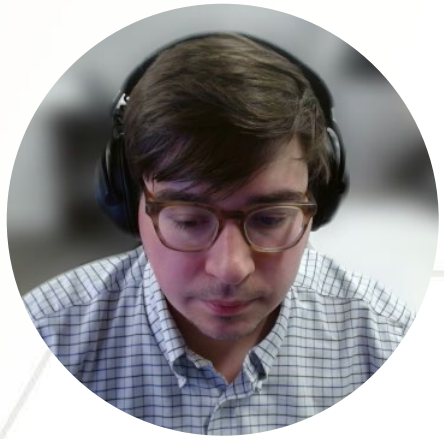


https://www.simplypsychology.org/what-is-an-eeg.html

Band	Frequency (Hz)
Delta (δ)	0.5 – 4
Theta (θ)	4 – 8
Alpha (α)	8 – 13
Beta (β)	13 – 25
Gamma (γ)	25 – 45

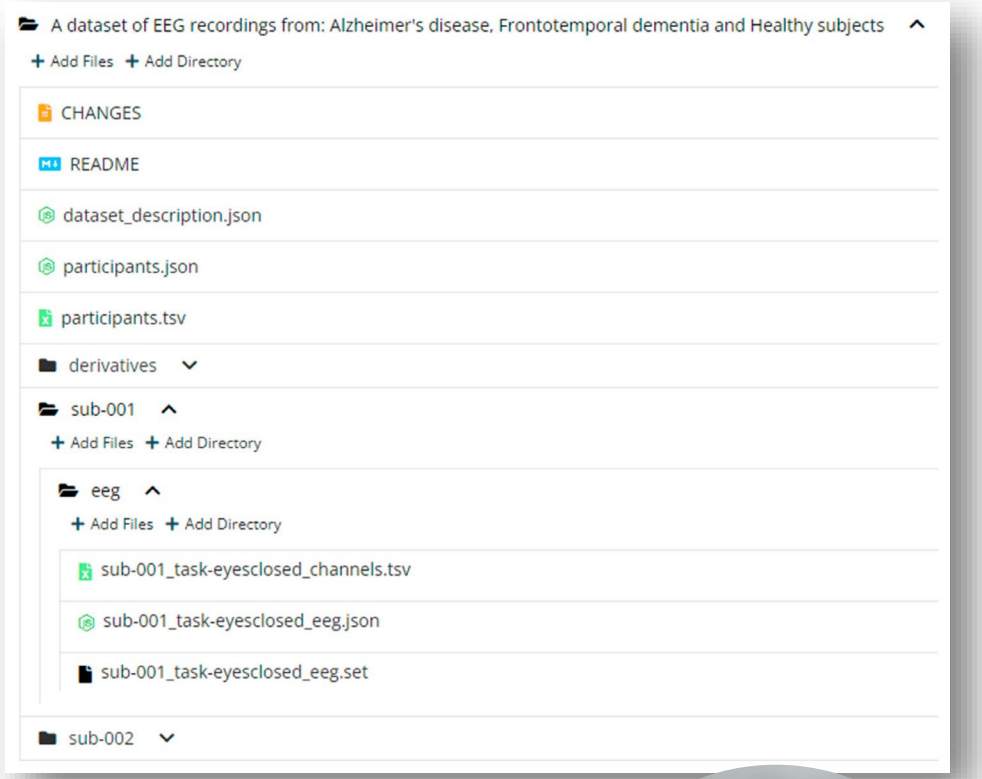
The signal data gathered via the EEG can be decomposed into these bands using techniques such as Fast Fourier Transforms to calculate the Power Spectral Density and Relative Band Power of each of the bands that makes up the overall signal.

Data

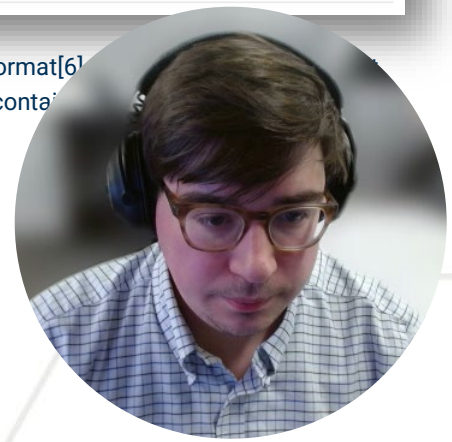


The Dataset – Overview

- After researching potential datasets, I settled on a dataset from Greek researchers[6] which was very well documented.
- The dataset includes EEG data from 88 participants, 36 with Alzheimer's Disease (A), 23 with Frontotemporal dementia (F), and 29 healthy controls (C).
- The data was provided as a ZIP file by the researchers with the folder structure shown in the image to the right. The structure, for the purposes of my analysis consisted of three different datasets:
    1. Information related to the classification of the participants (*participants.tsv*)
    2. The raw EEG data as *.set* files broken out by participant.
    3. The EEG data with some domain specific preprocessing to remove artifacts in the EEG readings such as eye movement, also provided as *.set* files broken out by participant.
- The two EEG datasets contained information from 19 different electrodes placed at various locations which were sampled at 500 Hz with a resolution of 10 μV/mm. Information related to the time in milliseconds for each sampled datapoint was also provided in the EEG data.



The structure of the dataset following the BIDS format[6] find that the *.tsv* and *.json* files for the EEG data conta *.set* files which is why the .set files were used.

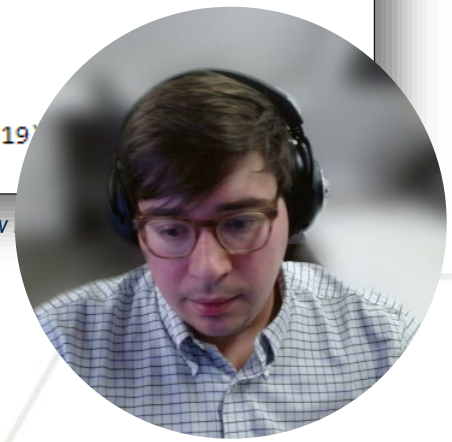


Loading & Consolidation

- Python is my language of choice and therefore, the first step of the analysis was to get the information out of the *.set* files provided for the EEG data and into something more Python friendly.
- To accomplish this a set of custom functions were developed which extracted the information out the partitioned *.set* files and loaded them into consolidated DataFrames for each dataset.
- This process involved inspecting the data structure of the *.set* files and developing an algorithm to recursively exploding the nested data into long columns.
- Following this processing, I then performed some datatype correction and exported the data to a *.pkl* file for later processing. An example of the *.info()* for the raw EEG dataset can be seen in the picture to the right.
    - Column #'s 3 - 21 are the labels associated with the 19 sensors mentioned previously.

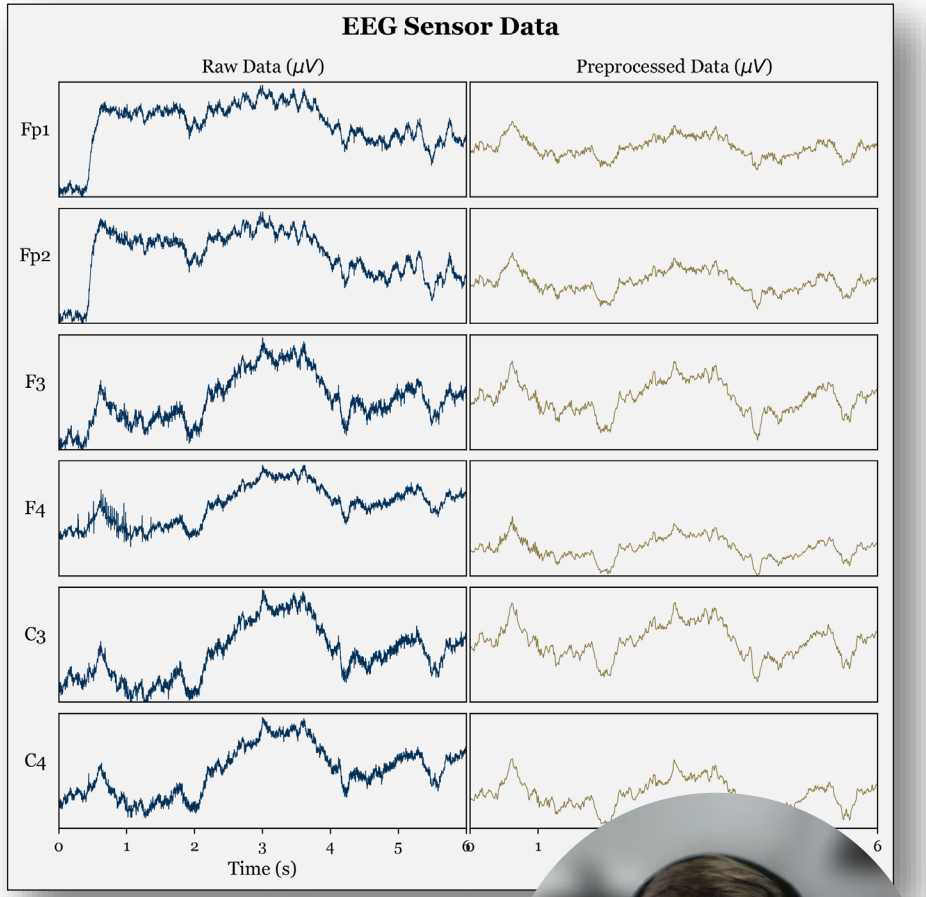
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34915560 entries, 0 to 34915559
Data columns (total 22 columns):
 #   Column          Dtype
---  ------          -----
 0   participant_id  category
 1   time_s          float64
 2   time_ms         int32
 3   Fp1             float32
 4   Fp2             float32
 5   F3              float32
 6   F4              float32
 7   C3              float32
 8   C4              float32
 9   P3              float32
 10  P4              float32
 11  O1              float32
 12  O2              float32
 13  F7              float32
 14  F8              float32
 15  T3              float32
 16  T4              float32
 17  T5              float32
 18  T6              float32
 19  Fz              float32
 20  Cz              float32
 21  Pz              float32
dtypes: category(1), float32(19)
memory usage: 2.9 GB
```

Example of the .info() returned for the Raw

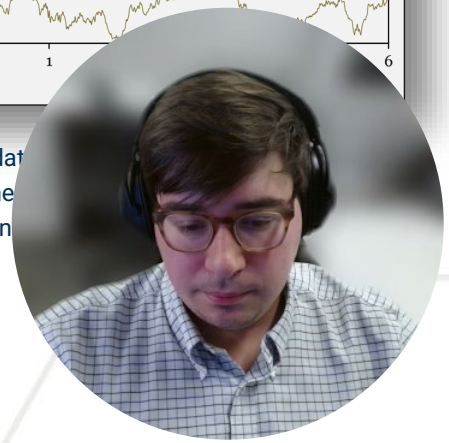


Raw or Preprocessed?

- One nuance of the data provided by the researches[6] was that I needed to decide which of the two datasets to use, the raw EEG data or the data that was preprocessed to correct a variety of data issues such as:
  - Re-referencing the electrodes based on two reference electrodes
  - Frequency range correction
  - Performing artifact rejection routines related to eye and jaw artifacts
- As seen in the same visualization to the right, the raw and preprocessed data generally follow the same general trend, but the preprocessed data does not exhibit some of the start-up lag from the first few milliseconds and, overall, the amount of noise in the signal is reduced.
- After consideration and research, I determined that some of the corrections done by the re-searchers, especially in relation to EEG artifacts, would have been time consuming and required very domain specific knowledge to perform correctly.
- After consideration and research, I determined that some of the corrections done by the researchers, especially in relation to EEG artifacts, would have been time consuming and required very domain specific knowledge to perform correctly.



Visualization of the Raw vs. Preprocessed dat
sample sensors for the first 6 seconds of the
axis for all sensors is scaled to the same ran



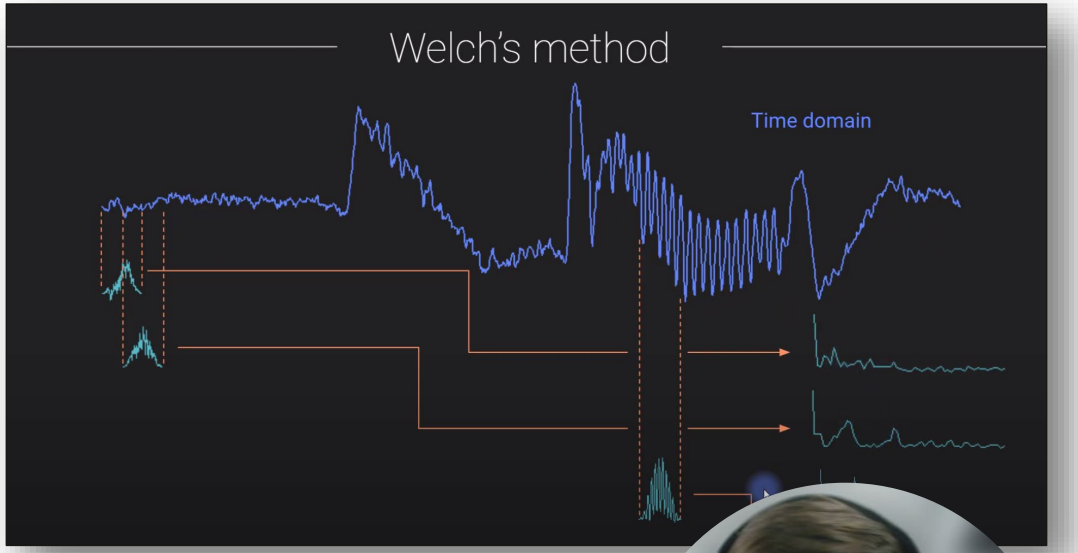
Feature Extraction

- Before I could begin any modeling of the data, I needed to preformed signal decomposition for each of the participants as well as for each of the 19 sensors.
- The idea of the signal decomposition is to take the signal data from the EEG and decompose the signal into the respective power spectral density (PSD) for the 5 different brain wave bands shown in Table 1. From the PSD, we can also calculate the Relative Band Power (RBP) – a common metric used when analyzing EEG Data[7] – of each band giving us a normalized metric between 0 and 1 for the contribution of a given PSD to the overall signal.
- To calculate the PSD of each participant and sensor combination, I utilized the Welch method[8], a widely used model in research to estimate the PSD[9]. This method works by dividing a given time series into successive blocks to form the squared-magnitude of a Discrete Fourier Transform (DFT) divided by the length of the DFT[10].
- To calculate the PSD, the *welch()* function from SciPy's signal module was utilized. After deriving the PSDs of each brain wave frequency band for each patient and sensor combination, I then calculated the RBP which can be defined formulaically as:

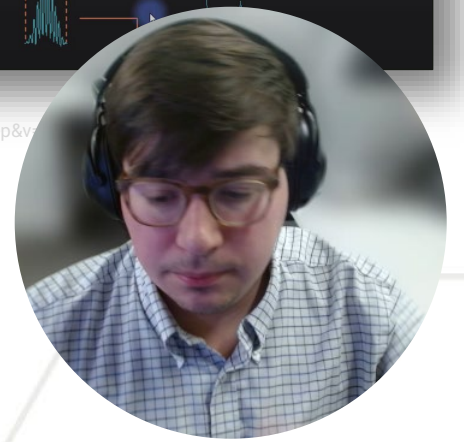
$$\boldsymbol{RBP_a} = \frac{\sum_{f_a} \boldsymbol{P(f)}}{\sum_{f=f_1}^{f_2} \boldsymbol{P(f)}}$$

Where $P(f)$ represents the PSD of the EEG signal, f_1 and f_2 represent the minimum and maximum frequencies of the EEG signal, respectively, and f_a represents the frequency range of a band[7].

Band	Frequency (Hz)
Delta (δ)	0.5 – 4
Theta (θ)	4 – 8
Alpha (α)	8 – 13
Beta (β)	13 – 25
Gamma (γ)	25 – 45



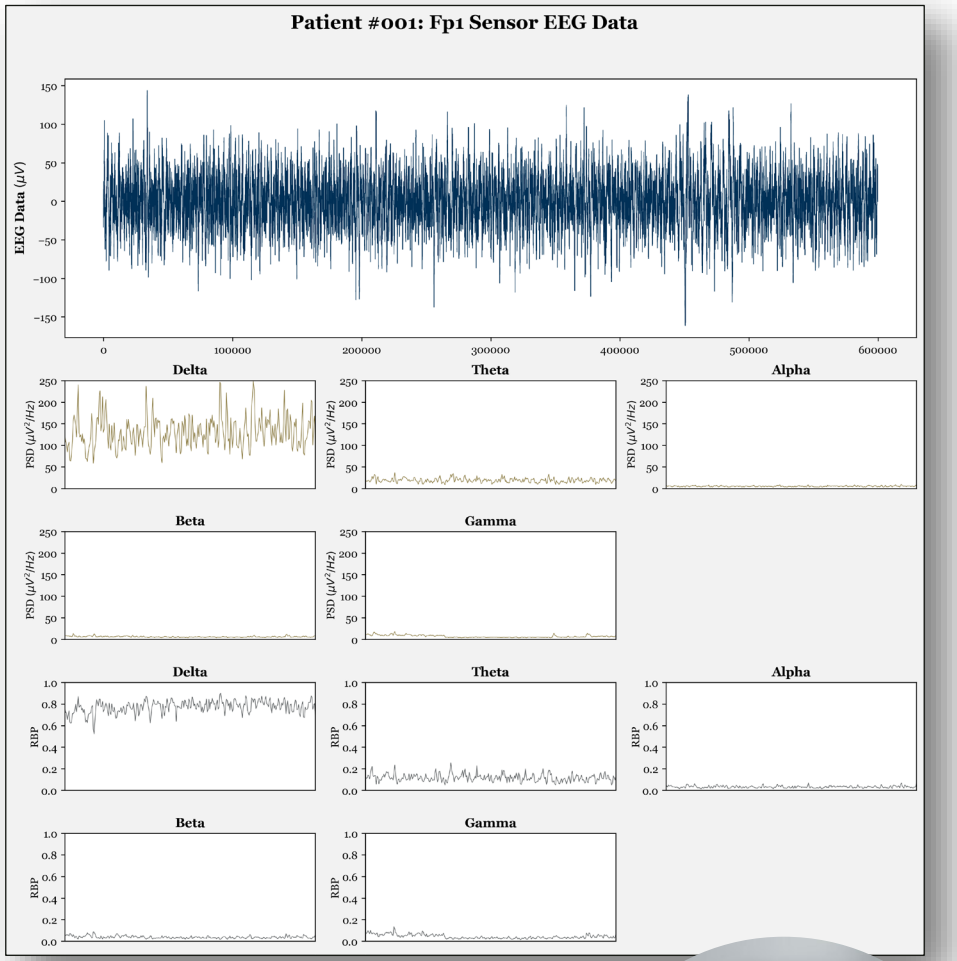
https://www.youtube.com/watch?app=desktop&v



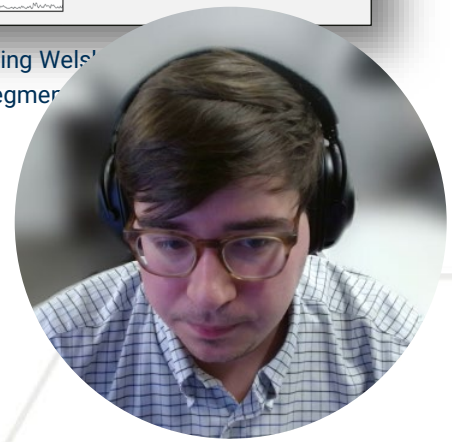
Feature Extraction

- Regarding Welch's method, most of the arguments to this function were left as the defaults including utilizing the Hanning window because it is widely used due to the good frequency resolution and reduced spectral leakage[11].
- As a part of the project I did want to experiment with two critical components of Welch's method, the window size (i.e., the length of the time segments) and the amount of overlap for each successive time segment, as these have been shown to have an impact on the signal decomposition[11].
- Using a basis of the same criteria used by Miltiadous et al. (2021) of 4000 ms segments with 50% overlap, I created datasets using 2000, 3000, 4000, 5000, and 6000 ms segments with 25%, 50%, and 75% overlap for each giving me a total of 15 different datasets to utilize.
- In total, for each of the 15 datasets, after calculating the PSD and RBP of each patient/sensor combination, I was left with 95 features (19 sensors x 5 bands) related to PSD and RBP, respectively (190 in total), for each participant.
- The table below shows the data structure for each of the datasets and the figure to the right shows and example of the signal decomposition.

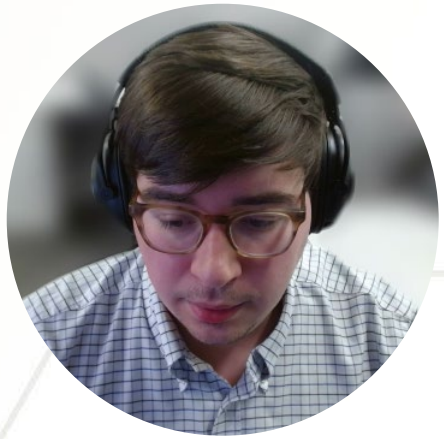
participant_id	seg_start	seg_end	Fp1_delta_psd	Fp1_delta_rbp	...	Pz_gamma_psd	Pz_gamma_rbp
sub-001	0	4000	122.883514	0.7605439	...	2.4622881	0.017862573
sub-001	2000	6000	108.59866	0.7212657	...	2.0737097	0.015800584
...	...	...	...	...	...	...	...
sub-088	782000	786000	76.35473	0.7201604	...	2.0698907	0.020465318
sub-088	784000	788000	59.0055	0.66316277	...	2.44103	0.015787695



A visualization of the decomposed EEG data using Welch's method. The data used in this figure utilized the basis 4000 ms segmen

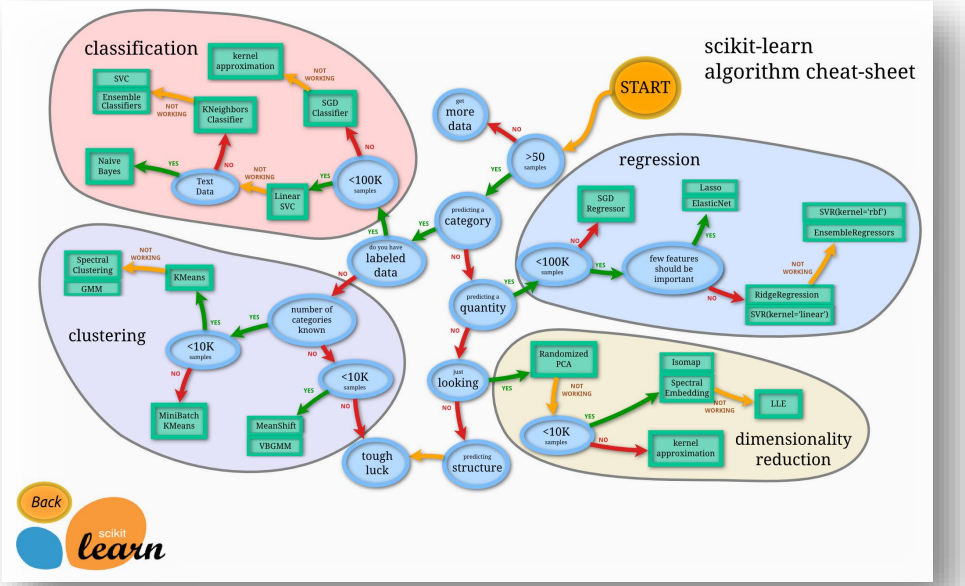


Modeling



Model Introduction

- The problem of classifying the participants into the groups of Alzheimer's, FTD, or healthy is exactly, that, a classification problem. Identifying this is the first step and while there are numerous classification models that exist, I chose to build and test four different models:
  - Random Forest
  - XGBoost
  - Logistic Regression
  - K-Nearest Neighbors (KNN)
- I also experimented with some other models during the process such as sklearn's GradientBoostingClassifier and SVC algorithms.
  - The main reason for abandoning these algorithms was due to their extremely slow runtimes which were not feasible for the later hyperparameter tuning performed.
- The chosen models fall into three different camps:
  - Tree-based (Random Forest and XGBoost)
  - Regression (Logistic Regression)
  - Clustering (KNN)
- The idea was that these models should offer an interesting and diverse set of results which could help to determine what kind of model is best suited for the problem at hand.



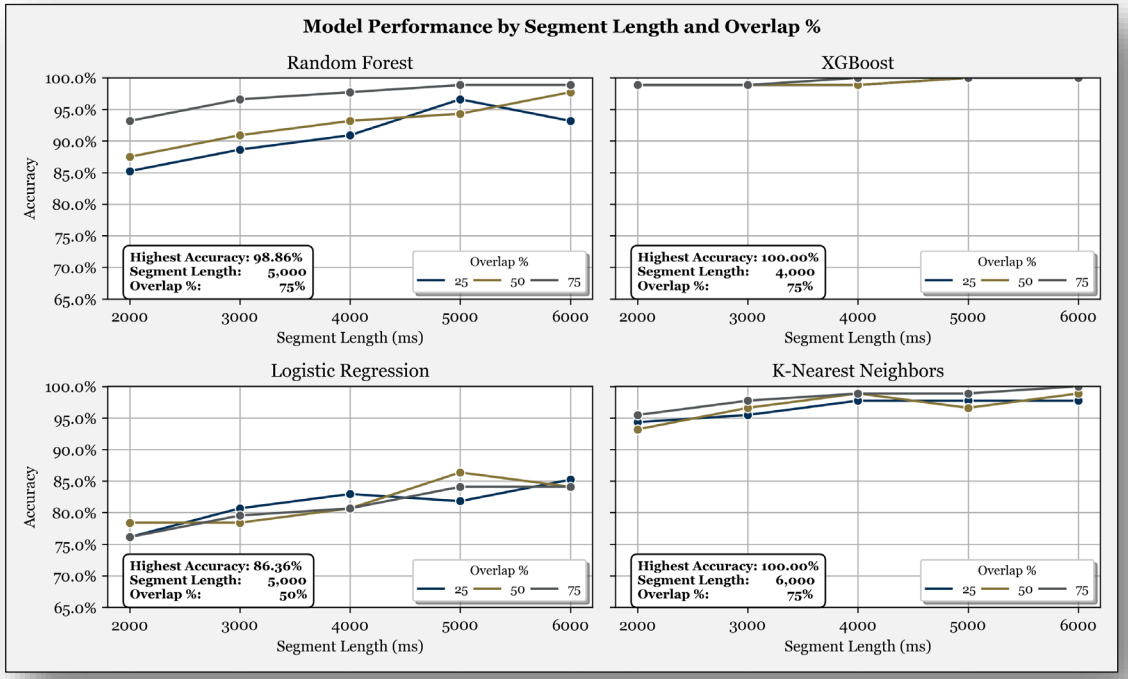
https://scikit-learn.org/stable/_static/ml_map.png

Feature Selection

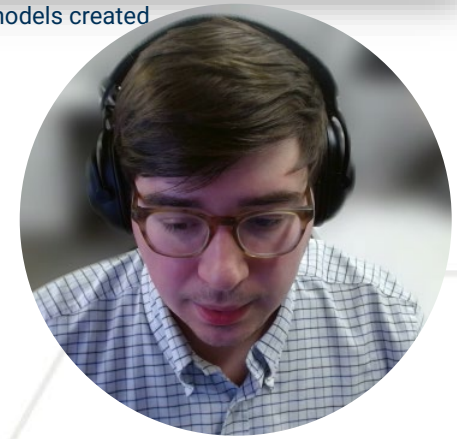
- For my problem, one of the most significant considerations I needed to work around was related to the time required train and test the various models.
    - 15 datasets x 3 feature set combinations (PSD, RBP, or both) meant 180 models would need to be trained, and before considering hyperparameter tuning which would increase the number of models by orders of magnitude.
- To start limiting the scope, I first determined that I would only utilize the 95 features related to RBP.
- The logic behind this decision was that, in my experience, normalized metrics tend to lead to better performance for the models being utilized; therefore, choosing RBP over the PSD was the better choice.
- When dealing with a large problem space, one of the best techniques is to use experience and logic to narrow that scope and test the simpler cases first. Afterall, if the 95 features related to RBP did not produce models with acceptable performance characteristics, I could always go back and test the other feature sets.

Basis Models

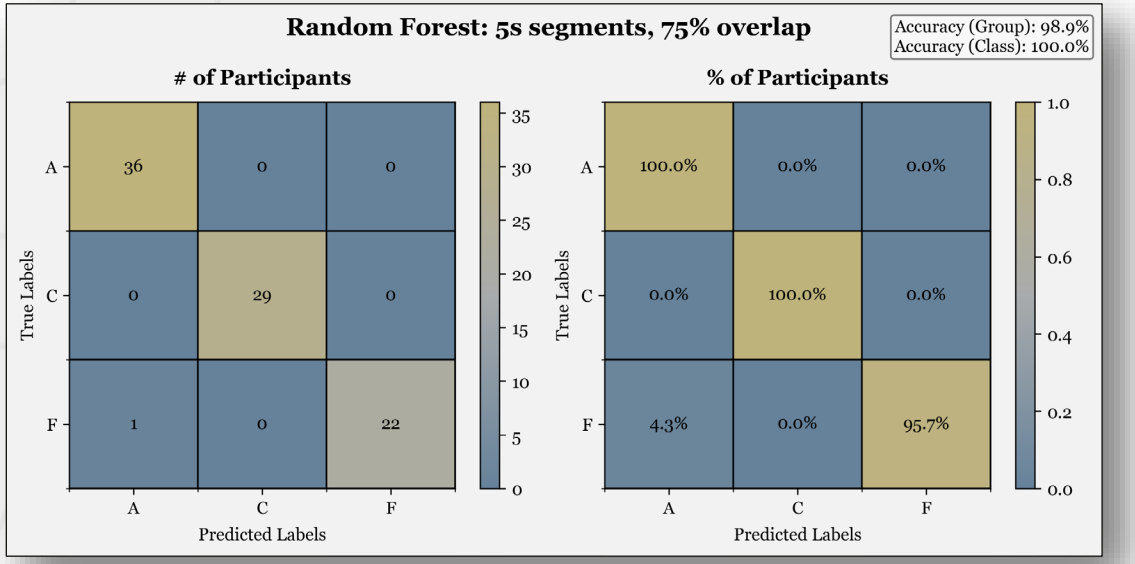
- Having selected the features to use, my next step was to determine which of the datasets to use for each model. The idea here was to determine which combination of segment length and overlap lead to the best model using the default settings from the respective libraries with no hyperparameter tuning.
- To accomplish this, I built 15 basis models for each of the 4 algorithms with all other parameters left as their respective defaults and an 80/20 train/test split for the input dataset.
- To measure the performance of these models, I chose to utilize accuracy as the metric.
- However, that the data we built the model on is organized into X millisecond segments, therefore the predictions are only a prediction based on that small segment of time. It would not make sense to judge the model accuracy this way.
- Instead, I wrote a function that would take in the predictions and then calculate the most popular label assigned to a given participant. This allowed me to have a single classification for each participant's diagnosis. The ac-curacy score was then calculated based on the number of correct and incorrect participant classifications.

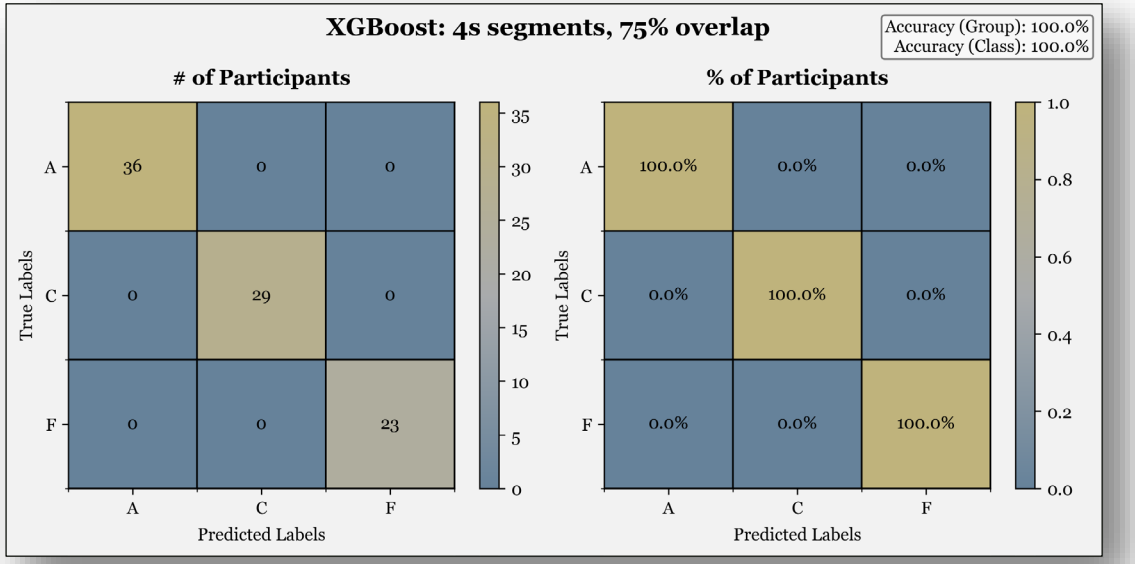


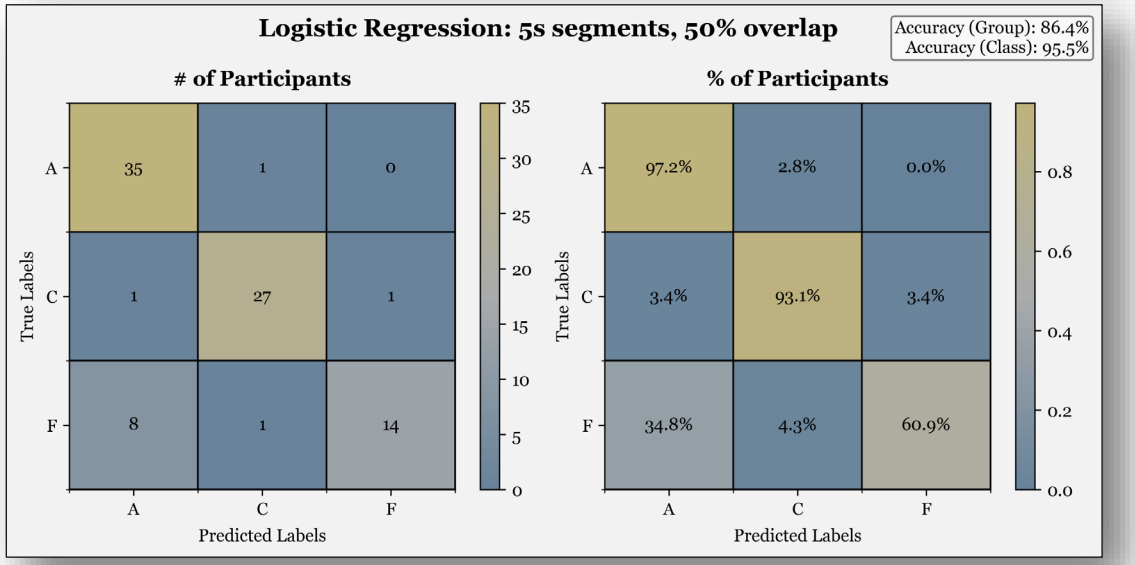
Accuracy Scores for all 60 basis models created

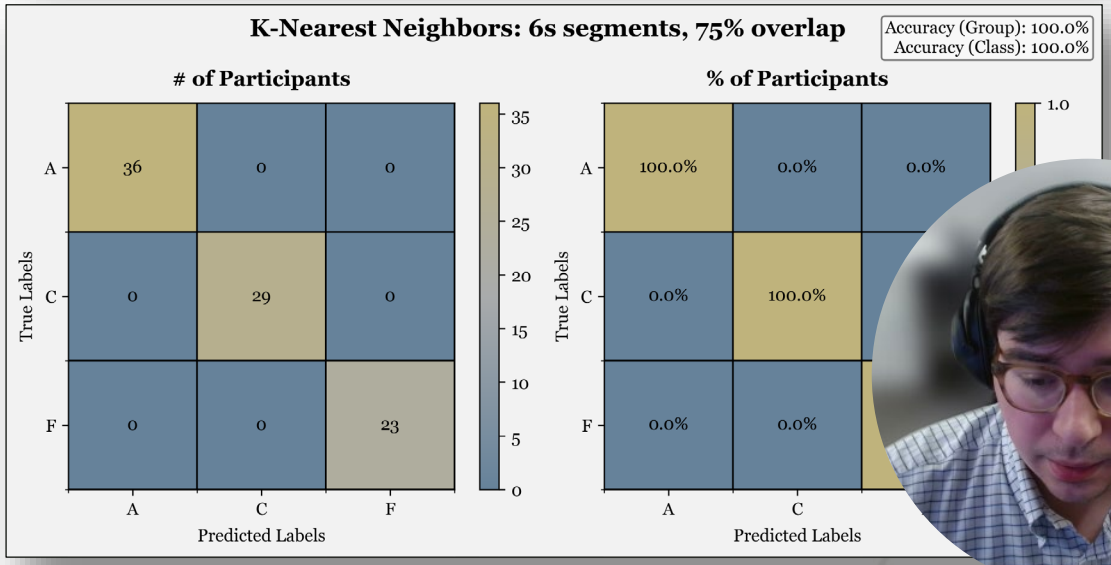


Basis Models









Hyperparameter Tuning

- After determining the datasets to use for each model, the goal became to utilize those datasets to tune and test a robust set of parameters for each model.
- To add ensure more reliable results during the tuning process, instead of using a single 80/20 train/test split dataset, K-Fold cross validation with 5 folds was also employed.
- To perform the hyperparameter tuning, I created a large grid of options for various inputs to each model, then generated a list of all possible combinations of those inputs.
- Due to time and processing constraints, not all the possible model parameters could be tested, and I instead utilized a random search and took a random subset of 100 combinations for each model and used that as a basis for the hyperparameter tuning. This resulted in of 2,000 different models (4 different models with 100 sets of hyperparameters fit across 5 folds).

Index 0 Parameters for Each Model

RF

```
n_estimators = 400
max_depth = None
min_samples_split = 10
min_samples_leaf = 2
bootstrap = True
max_features = auto
```

LR

```
C = 10
solver = lbfgs
max_iter = 300
penalty = l2
l1_ratio = 0.5
```

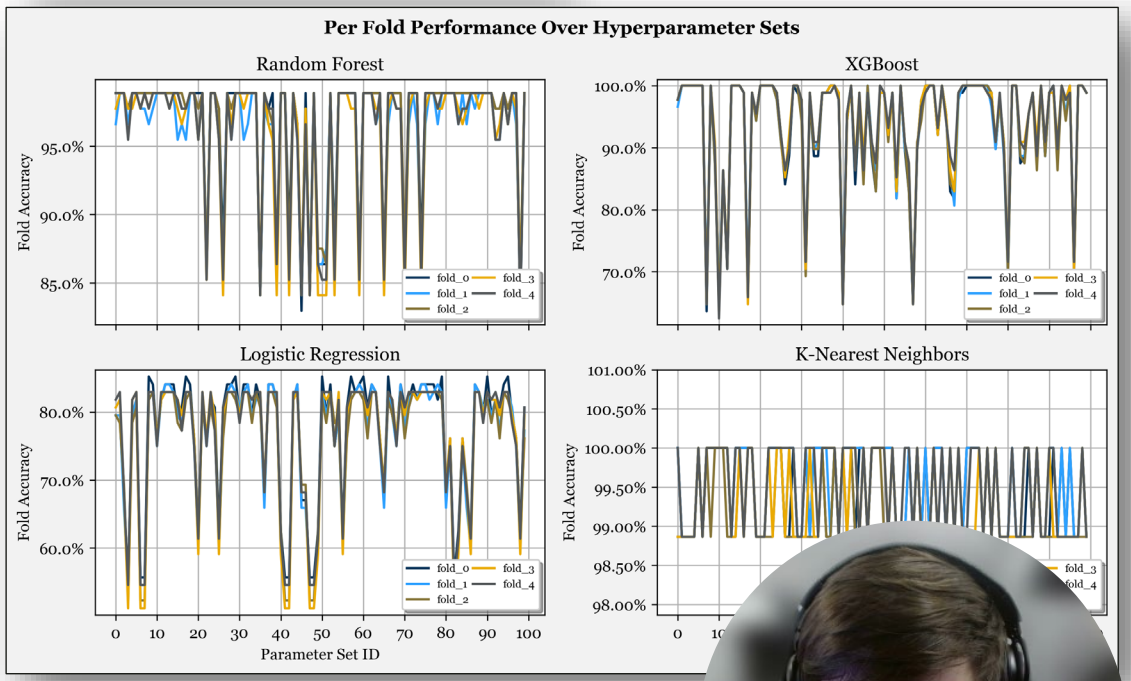
XGB

```
n_estimators = 200
learning_rate = 0.2
max_depth = 3
min_child_weight = 5
subsample = 0.8
colsample_bytree = 0.7
```

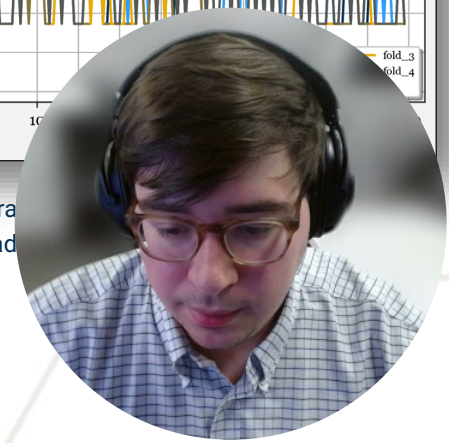
KNN

```
n_neighbors = 7
weights = distance
metric = minkowski
algorithm = brute
leaf_size = 10
```

First set of parameters used for each of the respective models used during hyperparameter tuning.



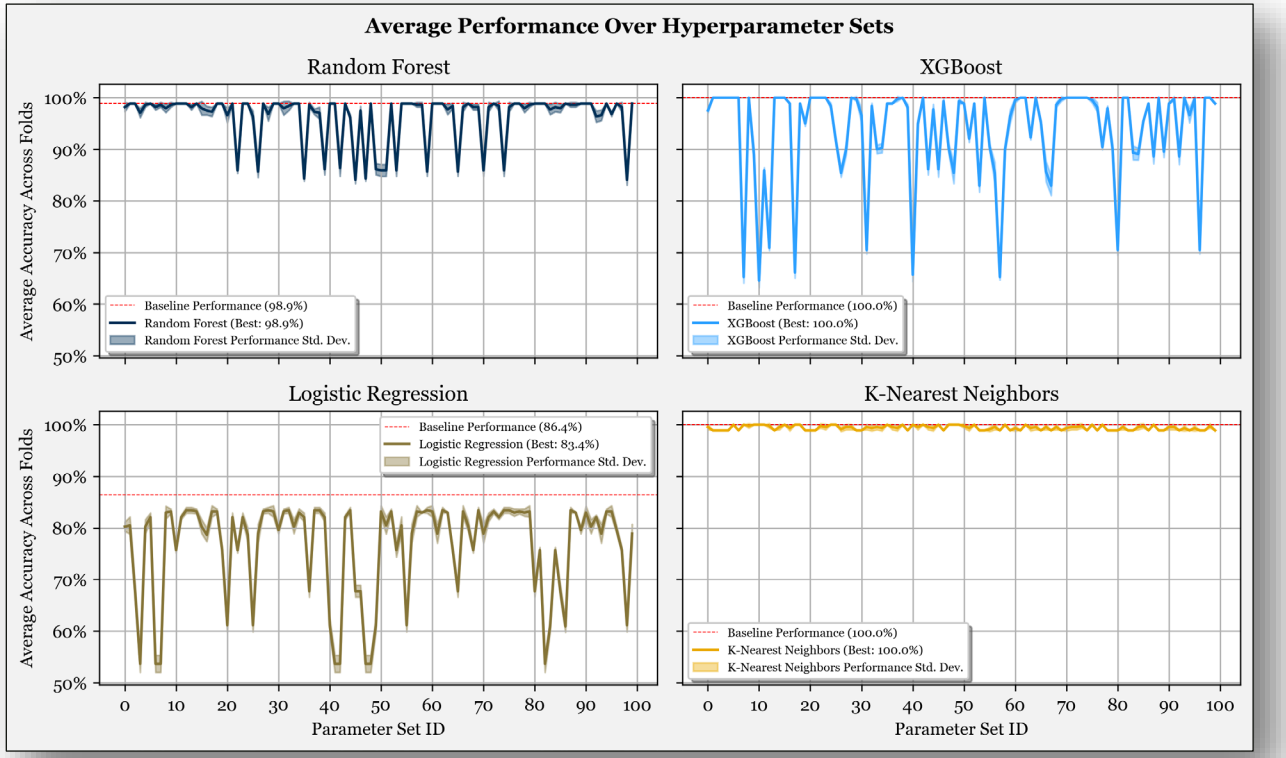
Model performance on each fold across 100 different hyperpara
each plot are intentionally not scaled the same to allow the read
relative to its range more easily.



Results & Conclusions

Results

- Having developed all the different tuned models, we can now explore the results and com-pare the performance of the models.
- The figure to the right more neatly visualizes the performance across the folds for the different models and hyperparameters and compares them against the baseline performance of each model.
- The table below summarizes some basic statistics from the visualization.



Model average performance across folds for each model for 100 different hyperparameter sets (solid line). The shaded area around each of the lines represents the standard deviation of the model performance across the folds.

Metric	Random Forest	XGBoost	Logistic Regression	K-Nearest Neighbors
Average	96.30%	93.48%	76.47%	99.39%
Std. Dev.	4.79%	9.58%	9.46%	0.46%
Minimum	84.09%	64.55%	53.64%	98.86%
25th Percentile	97.22%	90.00%	75.68%	98.86%
50th Percentile	98.64%	98.75%	80.45%	99.55%
75th Percentile	98.86%	100.00%	83.18%	99.77%
Maximum	98.86%	100.00%	83.41%	100.00%

Thank You!

Check out the details of my work on GitHub!

Repository: [Public-GT-Practicum-Alzheimers-and-FTD-Classification]