Sandia National Laboratories Practicum Project Alzheimer's and FTD Classification from EEG Data

Brian Keith, April 2024
bkeith9@gatech.edu

***Abstract*—**This paper will walk through my project done in conjunction with Sandia National Laboratories (SNL). The goal of this project was to try to solve the problem classifying patients as healthy or affected by Alzheimer's and Frontotemporal dementia (FTD) based on electroencephalogram (EEG) readings. To start, I will review the background of the project, my motivations for tackling this problem, and EEGs. This paper will then discuss the data set utilized, the feature extraction process, and the modeling performed to arrive at four different classification models with various performance characteristics. The overall results are very promising, and further work could lead to even greater improvements.

1 PROJECT BACKGROUND

The project sponsored by Sandia National Laboratories (SNL) revolved around using machine learning models to detect degradation in hardware components as they aged. This idea makes for an interesting project and has numerous real-world use cases as well as approaches (Shahraki et al., 2017) as physically testing components is time consuming and costly. Although SNL's introduction mostly focused on hardware components (e.g., machine bearings), the core concept of the project is to highlight how signals change over time and how we can use models to detect those changes.

Part of the reason this project was chosen was due to my background in signal anomaly detection . During my time with my previous employer, ANDRITZ, my team was responsible for remote monitoring of vibration sensors installed on Pulp and Paper manufacturing equipment. This experience highlighted the importance of models that detect issues before they occur. Doing so reduces workloads for the on-site maintenance staff and allows for maintenance to be scheduled during planned downtime translating to cost savings in a production environment.

While it would have been easy for me to take my existing domain knowledge and directly apply it to this project, challenged myself and took the concept proposed by SNL to apply it in a way that perhaps SNL had not directly considered when creating the project. As I researched and considered different possibilities for this project, I had an idea that felt like a revelation as someone without a medical background: The human brain is really 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. While the idea does stray from the exact motivations outlined by SNL, it retains the core concept of using signal data to detect and utilize patterns that exist in the electrical signals for the purpose of detecting a fault or failure.

2 MOTIVATION

Before I provide specific details of the project, it's important to discuss the motivation behind using models to diagnose neurodegenerative diseases as it differs in some ways from the direct monetary motivations laid out in SNL's project proposal.

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 (GBD 2019 Dementia Forecasting Collaborators, 2022). I as well as many others reading this may have a loved one or friend who they have seen affected by these diseases.

Although any medical diagnosis should include a variety of factors such as consultations with medical professionals and clinical diagnostics, one proposed method to assist medical professionals in diagnosing neurodegenerative diseases is electroencephalography (EEG), which measures brain electrical activity. The signals from the EEG readings can then be used to automatically find patterns that might indicate the presence of neurodegenerative diseases using machine learning models (Miltiadous et al., 2022). While there are currently no 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 (Rasmussen et al., 2019).

3 EEG OVERVIEW

An electroencephalogram (EEG) is a medical test that utilizes electrodes attached to a patient's scalp to measure electrical activity in the brain (Guy-Evans, 2023). 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 (Guy-Evans, 2023). These tests are used to detect a variety of neurological phenomena such as epilepsy, strokes, dementia, and other brain disfunctions (Guy-Evans, 2023).

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 (Miltiadous et al., 2021) as shown in Table 1.

Table 1 - Brain wave frequency bands.

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.

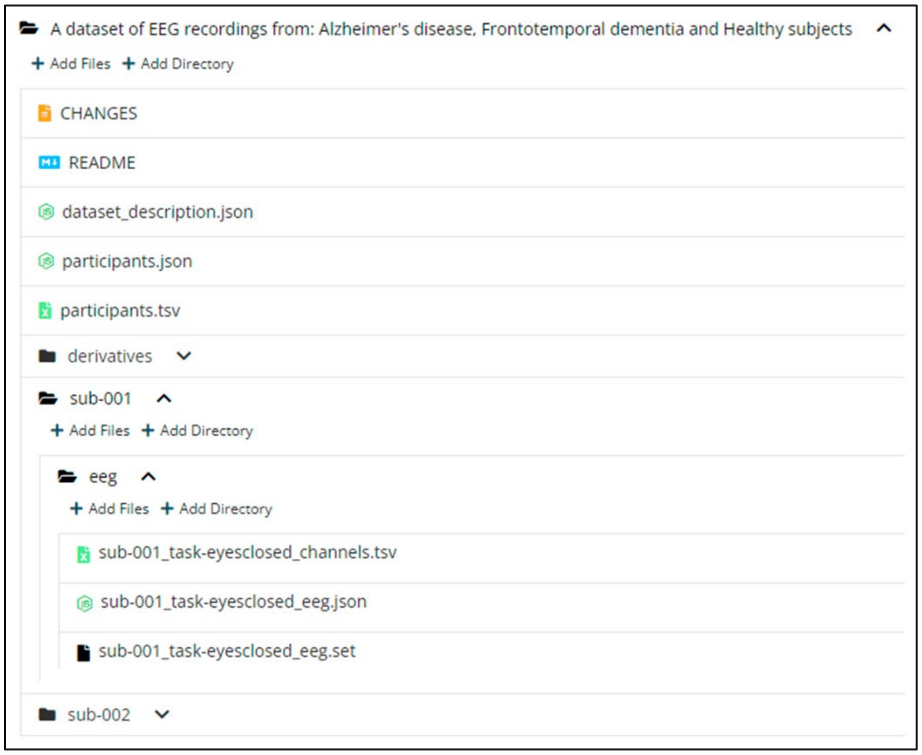
4 DATA SET

After researching potential datasets, I settled on a dataset from Greek researchers who published the dataset under a Creative Commons CC BY license[i] (Miltiadous et al., 2023). The dataset was very well documented and some of the very domain specific preprocessing of the dataset had already been performed (see Section 5.2).

The data set includes EEG data from 88 participants, 36 with Alzheimer's Disease, 23 with Frontotemporal dementia, and 29 healthy control subjects.

The data was provided as a ZIP file by the researchers with the folder structure shown in Figure 1.

[i] CC BY 4.0 Deed | Attribution 4.0 International | Creative Commons. (n.d.). https://creativecommons.org/licenses/by/4.0/


A dataset of EEG recordings from: Alzheimer's disease, Frontotemporal dementia and Healthy subjects

+ Add Files + Add Directory

```
CHANGES
README
dataset_description.json
participants.json
participants.tsv
derivatives
sub-001
    + Add Files + Add Directory
    eeg
        + Add Files + Add Directory
        sub-001_task-eyesclosed_channels.tsv
        sub-001_task-eyesclosed_eeg.json
        sub-001_task-eyesclosed_eeg.set
sub-002
```

Figure 1 - The structure of the dataset following the BIDS format (Miltiadous et al., 2023).

The structure, for the purposes of my analysis consisted of three different datasets:

1. Information describing the participants, which was extracted from the participants .tsv file. This dataset provides a link between the dataset for a given participant and the classification of the participant, i.e., Alzheimer's Disease, Frontotemporal dementia, or healthy control as well as supplementary information such as their gender.
2. The raw EEG data as .set files broken out by participant.
3. The EEG data with some domain specific pre-processing to remove artifacts in the EEG readings such as eye movement, also provided as .set files broken out by participant.

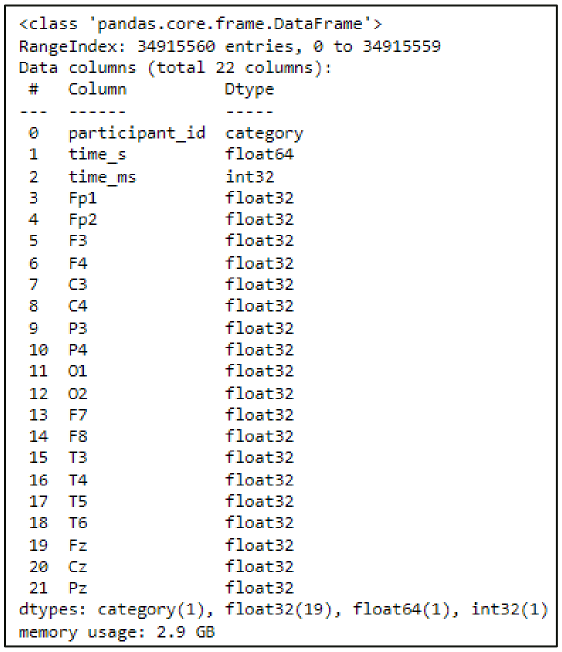
The two EEG datasets contained information from 19 different electrodes placed at various locations which were sampled at 500 Hz with a resolution of 10 μV/mm. Information related to the time in milliseconds for each sampled datapoint was also provided in the EEG data.

5 DATA HANDLING

5.1 Loading and Consolidation

Analyses were conducted using Python. Thus, I needed to load and transform the data out of the .set files and into a more Python friendly format such as pandas. To accomplish this, custom algorithms were written that scraped the folder directories to locate files of interest then used specific keys within the .set files to extract the data, recursively explode the nested data structure in the raw data and consolidate the data into one large pandas DataFrame object with approximately 35M rows.

The result of this consolidation can be seen in Figure 2. The columns numbered 3 – 21 represent the data 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), float64(1), int32(1)
memory usage: 2.9 GB
```

Figure 2 - Example of the .info() returned for the Raw EEG Data after data consolidation.

To avoid needing to redo this process every time I wanted to utilize the data, I performed some data type corrections and exported the data into a .pkl file for later use.

5.2 Choosing the Dataset

One nuance of the data provided by the researchers (Miltiadous et al., 2023) 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

A sample visualization of the EEG sensor data for sensors FP1, FP2, F3, F3, C3, and C4 can be seen in Figure 3. As can be seen in the data, 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.

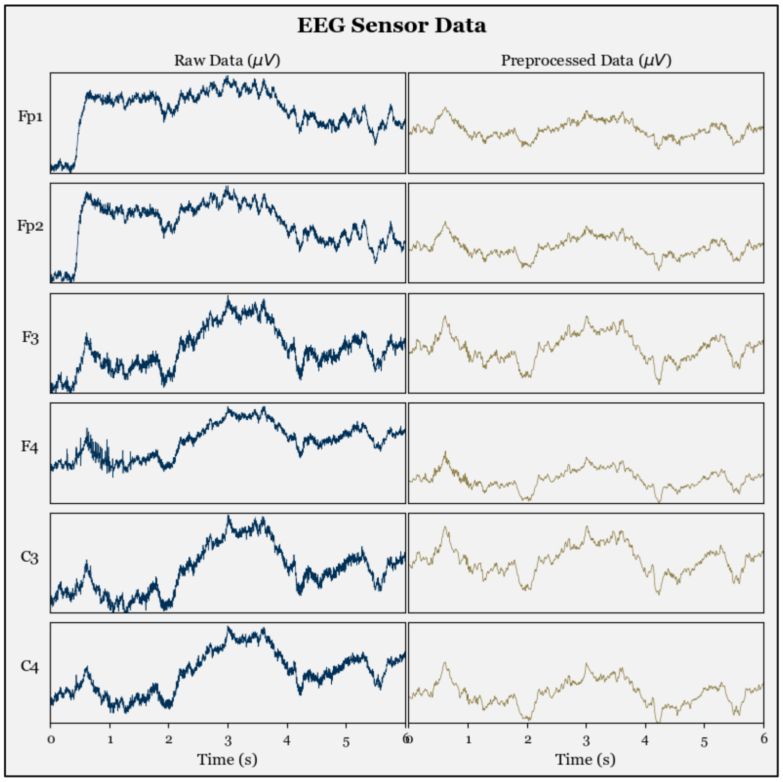


Figure 3 - Visualization of the Raw vs. Preprocessed data for Participant #1 across a few sample sensors for the first 6 seconds of the EEG recording. Note that the y-axis for all sensors is scaled to the same range for all the subplots.

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. For example, EEG artifacts can occur due to minor body movements, ECG interference, eye movements, sweating, as well as a variety of other factors (Abhang et al., 2016). Based on this, some of the techniques needed to recreate the preprocessed data would be outside of the scope of my capabilities as well as the time constraints of the project.

Therefore, the preprocessed data was used for my analysis. While this did simplify some of the data wrangling work required, in the spirit of the project from SNL, there was still significant signal processing work to extract the features from the EEG signals which I will be performing myself.

5.3 Feature Extraction

Before I could begin any modeling of the data, I needed to perform signal decomposition for each of the participants as well as for each of the 19 sensors in the preprocessed dataset.

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 (Xu et al., 2023)—of each band giving us a normalized metric between 0 and 1 for the contribution of a given PSD to the overall signal. Although normalization is not necessarily needed or beneficial to every model (Singh et al., 2020), when building classification models, using normalized features can lead to better model performance.

To calculate the PSD of each participant and sensor combination, I utilized the Welch method (Welch, 1967); a widely used model in research to estimate the PSD (Xiong et al., 2020). The Welch method works by dividing a given time series into successive blocks to form the periodogram, or squared-magnitude of a Discrete Fourier Transform (DFT) divided by the length of the DFT (Smith, 2011). To accomplish this calculation, the `welch()` function from SciPy's signal module[ii] was utilized. After deriving the PSDs of each brain wave frequency band for each patient and sensor combination with the `welch()` function, I then calculated the RBP which can be defined formulaically as:

$$\boldsymbol{RBP_a} = \frac{\sum_{f_a} \boldsymbol{P(f)}}{\sum_{f=f_1}^{f_2} \boldsymbol{P(f)}} \qquad \textbf{(1)}$$

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 (Xu et al., 2023).

While there are a variety of inputs to the `welch()` function, most of the arguments to this function were left as the defaults including the `window='hann'` argument because the Hanning window is widely used due to the good frequency resolution and reduced spectral leakage (Subramaniyam, 2018). With that being said, 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 (Subramaniyam, 2018). The sampling frequency (`fs` argument) was also set to a constant 500 because that is the sampling frequency of the data as noted in section 4.

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.

[ii] Scipy.Signal.welch — SciPy v1.12.0 Manual. (n.d.). Retrieved from https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html

An example visualization for the signal decomposition can be seen in Figure 4. As can be seen in the figure, the signal decomposition enabled me to break down the noisy signal into the PSD and RBP of each of brain wave from Table 1.

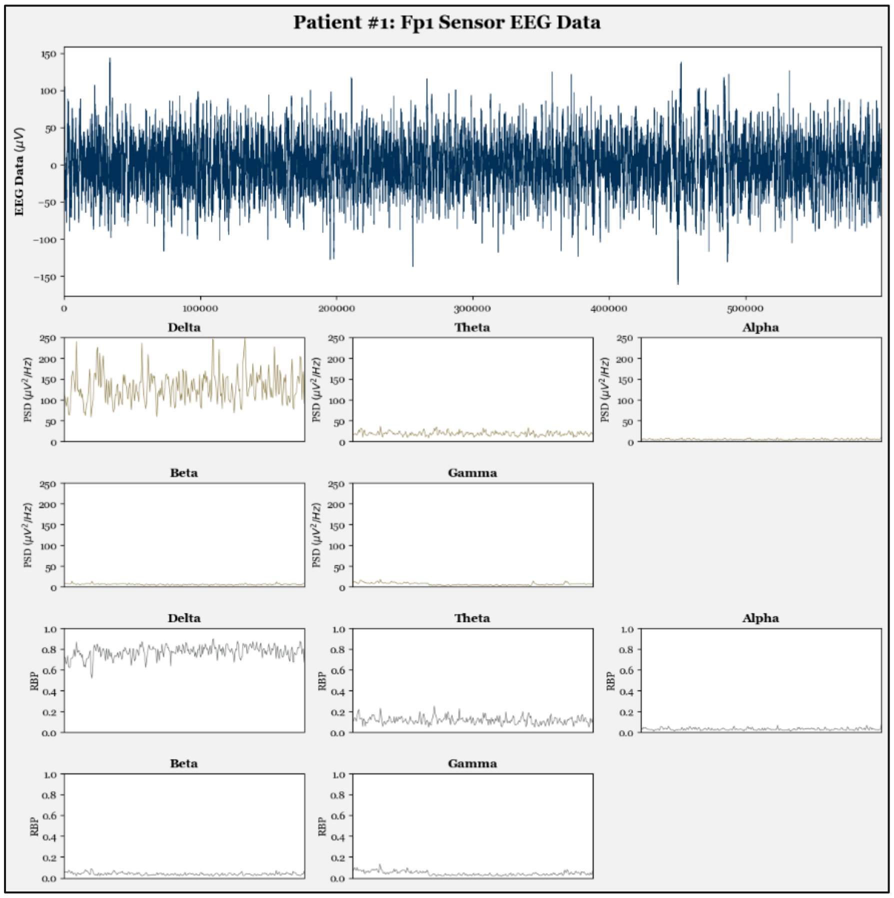


Figure 4 - A visualization of the decomposed EEG data using Welsh's method. The dataset used in this figure utilized the basis 4000 ms segments with 50% overlap.

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. An example of the final data structure that results for this process can be seen in Table 2.

Table 2 - Example of the head and tail of the final dataset structure using 4000 ms segments and 50% overlap. The seg_start and seg_end columns denote the start and end of the segments in milliseconds and the other columns to the right are the calculated PSD and RBP of each sensor/brain wave frequency combination.

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

6 MODELING

As anyone who has spent many years in the field would understand, choosing the right model and developing a reliable model can be quite a challenging problem. A common phrase in the analytics field is that modeling is more of an art than an exact science. Developing a robust and accurate model will many times involve tradeoffs between computational load, time complexity, and differing performance characteristics in different scenarios.

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, and K-Nearest Neighbors (KNN).

My choice of these models is primarily based on my experiences using them for other projects in my professional experience as well as in my classes at Georgia Tech. I would also like to note that originally, I was using sklearn's `GradientBoostingClassifier`[iii] and `SVC`[iv] algorithms. However, I found that the time required to fit the models was simply too long for the number of models I was implementing and was forced to utilize XGBoost and KNN instead.

The four models I ended up selecting should offer a board range of performance characteristics and offer a diverse set of results to analyze. The Random Forest and XGBoost models are similar in the sense that they are both tree-based classification algorithms. Random Forests are a bagging algorithm where the predicted classification is ensembled from the majority vote of the underlying "forest" of Decision Trees. This differs from XGBoost, which is a gradient boosting algorithm that attempts to minimize a loss function by learning from previous decision trees (Faghri, 2019). The Logistic Regression model in the context of classification is a probabilistic regression model that generates predictions as the probability that a given prediction belongs to a class (Rundel, 2013). Finally, KNN is a clustering model which makes predictions based on the most common classification of the neighbors within the feature space (Zemel, 2019). Both Logistic Regression and KNN are instance-based prediction algorithms which means that they generate their predictions from a single model whereas the other two models are ensemble

[iii] sklearn.ensemble.GradientBoostingClassifier. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn-ensemble-gradientboostingclassifier
[iv] sklearn.svm.SVC. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

models which draw their predictive power from many underlying models. By using this diverse set of models, the goal here is that we can determine which is best suited for the problem at hand. Based on my experience, the Random Forest and XGBoost models will likely have the best performance for this problem due to the large feature space which is being analyzed here.

Of note, when I determined these were the models I wanted to utilize, I did not view this as a finalized decision. This is illustrated by my pivoting away from some of my initial models as discussed previously. The core concept was to start with a simple set of models I was familiar with and revisit them if the performance of the models was not satisfactory. To implement each of these models in Python, I utilized sklearn's `RandomForestClassifier`[v], `LogisticRegression`[vi], and `KNeighborsClassifier`[vii] classes, respectively. For the XGBoost algorithm, xgboost's `XGBClassifier`[viii] class was utilized. For all random seeding/state setting within my code, I utilized a random seed based on my GTID.

Also of note, I encoded the labels provided in the dataset because the algorithms being used are all supervised learning algorithms. This was done by joining the dataset shown in Table 2 with the labels for each participant provided by the researchers (A = Alzheimer's disease, F = Frontotemporal dementia, and C = Healthy Control) and subsequently encoding those labels using sklearn's `LabelEncoder` function[ix].

6.1 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. As discussed previously, the analysis involved 15 different datasets with at least 3 different combinations of features to choose from (PSD, RBP, or both). Before even considering hyperparameter tuning or using a non-linear combination of features, the basic combinations would mean 180 total models to build. This is before considering hyperparameter tuning which would increase the number of models by orders of magnitude. In a perfect world, it would be great to be able to train and test every possible

[v] sklearn.ensemble.RandomForestClassifier. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[vi] sklearn.linear_model.LogisticRegression. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression
[vii] sklearn.neighbors.KNeighborsClassifier. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[viii] Python API Reference — xgboost 2.0.3 documentation. (n.d.). Retrieved from https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier
[ix] sklearn.preprocessing.LabelEncoder. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder

combination, however, that is simply not feasible. To simplify my scope, I decided to first test some base cases and approach the issue in a stepwise manner.

To begin, for all my models, I planned to use 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. Additionally, using all 190 features (both RBP and PSD features) did not seem logical to me due to the fact that the RBP is calculated based on the PSD, and therefore both sets should contain the same information and could introduce cross correlation which may increase the likelihood of issues with the resulting model (Dean et al., 2016). 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.

6.2 Basis Model Creation

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. This would allow me to further narrow the scope of the problem by eliminating 14 of the 15 datasets for each model, reducing the number of models I needed to tune from 60 (15 for each type of model) to just 4, thus allowing me to focus closely on a small set of models and robustly tuning and testing them to generate the best possible performance.

Therefore, to accomplish this, I built 15 basis models for each of the 4 different sklearn classes with all other parameters left as their respective defaults and an 80/20 train/test split[x] for the input dataset. After training the models on the training dataset, I then predicted the resulting classifications for each row in the testing dataset.

To measure the performance of these models, I chose to utilize accuracy as the metric. While there are other metrics that can be important when building classification models such as recall, precision, and f1 score, accuracy was chosen to evaluate the baseline models as accuracy is the most pivotal metric when measuring the model's ability to classify the participants. The goal for the basis models was to determine a baseline performance and to eliminate datasets, so the measure of accuracy is sufficient for that purpose. A more thorough model validation process was performed during the later stages of modeling during hyperparameter tuning.

[x] Data for the train/test split was done using sklearn's `train_test_split` function. sklearn.model_selection.train_test_split. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

Recall, 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 accuracy score was then calculated based on the number of correct and incorrect participant classifications. The results for the four sets of models can be seen in Figure 5.

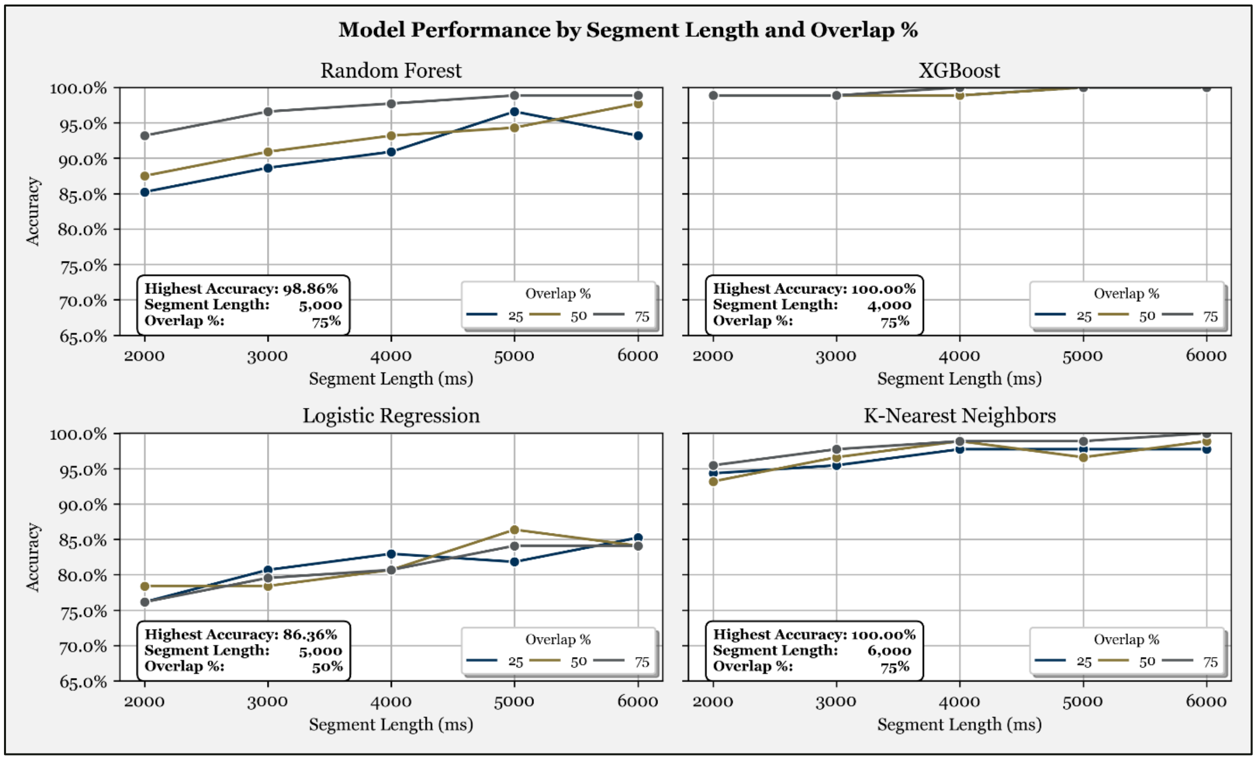


Figure 5 - Accuracy Scores for all 60 basis models created.

At this point, we are not necessarily interested in comparing the models to each other especially since some models (e.g., Logistic Regression) could require more tuning to experience high levels of accuracy on high dimensional datasets. I did findthat the performance of the KNN model to be quite surprising given the breadth of the feature space here; however, this could just be an indication of overfitting—an issue I will investigate further in the coming sections.

We can see that for all the models, the performance generally tended to increase with larger segment lengths overall. While I cannot be certain what causes this, my suspicion is that with larger segment sizes, there is more signal data in each datapoint being fed into the model, leading to better predictions.

The effect of overlap seems more varied across the models and is inconsistent. My initial thought on this was the amount of overlap in the segments could lead to some noise since the RBP features are derived from underlying data that contains the same information. However, we can see that all the best models either came from using 75% or 50% overlap which would indicate a net positive effect of higher overlap on the predictions. After some consideration, the most plausible explanation for this is that having a higher overlap percentage means that the dataset will be overall larger because additional overlap means that the total time is subdivided into more of the equal sized segments. This theory in conjunction with the models generally tending toward better predictive power at higher segment lengths illustrates a balancing act here between the total number of datapoints (which is proportional to the overlap percentage an inversely proportional to the segment length).

Regardless of the exact reasons, based on the basis models, the following datasets were used for each of the respective models:

- Random Forest → 5000 ms segments with 75% overlap
- XGBoost → 4000 ms segments with 75% overlap
- Logistic Regression → 5000 ms segments with 50% overlap
- KNN → 6000 ms segments with 75% overlap.

Figure 6 through Figure 9 show the confusion matrix for each of the best performing basis models.

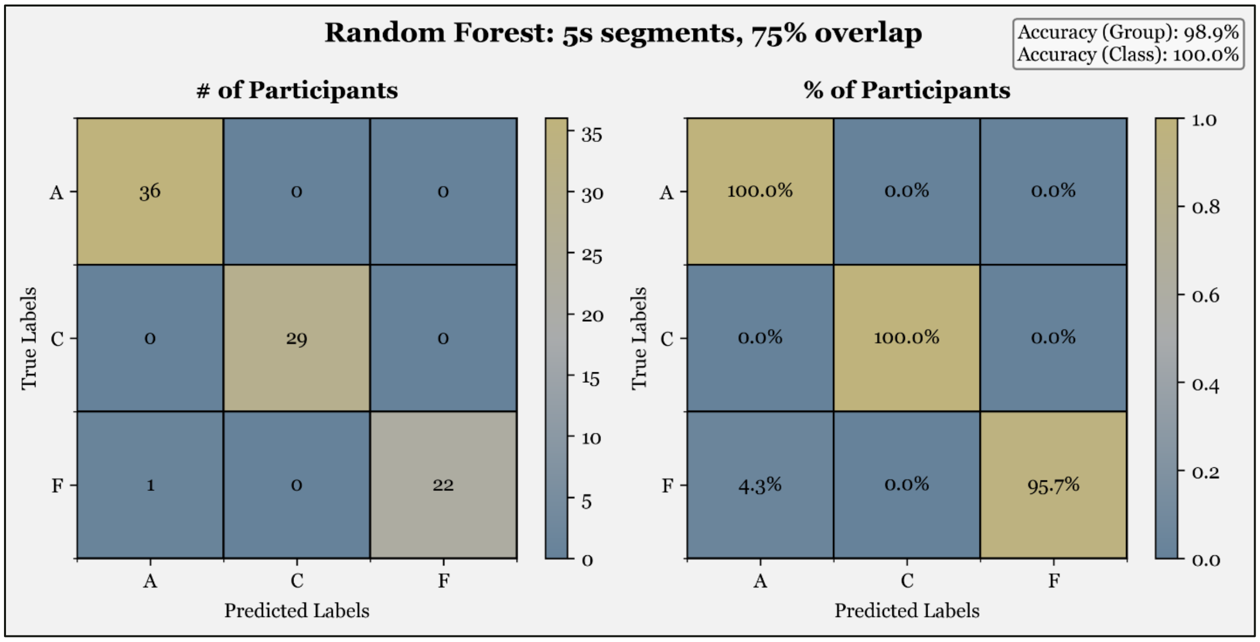


Figure 6 - Best performing Random Forest model (5000 ms segments with 75% overlap) confusion matrices. The overall accuracy across predictions of Alzheimer's, FTD, and control is shown in the top right. Note that the Accuracy (Class) Label

indicates the overall accuracy when predicting only between healthy control participants and those with either of the two neurodegenerative diseases.

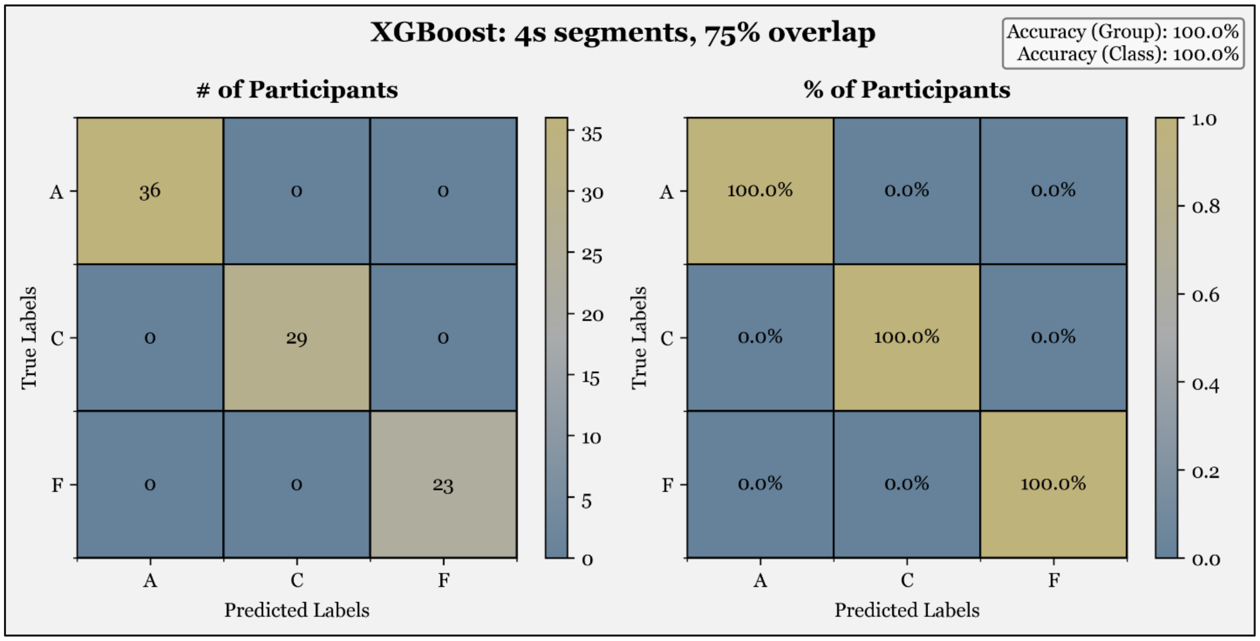


Figure 7 - Best performing XGBoost model (4000 ms segments with 75% overlap) confusion matrices. The overall accuracy across predictions of Alzheimer's, FTD, and control is shown in the top right. Note that the Accuracy (Class) Label indicates the overall accuracy when predicting only between healthy control participants and those with either of the two neurodegenerative diseases.

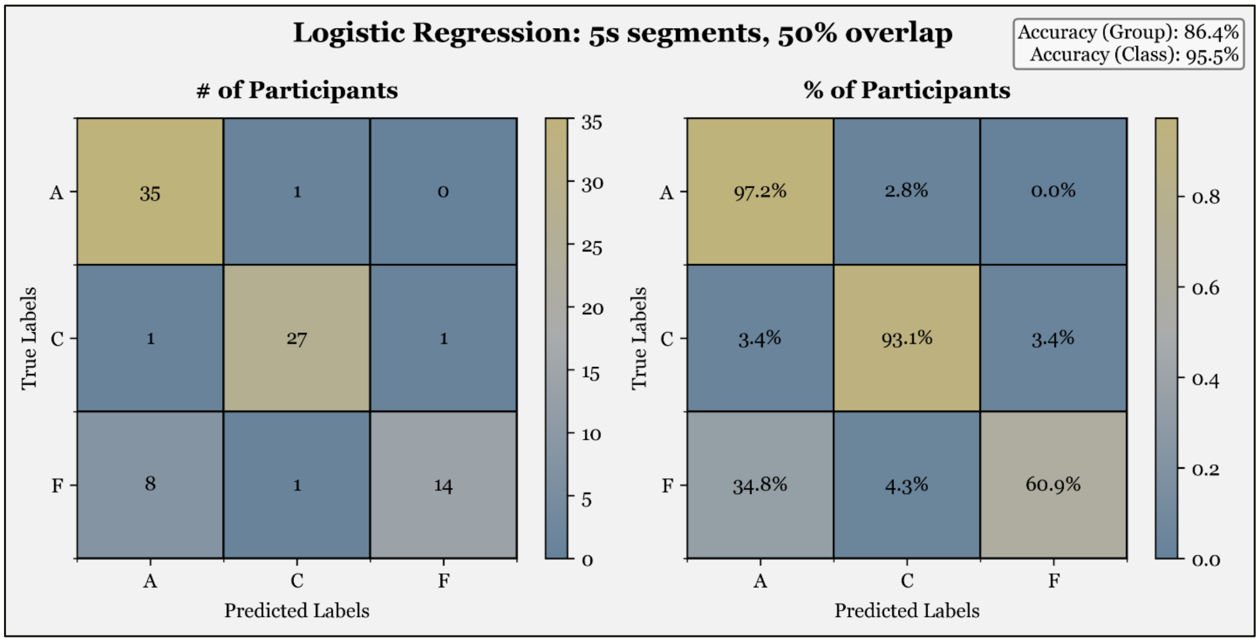


Figure 8 - Best performing Logistic Regression model (5000 ms segments with 50% overlap) confusion matrices. The overall accuracy across predictions of Alzheimer's, FTD, and control is shown in the top right. Note that the Accuracy (Class) Label indicates the overall accuracy when predicting only between healthy control participants and those with either of the two neurodegenerative diseases.

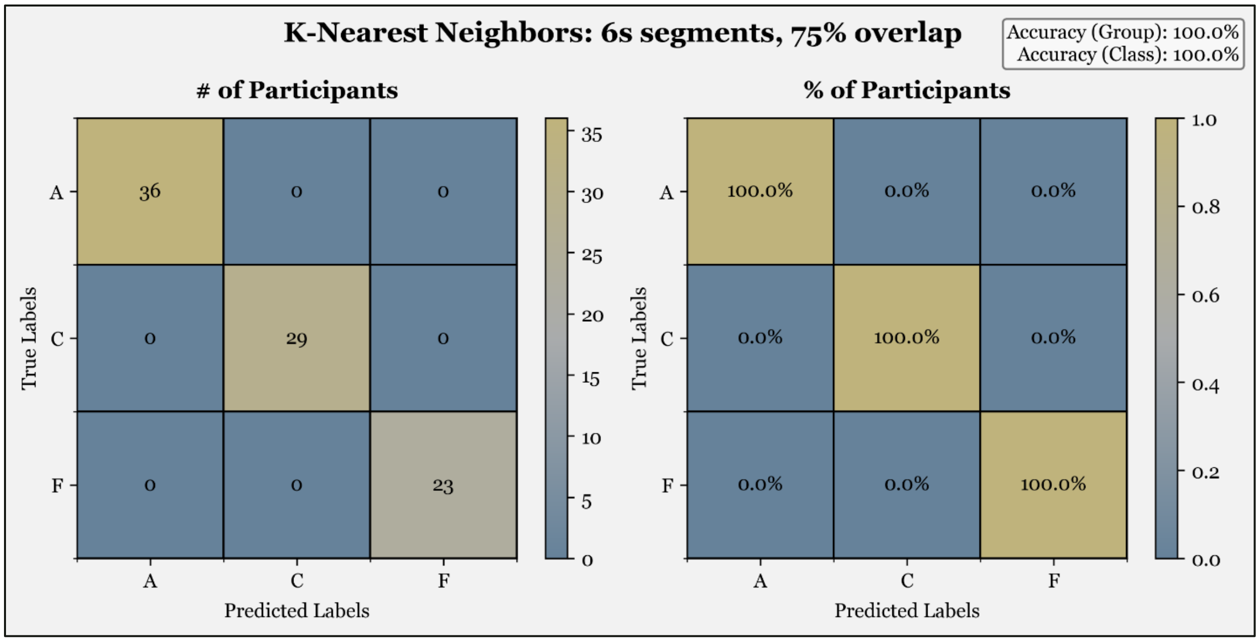


Figure 9 - Best performing KNN model (6000 ms segments with 75% overlap) confusion matrices. The overall accuracy across predictions of Alzheimer's, FTD, and control is shown in the top right. Note that the Accuracy (Class) Label indicates the overall accuracy when predicting only between healthy control participants and those with either of the two neurodegenerative diseases.

Looking at these figures offers some interesting insight, especially for the Logistic Regression model which appeared to have lower overall performance in Figure 5. What we can see from the confusion matrices is that the lower classification accuracy is generally driven by FTP patients being classified as having Alzheimer's disease. While it would obviously be better if this error source did not exist, it is at least promising as this is somewhat expected for the early models given the similarities and more nuanced differences between the two diseases (Miltiadous et al., 2021).

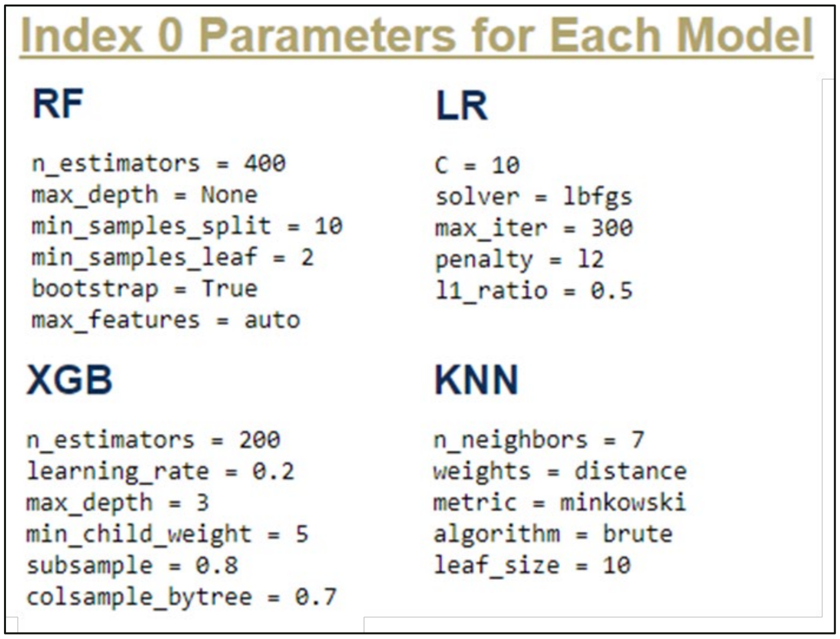
6.3 Hyperparameter Tuning

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. Like with the basis models discussed previously, the key metric to determine the best performance was the classification accuracy. However, to determine the accuracy instead of using a single 80/20 train/test split dataset, K-Fold cross validation[xi] with 5 folds was also employed to help ensure the reliability of the results from the hyperparameter tuning. In K-Fold cross validation, the model is trained on $k-1$ of the

[xi] K-Fold cross validation was performed using sklearn's `Kfold` class with shuffling set to True. Sklearn.model_selection.KFold. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

folds then evaluated based on the remaining fold. This process is then repeated k times and the performance is evaluated as the average performance of the folds (Mackey, 2015).

To perform the hyperparameter tuning, due to the scoring method employed where the most common category for each participant was used as discussed in the previous section, I could not use grid search classes such as sklearn's `GridSearchCV`[xii] class. Instead, I utilized the itertools library[xiii] to generate all possible combinations of the input parameters as a list of items[xiv]. Due to time and processing constraints, not all the possible model parameters could be fitted 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. Each set of parameters used for the respective models were saved and assigned an index for the sake of traceability. An example of the first set of parameters used during the hyperparameter tuning can be seen in Figure 10.



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
```

Figure 10 – First set of parameters used for each of the respective models used during hyperparameter tuning.

This resulted in a total of 2,000 different models (4 different models with 100 sets of hyperparameters fit across 5 folds). The results for the performance of the models are based on the accuracy of the models using the same methodology outlined earlier. These 2,000 models are visualized in Figure 11.

[xii] sklearn.model_selection.GridSearchCV. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV
[xiii] itertools — Functions creating iterators for efficient looping. (n.d.). Retrieved from https://docs.python.org/3/library/itertools.html
[xiv] The exact hyperparameter grid utilized for each model will not be discussed in this paper for the sake of brevity. However, this information can be made available upon request.

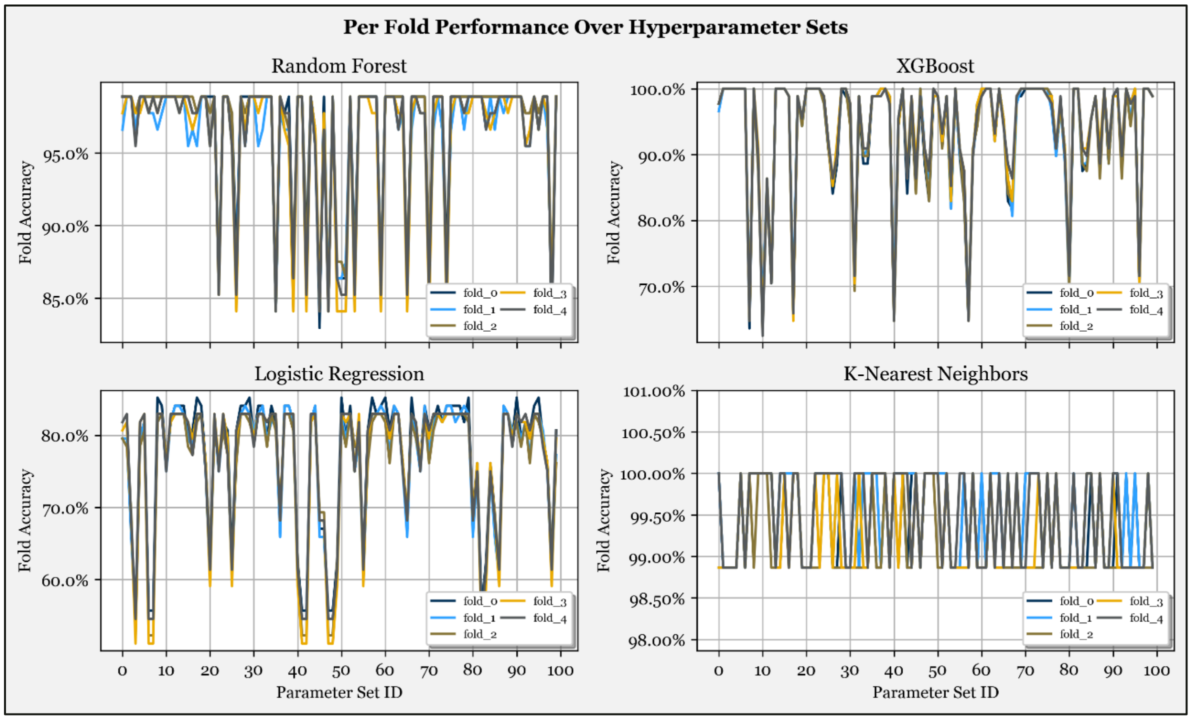


Figure 11 - Model performance on each fold across 100 different hyperparameter sets. Note that the y-axes for each plot are intentionally not scaled the same to allow the reader to see the variation for each model relative to its range more easily.

In Figure 11, we can see the accuracy of each model on the individual folds for each model's set of 100 parameters. What we are interested in seeing for this figure is significant variance in the any the performance across folds which indicates an inability for the model to generalize well to different datasets. Overall, based on this figure, we can see that while there is some variance across different folds, the inter-fold variance in accuracy is relatively minimal, which indicates that the models are robust.

Having developed all the different tuned models, I will now discuss the results, discuss the findings from this analysis, and compare the performance of the models.

7 FINAL RESULTS

Having developed all the different tuned models, we can now explore the results and compare the performance of the models.

To begin, I generated Figure 12 which is a more tidy version of Figure 11 and shows the average and standard deviation of the accuracy across the five folds and compares the performance against the baseline performance achieved before hyperparameter tuning. To add some hard numbers to the visualization, Table 3 summarizes the average performance (solid line for each model from Figure 12) across the 100 parameter sets for each model.

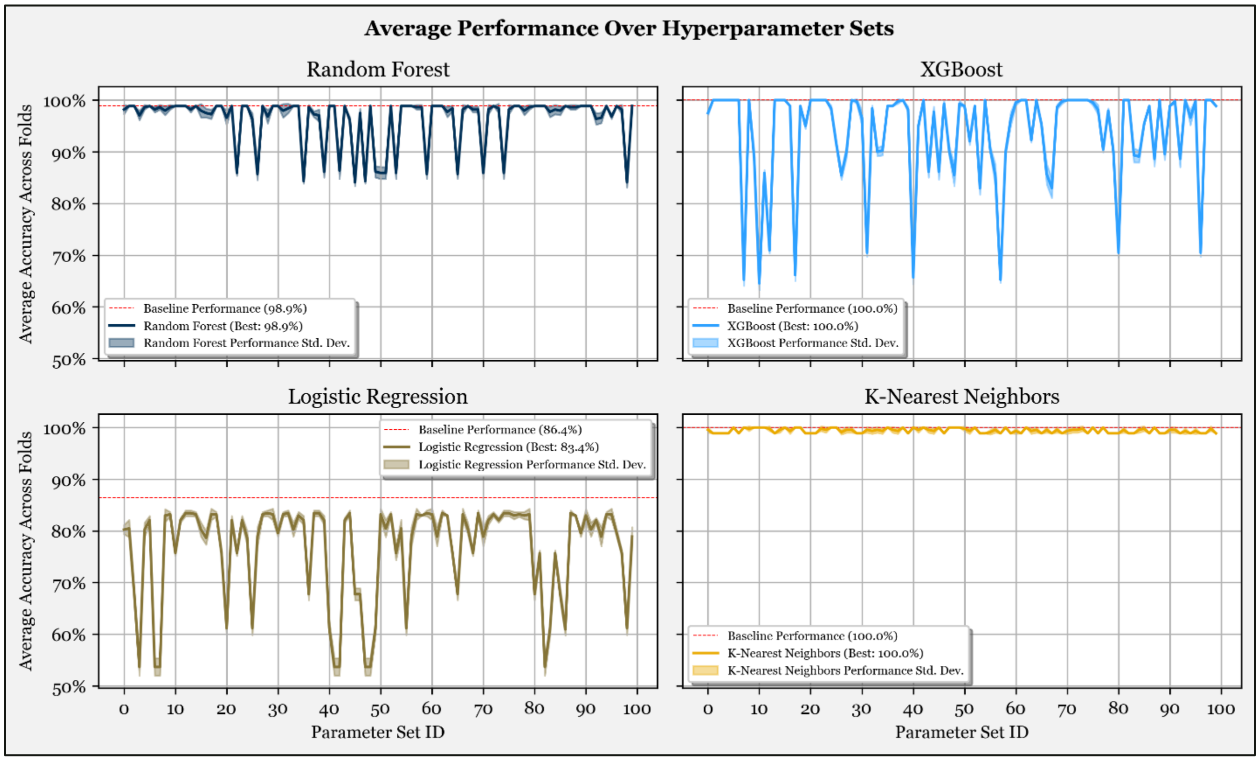


Figure 12 - Model average performance across folds for each model across 100 different hyperparameter sets (solid line). The shaded area around each of the lines represents the standard deviation of the model performance across the folds. The dotted red line on each subplot represents the baseline performance of the model with no tuning on the respective datasets. All datasets are scaled to the same y-axis range to allow for comparison of performance.

Table 3 – Average fold accuracy statistics for models across all sets of hyperparameters.

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%

Figure 12, allows us to compare the models and their performance more easily across their respective tuning parameters while Table 3 summarizes some basic statistics from the visualization. There are a few things that emerged from this which I believe are quite interesting as well as some unexpected behavior.

First, the primary behavior that surprised me is the consistency of performance for the KNN models. Not only was the classification accuracy very high for all the models, but it had extremely low variance with a minimum accuracy of 98.86%. Earlier in this paper, I postulated that the performance shown in Figure 5 could be due to overfitting. Based on these results, however, I do not think that would be fair to say as it maintained the high performance levels across the tested parameters as well as during the cross validation. It is clear that while I had some doubts regarding the usefulness of KNN, it is very well suited for the classification problem.

Second, both tree-based models, Random Forest and XGBoost had significant variance based on the parameter set utilized with the accuracy varying from 84.09% - 98.86% and 64.55% - 100%, respectively. This is consistent with some of my previous experience using these models in the past as their ability to generalize can be significantly impacted by the hyperparameters. With that being said, within a given set of hyperparameters, the variation in the scores across different folds is generally minimal. What this means in the end is that those sets of parameters with low overall performance are indicative of tunings which are not appropriate and should be discarded.

Finally, it is clear based on the analysis that Logistic regression is not a good model to fit the problem here with a maximum average accuracy across folds of only 83.41% and it failed to achieve the baseline performance found using the default parameters and cross validation. After looking in more detail at the classifications, it appears that the Logistic Regression model is not able to distinguish between FTD and Alzheimer's effectively with the data utilized. While the acceptable level of performance for a model can often be a relative measure, it is clear the other models are better suited for the problem.

Overall, based on the findings from this analysis, KNN, Random Forest, and XGBoost are all good models and had very promising results. In terms of which is best, I have mixed thoughts on the subject. Based on the analysis, the three models appear to have different tradeoffs which could make them more appropriate in different scenarios. For example, we saw in Figure 5 that the XGBoost appeared to have consistent performance regardless of the segment length and overlap, which could make it more flexible to RBP signals generated with different metrics. On the other hand, KNN seemed to be more sensitive to the segment length and overlap but less sensitive to the hyperparameters used. Lastly, while the Random Forest appeared sensitive to both the hyperparameters and segment length/overlap, it did maintain generally good performance, and I would classify it somewhere in middle of the two tradeoffs experienced with the others.

If I could only choose one of the three models to deploy and continue developing with additional datasets, KNN would likely be the model of choice due to its consistent performance against different hyperparameter sets which indicates more generalizability of the model and those parameters are harder to adjust than the input parameters.

8 CONCLUSIONS

To conclude, during this project, I was able to work with Sandia National Laboratories to develop a topic with real world applications that leverages data analytics and machine learning models to classify patients into three different categories, Alzheimer's Disease, Frontotemporal Dementia and Healthy. This analysis required data processing, feature extraction, feature selection, and multiple rounds of modeling to arrive at the cross-validated results for the sets of hyperparameters tested..

At the end, I was able to build out models based on four different algorithms, Random Forest, XGBoost, Logistic Regression, and KNN. Of these models, it was clear that the Logistic regression model was the least suited to the problem with a maximum average fold accuracy of 83.41% due to its inability to distinguish well between FTD and Alzheimer's Disease. The other three models offered very promising results with maximum average fold accuracies of 98.86%, 100%, and 100.00% for Random Forest, XGBoost, and KNN, respectively.

While I am somewhat skeptical that accuracies this high could be achieved on new datasets or in a clinical environment, it does show significant promise for the potential for machine learning models to be used in assisting physicians in the future to help diagnose these neurodegenerative diseases more efficiently. I would of course welcome any others to continue work on this topic, try to replicate these results, and/or take their own approaches to this problem.

Thank you for your attention and I hope you enjoyed reading this paper as much as I enjoyed working on this project.

9 REFERENCES

Abhang et al. (2016). Technological Basics of EEG Recording and Operation of Apparatus. In Abhang et al., *Introduction to EEG- and Speech-Based Emotion Recognition* (pp. 19-50). Academic Press. doi:10.1016/B978-0-12-804490-2.00002-6

Dean et al. (2016). Dangers and uses of cross-correlation in analyzing time series in perception, performance, movement, and neuroscience: The importance of constructing transfer function autoregressive models. *Behav Res, 48*, 783-802. doi:10.3758/s13428-015-0611-2

Faghri, F. (2019). *Random Forests & XGBoost.* Retrieved from cs.toronto.edu: https://www.cs.toronto.edu/~rgrosse/courses/csc2515_2019/tutorials/tut4/tut4_boost.pdf

GBD 2019 Dementia Forecasting Collaborators. (2022). Estimation of the global prevalence of dementia in 2019 and forecasted prevalence in 2050: an analysis for the Global Burden of Disease Study 2019. *The Lancet, 7*(2), 105-125. doi:10.1016/s2468-2667(21)00249-8

Guy-Evans, O. (2023, September 19). *EEG Test (Electroencephalogram): Purpose, Procedure, And Risks*. Retrieved from SimplyPsychology: https://www.simplypsychology.org/what-is-an-eeg.html

Mackey, L. (2015, October). *Lecture 11: Cross validation.* Retrieved from web.stanford.edu: https://web.stanford.edu/~lmackey/stats202/content/lec11-condensed.pdf

Miltiadous et al. (2021). Alzheimer's Disease and Frontotemporal Dementia: A Robust Classification Method of EEG Signals and a Comparison of Validation Methods. *Diagnostics, 11*(8), 1437. doi:10.3390/diagnostics11081437

Miltiadous et al. (2022). Machine Learning Algorithms for Epilepsy Detection Based on Published EEG Databases: A Systematic Review. *IEEE Access*, 564-594. doi:10.1109/ACCESS.2022.3232563

Miltiadous et al. (2023). A Dataset of Scalp EEG Recordings of Alzheimer's Disease, Frontotemporal Dementia and Healthy Subjects from Routine EEG. *Data, 8*(6), 95. doi:10.3390/data8060095

Rasmussen et al. (2019). Alzheimer's disease – Why we need early diagnosis. *Degenerative neurological and neuromuscular disease*, 123-130. doi:10.2147/dnnd.s228939

Rundel, C. (2013, April). *Lecture 20- Logistic Regression.* Retrieved from stat.duke.edu: https://www2.stat.duke.edu/courses/Spring13/sta102.001/Lec/Lec20.pdf

Shahraki et al. (2017). A Review on Degradation Modelling and Its Engineering Applications. *nternational Journal of Performance Engineering, 13*(3), 299-314. doi:10.23940/ijpe.17.03.p6.299314

Singh et al. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing, 97*(B). doi:10.1016/j.asoc.2019.105524

Smith, J. (2011). *Welch's Method*. Retrieved from Spectral Audio Signal Processing: https://ccrma.stanford.edu/~jos/sasp/Welch_s_Method.html

Subramaniyam, N. (2018). *Factors that Impact Power Spectral Density Estimation*. Retrieved from Sapien Labs: https://sapienlabs.org/lab-talk/factors-that-impact-power-spectrum-density-estimation/

Welch, P. (1967). The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics, 15*(2), 70-73. doi:10.1109/TAU.1967.1161901

Xiong et al. (2020). A Parallel Algorithm Framework for Feature Extraction of EEG Signals on MPI. *Computational and Mathematical Methods, 2020*. doi:10.1155/2020/9812019

Xu et al. (2023). Mental Fatigue Degree Recognition Based on Relative Band Power and Fuzzy Entropy of EEG. *International Journal of Environmental Research and Public Health, 20*(2), 1447. doi:10.3390/ijerph20021447

Zemel, e. (2019). *CSC 411: Lecture 05: Nearest Neighbors.* Retrieved from cs.toronto.edu: https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/05_nn.pdf