DNA Methylation with Machine Learning: Prediction of Alzheimer's Disease and Novel Gene Therapeutics

Team 34, La Cueva High School

April 6, 2022

New Mexico Supercomputing Challenge Final Report

Team Members:

- Aditya Koushik
- Abitpal Gyawali

Teacher:

- Yolanda Lozano

Table of Contents

Executive Summary

Alzheimer's Disease (AD) is the most common form of dementia (loss of memory and other cognitive abilities) as it accounts for 60-80% of dementia cases. Over 50 million people worldwide live with Alzheimer's Disease and currently there is no cure/treatment, except a recent controversial monoclonal antibody therapy (Aduhelm). Therefore, Alzheimer's Disease serves as an important condition to target with various bioinformatics and machine learning methods for both prevention and therapy. Currently, Alzheimer's Disease can be diagnosed by MRI-PET and biomarkers in Cerebrospinal fluid (CSF). However, these methods can be expensive and invasive. Therefore, this project aims to offer an alternative of diagnosing Alzheimer's Disease using blood-based genomic DNA methylation levels and machine learning (Classification and Hierarchical Clustering). We downloaded a publicly available dataset from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE144858 which contained DNA methylation samples profiled for 300 patients (111 with Mild Cognitive Impair or MCI, 96 with No Alzheimer's Disease - Control, and 93 with Alzheimer's Disease). Using Mutual Information (Information gain), the top 10 Differentially Methylated Regions (DMRs) for only Alzheimer's Disease and Control patients were identified. Next, we trained seven machine learning classification models (Random Forest, K-Nearest Neighbor, Decision Tree, Support Vector Machine, Naive Bayes, Gradient Boosting, and Logistic Regression) on the data of the top 10 DMRs to evaluate the model accuracy in predicting the occurrence of Alzheimer's Disease. We then identified another set of DMRs which includes Mild Cognitive Impairment (MCI - an early stage of Alzheimer's Disease) data using ANOVA F-Test. We built and tested a hierarchical clustering model on the new set of DMRs to see if it could predict the occurrence of Alzheimer's Disease as well as MCI. With classification (only predicted Alzheimer's Disease and Control), the highest scoring model was Gradient Boosting with 90% model accuracy. In addition, our hierarchical clustering model could predict Alzheimer's Disease, Control, and MCI with 86% accuracy. Finally, we used the Cytoscape software and MCODE to predict the interactions of genes methylated by the top 2000 DMRs and to identify genes (RPL23, RPS13, VARS, and MINK1), that can be used for prediction and therapeutic targeting for Alzheimer's Disease.

Introduction

Our Project - Project Idea

Purpose - Using machine learning models to predict the severity of the patient's Alzheimer's Disease with the DNA Methylation levels, and to predict genes that can be used for therapeutic targets for Alzheimer's Disease.

Background - Epigenetics, DNA Methylation, Alzheimer's Disease

Epigenetics

What is epigenetics?

Epigenetics is a study of changes in the DNA's behavior that are not caused by alterations of the DNA sequence. Rather than causing changes in the DNA sequence, epigenetic processes alter the way the DNA is read. The simplest example of epigenetics can be observed in the difference in the functionalities of muscle cells in comparison to the functionalities of nerve cells. Muscle cells and nerve cells both contain the same replica of the DNA, however their functionalities differ by a large margin. This is because of epigenetics. Different epigenetic changes occur between the muscle and nerve tissues that cause all the genes that create proteins for nerve cells to turn off in the muscle cells and vice versa. In other words, epigenetic processes allow cells to have different functionalities by controlling which genes get expressed.

What causes epigenetic processes to occur?

Epigenetic processes are caused and changed due to environmental factors, behavioral factors, and aging factors. Here's an image from https://www.sun.edu.ng/knowledge-update/epigenetic-processes-and-human-health

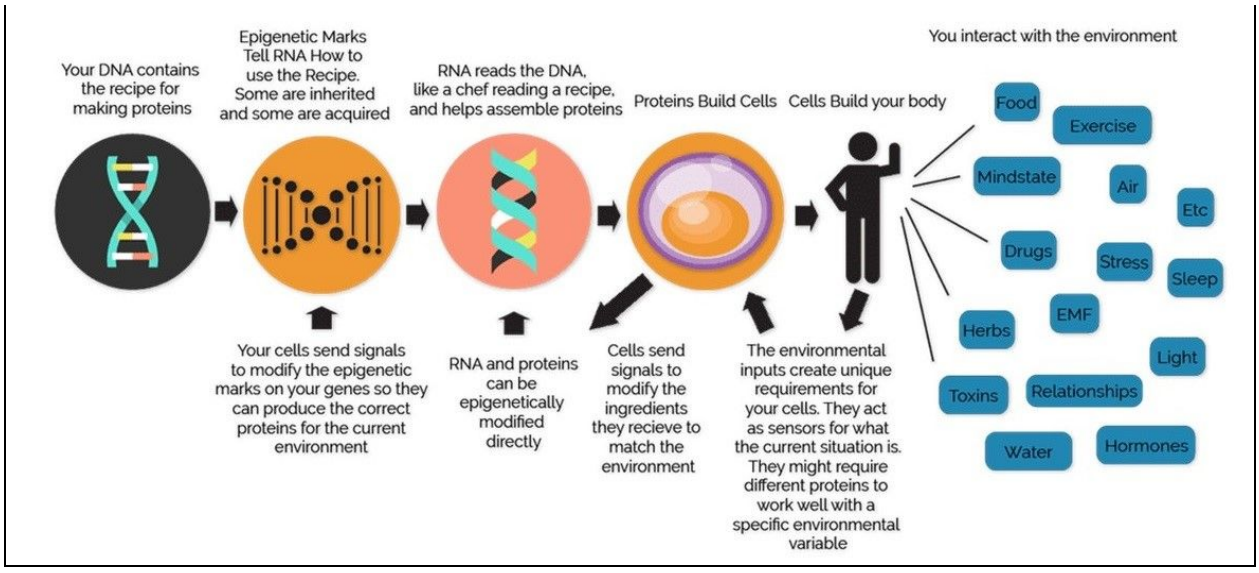


Figure 1

Significance of epigenetic processes

As said earlier, epigenetic processes control which genes are turned ‘on’ and ‘off’. So what happens if an epigenetic process accidentally turns on a gene that’s supposed to be off? This can sometimes be harmless, however, it can also be crucial to the body's ability to live. For example, irregular epigenetic processes are known to cause various diseases such as Alzheimer's Disease, and various forms of cancer.

DNA Methylation

What is DNA Methylation?

DNA Methylation is one of the epigenetic processes that regulates which genes are read/transcribed to make proteins. This is done through addition of methyl groups to the gene to repress its transcription, hence turning ‘off’ the gene. If genes are methylated without control, it can cause irregularities in expression of genes, and can cause different diseases (including Alzherimer’s Disease).

Differentially Methylated Regions (DMRs)

Differentially methylated regions are regions in DNA that have a different methylation level in a healthy patient compared to an Alzheimer's Disease. Methylated regions are methylated locations on genes denoted by cg######## (Eg. cg01813033). DMRs are methylated regions that are only significant to Alzheimer's Disease since they are differentially methylated. For example, if a certain region in the gene has a high methylation in people with Alzheimer's Disease compared to healthy people, it is a DMR. This also applies to low methylation.

Purpose

The purpose of this project is to predict the occurrence of Alzheimer's Disease using Classification and Hierarchical clustering models, and to predict gene interactions of 2000 DMRs for novel gene therapeutic targets for Alzheimer's Disease.

Materials

- Jupyter notebook running Python 3.8
- https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE144858 for data on ~411,000 DNA methylation values across the **whole human genome** in 300 people (publicly available from GEO – gene expression omnibus)
- Cytoscape to predict gene interactions
- MCODE to predict gene interactions clusters
- BioRender to make models of the concepts
- Python Modules used include SciPy, Pandas, NumPy, Matplotlib, Sklearn and Seaborn.

Methods

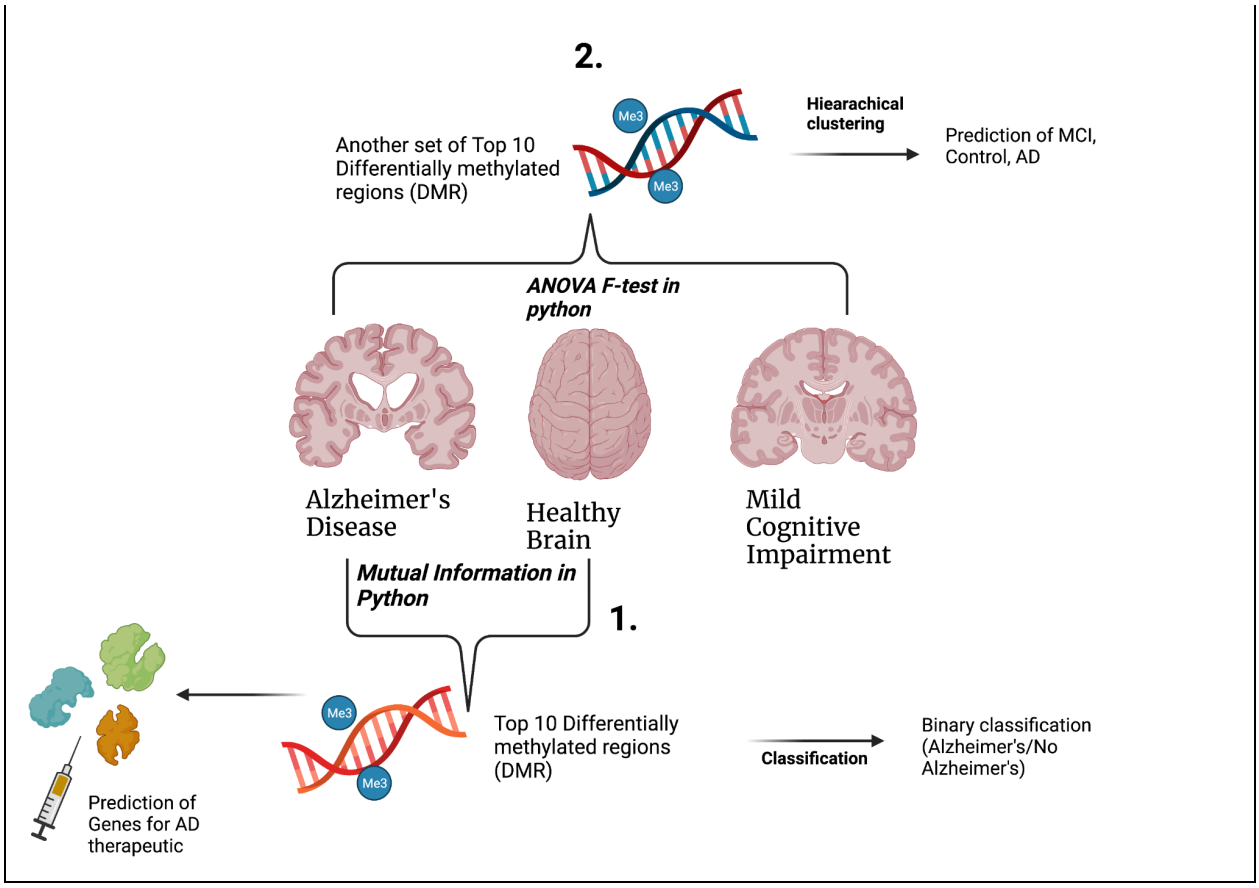


Figure 2

Github: https://github.com/pauldbd/Supercomputing-challenge-2021-2022

(1851 Lines of Code in total)

1. Gather materials
    a. Data was publicly available from GEO - https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE144858
2. Using Mutual Information, find the top 10 Differentially Methylated Regions (DMRs). This is only for Alzheimer's and Control (MCI is excluded for classification)
3. Load the data of the top 10 DMRs into a new Jupyter Notebook. Shuffle the data to prevent bias.
4. Split the data into training and testing with 70% - 30% split (70 for training, 30 for testing)

5. Perform machine learning analysis for the data with seven models (K-nearest neighbor, Random Forest, Decision Tree, Support Vector Machine, Naive Bayes, Gradient Boosting, and Logistic Regression).
6. Evaluate the best machine learning model on accuracy, area under ROC curve (True positive vs False positive rate), and confusion matrix
7. Hyperparamter tuning was performed to improve prediction accuracies.
8. Perform Stratified K-fold Cross Validation and compare accuracies with train-test split (normal machine learning).
9. Using the ANOVA F-test, find another set of top 10 Differentially Methylated Regions (DMRs). This is for Alzheimer's, Control, and Mild Cognitive Impair (MCI). This data will be used for hierarchical clustering.
10. Build a hierarchical clustering model with a dendogram to predict the occurrence of not just Alzheimer's, Control, and Mild Cognitive Impair occurrence using the data of the 10 DMRs obtained with ANOVA.
11. Using Mutual Information for a second time, find the top 2000 DMRs instead of the top 10.
12. Find the corresponding genes that these 2000 DMRs are located in using the original dataset.
13. Using cytoscape, graph the interactions of these genes in a gene interaction network.
14. Use the MCODE algorithm in Cytoscape to find the strongest gene interaction cluster to predict new genes for a potential Alzheimer's Disease therapeutic.
15. Build a Python program to find which genes are most common among DMRs for another way to predict genes.

NOTE: We used hierarchical clustering to predict Alzheimer's Disease, MCI and Control because we found classification was unable to do so (could only predict Alzheimer's Disease vs Control)

Code Snippets - Data Filtering

Identifying the top 10 DMRs: Mutual Information

```
#information gain
from sklearn.feature_selection import mutual_info_classif as MIC
from sklearn.feature_selection import SelectKBest
best = SelectKBest(MIC, k=10)
X_kbest = best.fit_transform(X, y)
print(X_kbest)
cols = best.get_support(indices=True)
features_df_new = X.iloc[:,cols]

print('Original number of features:', X.shape)
print('Reduced number of features:', X_kbest.shape)
```

```
mi_score = MIC(X_kbest,y)
print(mi_score)
```

```
[0.19264773 0.17243542 0.17439579 0.16723334 0.16790445 0.16615951
 0.17311981 0.17650831 0.17988855 0.16598824]
```

Figure 3

The Mutual Information (MI) algorithm was used to identify the top 10 DMRs for the binary classification. Figure 3 shows the implementation of the algorithm in Python (we used SelectKBest to get the top 10 DMRs out of the original dataset). Mutual Information is a quantity that measures the effect of one variable on another and is a useful method of feature selection. In this project, Mutual Information was used to identify which Methylated Regions were DMRs and therefore had a significant effect on Alzheimer's Disease. A high mutual information shows the two variables are highly dependent (highly correlated) and a low mutual information shows the two variables are independent of each other. In this project, we are testing the mutual information between a Methylation site and the occurrence of Alzheimer's Disease. Mutual Information is :

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} = E_{P_{XY}} \log \frac{P_{XY}}{P_X P_Y} .$$

http://www.scholarpedia.org/article/Mutual_information

where x and y are variables in question, and PXY(x,y) is the joint probability distribution between them. In more simple terms, Mutual Information can be defined as:

MI(feature;target) = Entropy(feature) - Entropy(feature|target)

where entropy is a measure of uncertainty. Mutual information is essentially a measure of the loss of entropy, or uncertainty. The Mutual Information Score is a value between 0 and 1, where a higher value will indicate a better score and therefore a better DMR. In the screenshot, the mutual information scores are fairly low, which is one of the biggest limitations to this project. The data we originally downloaded may have flaws which could have reduced the overall MI score. However, machine learning models may still perform well with less ideal data through hyperparameter tuning (discussed later). The top 10 DMRs identified through MI were fed into the 7 classification models (see results).

Identifying the top 10 DMRs: ANOVA F-test

```
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import SelectKBest
fvalue_Best = SelectKBest(f_classif, k=10)
X_kbest = fvalue_Best.fit_transform(X, y)
print(X_kbest)
cols = fvalue_Best.get_support(indices=True)
features_df_new = X.iloc[:,cols]

print('Original numbermm of features:', X.shape)
print('Reduced number of features:', X_kbest.shape)
```

Figure 4

After classification prediction, the next step of our project was to predict not just Alzheimer's Disease/Control, but also predict Mild Cognitive Impair (3 classes) using hierarchical clustering. To do this, Mutual Information cannot be used since it only works for binary data (Alzheimer's vs No Alzheimer's). For multi-class data (Alzheimer's Disease, Control, Mild Cognitive Impair), the ANOVA F-test was used as a measure of feature selection (finding DMRs) as shown in Figure 4. The ANOVA F-Test finds the level of variance between Alzheimer's Disease, Control, and Mild Cognitive Impair samples to identify significant DMRs. In order for a DMR to be significant (cause Alzheimer's Disease/Mild Cognitive Impair) the ANOVA F-value must be

higher than the F-critical value and the p-value (significance) should be less than 0.01. The results are summarized below in Figure 5:

```
#f crit
import scipy.stats
dfn = 2
dfd = 297
scipy.stats.f.ppf(q=1-0.05, dfn=dfn, dfd=dfd)
```

```
3.0261533685653728
```

```
fit = fvalue_Best.fit(features_df_new,y)
print(fit.scores_)
print(fit.pvalues_)
```

```
[10.07311789 10.54066674 10.95856387 10.86863145 10.76073261 11.73166742
 11.29114208 11.98888219 10.00637878 10.27968937]
[5.85177876e-05 3.77930977e-05 2.55960459e-05 2.78327519e-05
 3.07776216e-05 1.24809625e-05 1.87846149e-05 9.83564781e-06
 6.22926562e-05 4.82312585e-05]
```

Figure 5

To calculate the F-critical value, the degrees of freedom numerator (dfn) and degrees of freedom denominator (dfd) must be calculated (Figure 5).
This can be calculated with:

$$dfn = a-1$$
$$dfd = N-a$$

where *a* is the number of groups (3 in our case for Alzheimer's Disease, CL, MCI) and N is the number of samples across all groups (300 in our case for 300 patients). The calculated f-critical value was 3.026. Next, we calculated the actual F-value for our own data. The F-values of the top 10 DMRs were in the range of 10-11 which is well above our F-critical value of 3.026 and therefore shows that the DMRs are statistically significant. The associated P-values are also shown in Figure 5 and most of the p values are extremely low (10^-5 and 10^-6) which also demonstrates significance of the 10 DMRs. These 10 DMRs were fed into a hierarchical clustering model, and the findings of the model will be discussed in the Results section.

Code Snippets - Machine Learning Classification

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 28, leaf_size = 1, p =1)
knn.fit(x_train, y_train)

print("{} NN Score: {:.2f}%".format(28, knn.score(x_test, y_test)*100))
knn_predictions = knn.predict(x_test)
scoreList.append(knn.score(x_test, y_test))
acc = max(scoreList)*100
accuracies['KNN'] = acc
confusion_matrix(y_test, knn_predictions)
#better but still bad - cross validation should be able to improve it
```

```
28 NN Score: 78.95%

array([[15,  2],
       [ 6, 15]])
```

Figure 6

Figure 6 is an example of building a machine learning model in Python. This classification model is K-Nearest Neighbor. First, data is split into 70% for training the model and 30% for testing using the train-test split from sklearn (a python library). The KNN model from sklearn is imported, and the model is fit (learns patterns in data using the KNN algorithm) for the training data. The trained model is tested on the other 30% of the data, and the prediction accuracy is displayed and the confusion matrix is displayed (explained in detail in the results section). The K-nearest neighbor classifies data by looking at similarity based on Hamming distance, or the neighboring points (if other points near the point at question fall under a certain class, then the point at question may fall into that class). Below is the Hamming Distance function (used for categorical prediction):

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

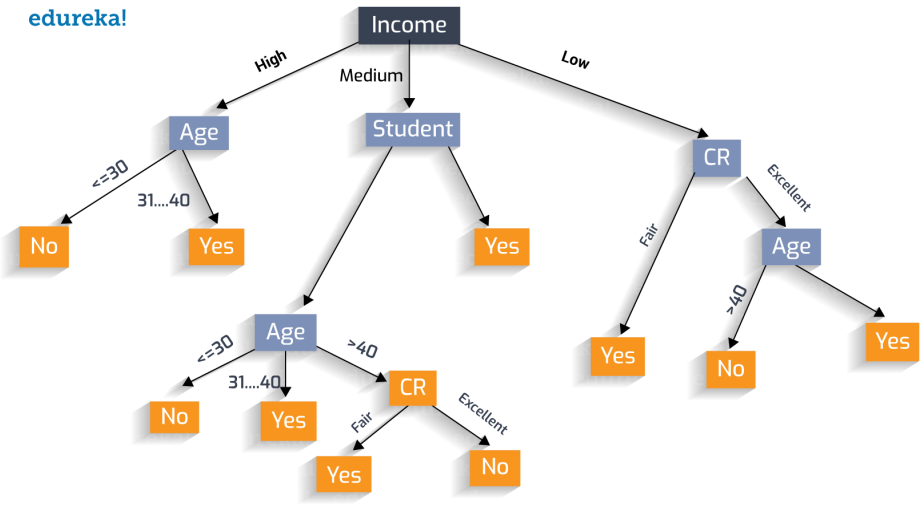
$$x \neq y \Rightarrow D = 1$$

https://www.saedsayad.com/k_nearest_neighbors.htm

where K determines the number of neighbors. The process for building machine learning models is repeated for all other classification models the same way.

Random Forest & Decision Tree:

Decision trees learn by splitting the dataset into smaller subsets to predict a target value (each condition is called a node, and possible outcomes are called "branches"), hence forming a tree. Random forest consists of many individual decision trees that operate as an ensemble (multiple learning algorithms). Decision trees work by creating a series of cases to aid the model into an accurate prediction. The image below is a great example of how it works.



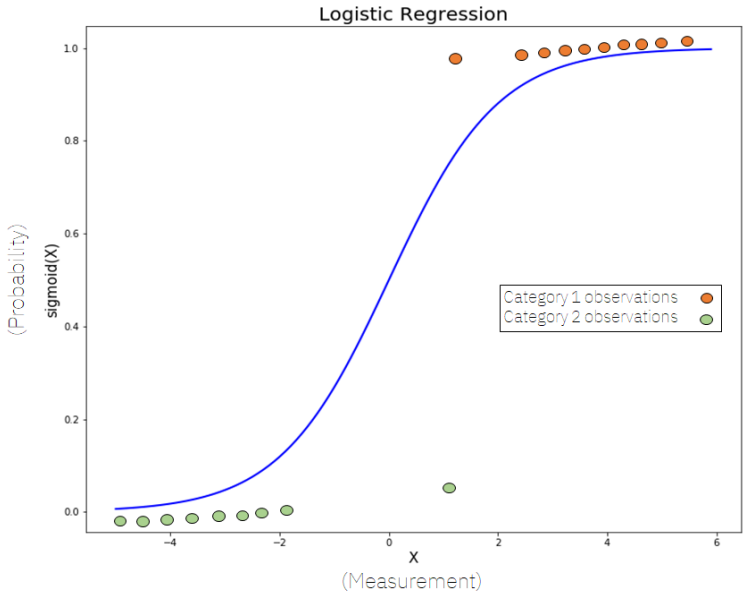
http://blog.davidvassallo.me/2019/08/06/3-uses-for-random-decision-trees-forests-you-maybe-didnt-know-about/

Logistic Regression:

Logistic Regression solves binary classification problems although it's a regression function. The basis of logistic regression is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

The function can take any real value number and map it to a value between 0 and 1 (DNA methylation values can be mapped in this way). The sigmoid curve can be graphed as:



https://towardsdatascience.com/logistic-regression-explained-9ee73cede081

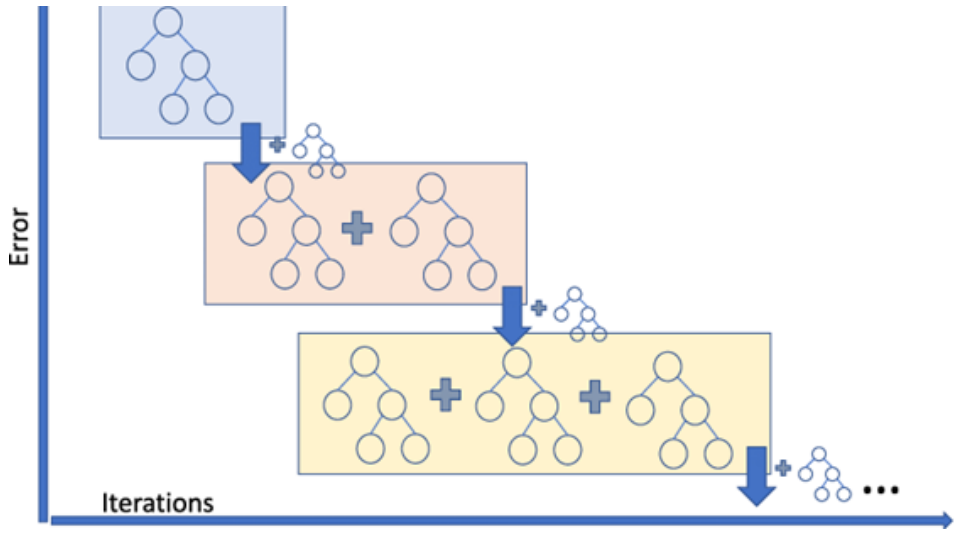
With this method, the location of the point on the curve can determine the classification of that point. In this study, the x-axis would be the DNA Methylation values, and the y-axis would be the probability of Alzheimer's Disease vs Control (No Alzheimer's Disease).

Gradient Boosting:

Examples and Definition

Gradient Boosting is a common way of minimizing the bias error in a classification problem. The system consists of several models that will work recursively to minimize the error. In most cases, the first model of the system is the model that predicts the output based on an input, and the models following the first model will predict the error of the predicted output based on the previous model, and apply that onto the previous output to create a new output.

https://medium.com/analytics-vidhya/what-is-gradient-boosting-how-is-it-different-from-ada-boost-2d5ff5767cb2



As in the image above, the error of the system decreases significantly as more and more models are added. However, this can hold a high risk of overfitting, so a learning rate is applied to the models to make sure the model doesn't overfit the train database, and splitting the train and test cases is used to check if the model is overfitting.

Code Snippets - Hyperparameter Tuning

```
gbc = ensemble.GradientBoostingClassifier()
parameters = {
    "n_estimators":[5,50,250,500],
    "max_depth":[1,3,5,7,9],
    "learning_rate":[0.01,0.1,1,10,100]
}
gridGB = GridSearchCV(gbc, param_grid = parameters,cv=6, refit = True, verbose = 3)
gridGB.fit(x_train, y_train)
[CV 6/6] END learning_rate=0.01, max_depth=1, n_estimators=500; total time=   0.2s
[CV 1/6] END learning_rate=0.01, max_depth=3, n_estimators=5; total time=   0.0s
[CV 2/6] END learning_rate=0.01, max_depth=3, n_estimators=5; total time=   0.0s
[CV 3/6] END learning_rate=0.01, max_depth=3, n_estimators=5; total time=   0.0s
[CV 4/6] END learning_rate=0.01, max_depth=3, n_estimators=5; total time=   0.0s
[CV 5/6] END learning_rate=0.01, max_depth=3, n_estimators=5; total time=   0.0s
[CV 6/6] END learning_rate=0.01, max_depth=3, n_estimators=5; total time=   0.0s
[CV 1/6] END learning_rate=0.01, max_depth=3, n_estimators=50; total time=   0.0s
[CV 2/6] END learning_rate=0.01, max_depth=3, n_estimators=50; total time=   0.0s
[CV 3/6] END learning_rate=0.01, max_depth=3, n_estimators=50; total time=   0.0s
[CV 4/6] END learning_rate=0.01, max_depth=3, n_estimators=50; total time=   0.0s
[CV 5/6] END learning_rate=0.01, max_depth=3, n_estimators=50; total time=   0.0s
[CV 6/6] END learning_rate=0.01, max_depth=3, n_estimators=50; total time=   0.0s
[CV 1/6] END learning_rate=0.01, max_depth=3, n_estimators=250; total time=   0.2s
[CV 2/6] END learning_rate=0.01, max_depth=3, n_estimators=250; total time=   0.2s
[CV 3/6] END learning_rate=0.01, max_depth=3, n_estimators=250; total time=   0.2s
[CV 4/6] END learning_rate=0.01, max_depth=3, n_estimators=250; total time=   0.2s
[CV 5/6] END learning_rate=0.01, max_depth=3, n_estimators=250; total time=   0.2s
[CV 6/6] END learning_rate=0.01, max_depth=3, n_estimators=250; total time=   0.1s
[CV 1/6] END learning_rate=0.01, max_depth=3, n_estimators=500; total time=   0.3s

gridGB.best_estimator_

GradientBoostingClassifier(learning_rate=1, n_estimators=250)
```

Figure 7

When training our classification models, we initially found the accuracies were fairly low in predicting Alzheimer's vs Control (could be from substandard training data with low MI scores). Hyperparameter tuning was used in order to compensate for this, and to boost model accuracies (Figure 7). We used GridSearchCV for hyperparameter tuning, which runs the model in an iterative process with many parameters to evaluate which parameters would make the model have the highest accuracy. The last line of code in Figure 7 shows what parameters to use to get the best model performance. These parameters are re-applied to the model as shown in Figure 8. Figure 8 below shows the change in accuracy after hyperparameter tuning. This process was repeated for all 7 machine learning models.

Before Hyperparamter tuning for Gradient Boosting model (accuracy is 84.21%)

```
#gradient boost
from sklearn import ensemble
gb = ensemble.GradientBoostingClassifier()
gb.fit(x_train, y_train)
acc = gb.score(x_test,y_test)*100
gb_predictions = gb.predict(x_test)
print("Accuracy of Gradient Boosting: {:.2f}%".format(acc))
confusion_matrix(y_test, gb_predictions)
```

```
Accuracy of Gradient Boosting: 84.21%

array([[15,  2],
       [ 4, 17]])
```

After Hyperparamter tuning for Gradient Boosting model (accuracy is 89.47%)

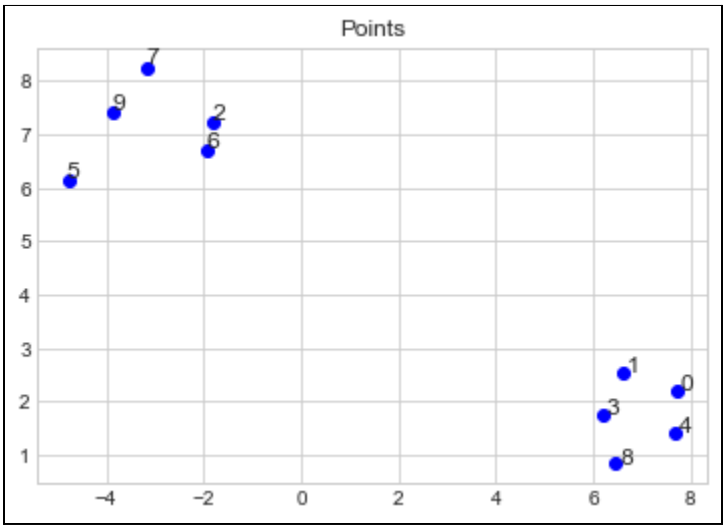
```
gb = ensemble.GradientBoostingClassifier(learning_rate=1, n_estimators=250)
gb.fit(x_train, y_train)
acc = gb.score(x_test,y_test)*100
accuracies['Gradient Boosting'] = acc
gb_predictions = gb.predict(x_test)
print("Accuracy of Gradient Boosting: {:.2f}%".format(acc))
confusion_matrix(y_test, gb_predictions)
```

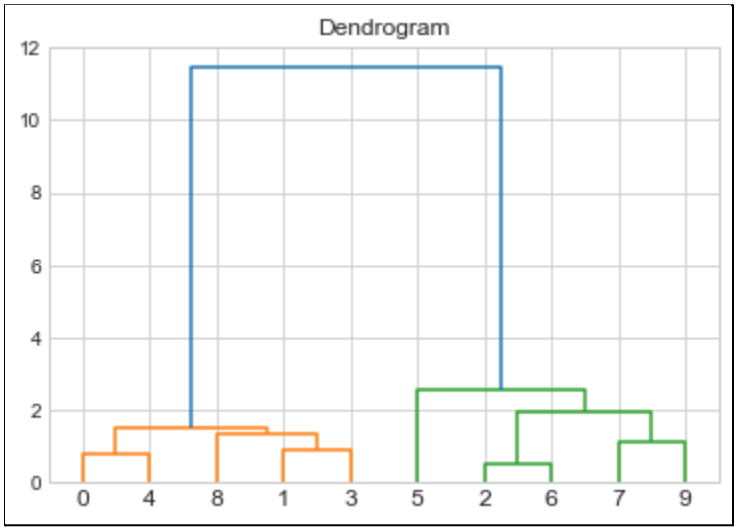
```
Accuracy of Gradient Boosting: 89.47%

array([[16,  1],
       [ 3, 18]])
```

Figure 8

Machine Learning - Hierarchical Clustering

Hierarchical Clustering is a clustering method that aims to cluster a dataset based upon finding the closest points and joining them, and repeating this until every single point on the dataset is in a single group. There are two main types of hierarchical clustering: agglomerative and divisive clustering. Agglomerative clustering is the clustering method described above, in which the model finds the two closest groups/points and joins them to create a new group. However, in divisive clustering, the model starts with a single group consisting of all the points in the dataset, and continually splits them into groups that are the farthest until there are only points left (it is the opposite of agglomerative clustering).





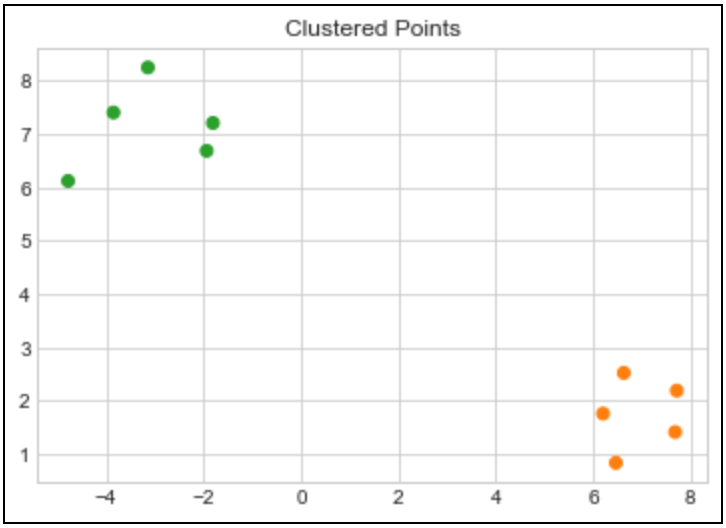


Figure 9

Example

Figure 9 shows examples of how hierarchical clustering works. As you see the graph labeled 'Points' consists of 10 points. Additionally, for the purpose of demonstration, the graph is made so that it consists of two separated groups of points (one on the top right, and another on bottom left). The dendrogram shows how the clustering model joined the points to create the final clusters. The closest pair points are joined together to make the first group (points 0 and 4, points 2 and 6, etc). These pairs of points are joined to create a new point (the x and y values of the new point is the average of the two points). Then, this new point is put through to be compared by other groups of points. The length of each line on the dendrogram shows how close the groups/points are (points 2 and 6 have the shortest lines because they were the closest pairs out of the dataset). Then the final graph titled "Clustered Points" shows the final graph with color coordinated points according to their clusters calculated by the model.

Results

Exploratory Data Analysis

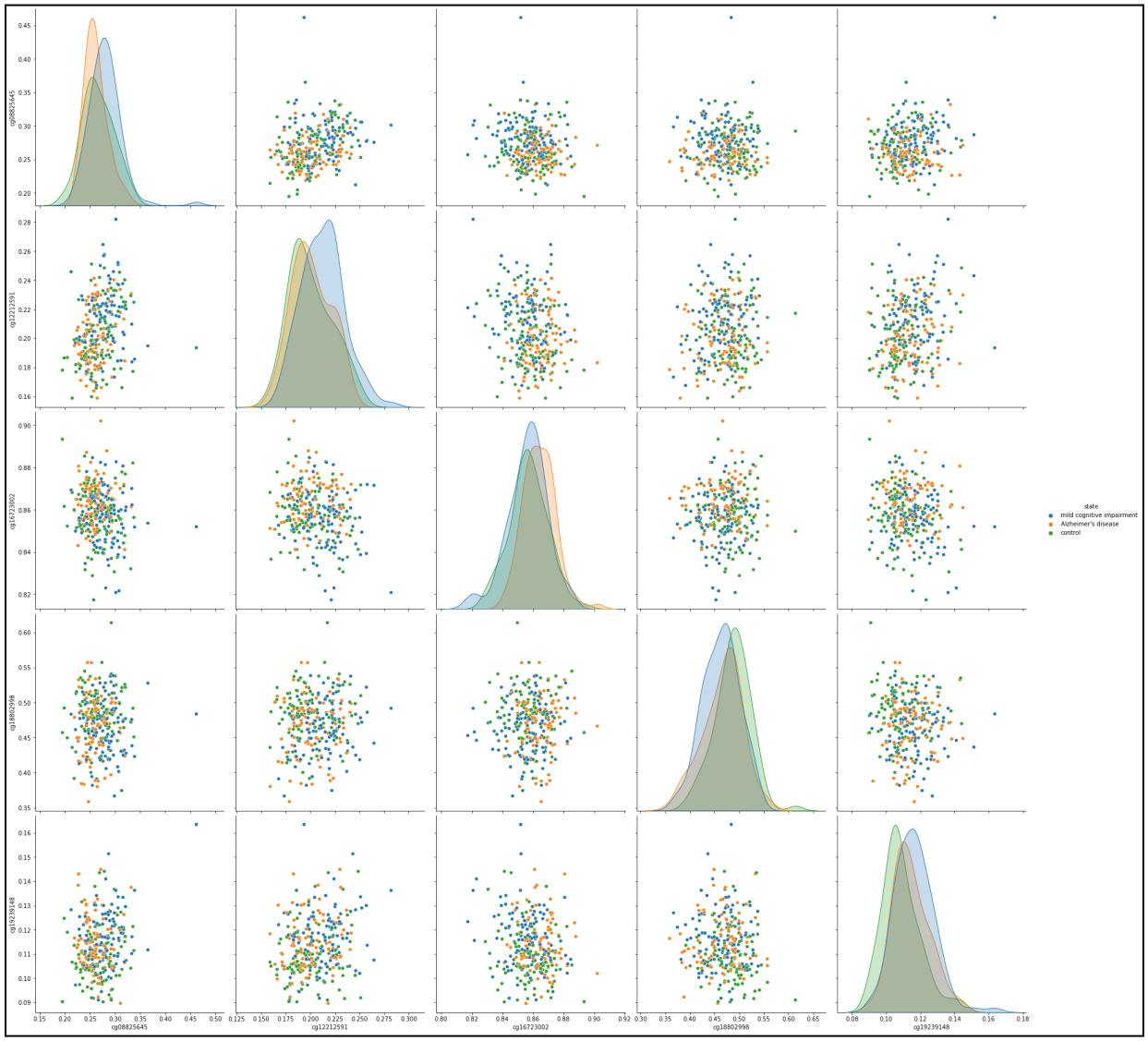
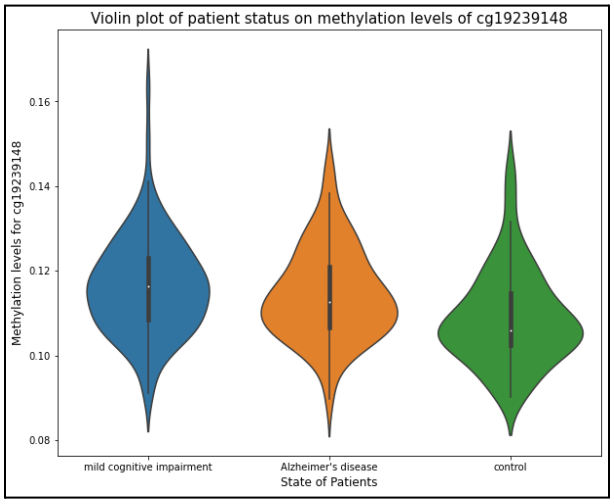


Figure 9

Figure 9 shows a seaborn scatterplot created in Python of the top 5 DMRs which were identified through ANOVA F-test (detailed in methods). The most important graphs are in the diagonal, since they show the distribution of methylation levels for each DMR and what class each distribution curve is classified under. For example, in the last methylated region (cg19239148), the No Alzheimer's curve (shown in green) is behind the Mild cognitive Impair which is shown

in blue. This means that a high methylation level correlates with mild cognitive impair. This can also be visualized with a violin plot and box plot shown below.



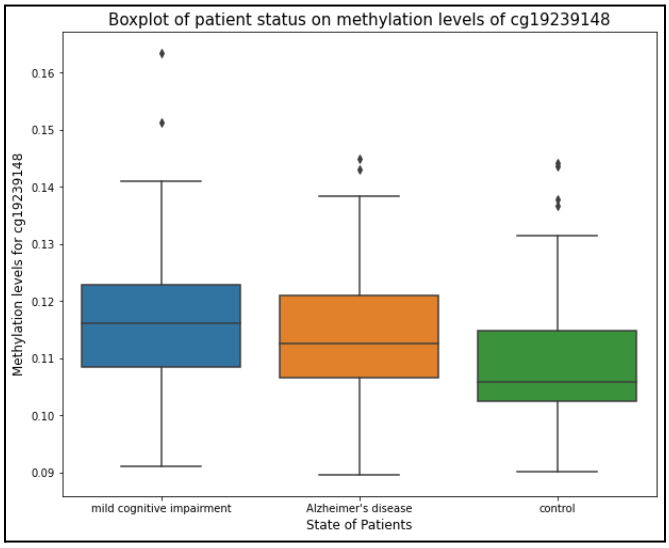


Figure 10

Figure 10 shows the violin plot and the boxplot for the most differentially methylated region in our dataset, **cg19239148** (identified with ANOVA F-test). The violin plot graphs the general distribution of the data as well as the median methylation levels for Alzheimer's Disease, Mild cognitive Impair, and Control (No Alzheimer's Disease). The shape of each plot follows a probability density function, where the widest point indicates a high probability that each sample

will have the given methylation value. A uniform shape shows a normal distribution, where data points are concentrated around the median. From the plots, patients with mild cognitive impairment seem to have the highest DNA methylation levels, then Alzheimer's patients, and finally healthy patients have the lowest DNA methylation levels.

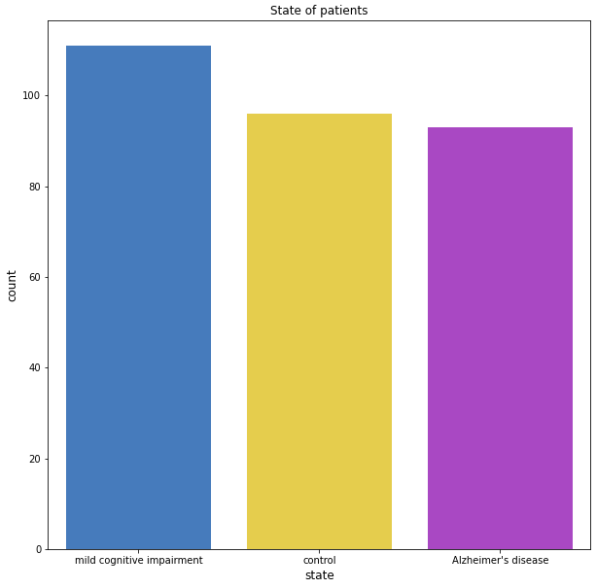


Figure 11

Figure 11 shows the distribution of the patient status in our original dataset. 111 had Mild Cognitive Impair or MCI, 96 had No Alzheimer's Disease - Control, and 93 had Alzheimer's Disease. Imbalances of classes in the dataset will be accounted for with Stratified K-fold Cross Validation (explained in results).

Machine Learning Classification Results

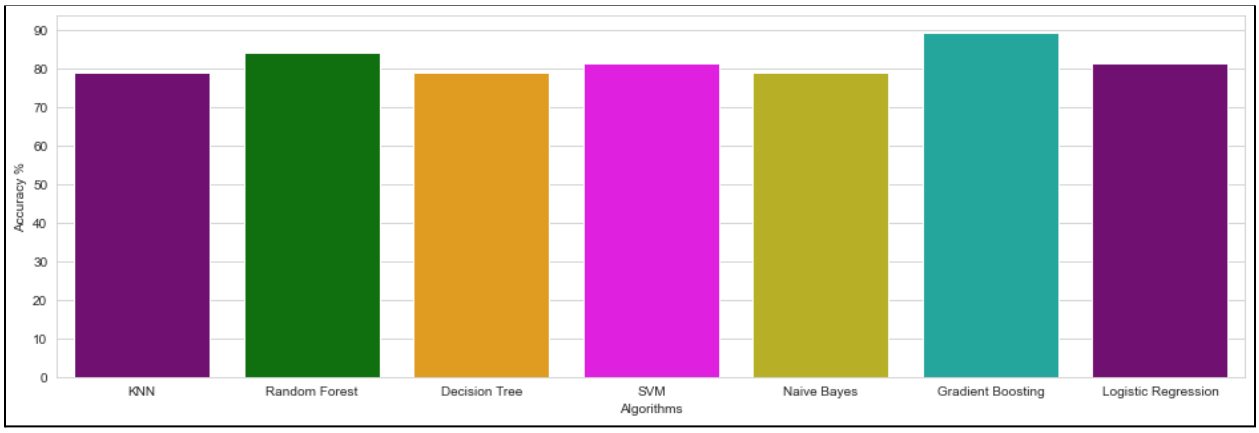


Figure 12

Model accuracies:

‘KNN': 78.95, 'Random Forest': 84.21, 'Decision Tree': 78.95, 'SVM': 81.58, 'Naive Bayes': 78.95, 'Gradient Boosting': 89.47, 'Logistic Regression': 81.58

Figure 12 shows the accuracies of 7 machine learning classification models in predicting Alzheimer's Disease vs Control after being trained on the top 10 DMRs (identified with MI). The average accuracy was 82%, and the highest scoring model was Gradient Boosting with 90% accuracy. Note that these were accuracies measured after hyperparameter tuning. Model performance can also be measured through confusion matrices and ROC curves.

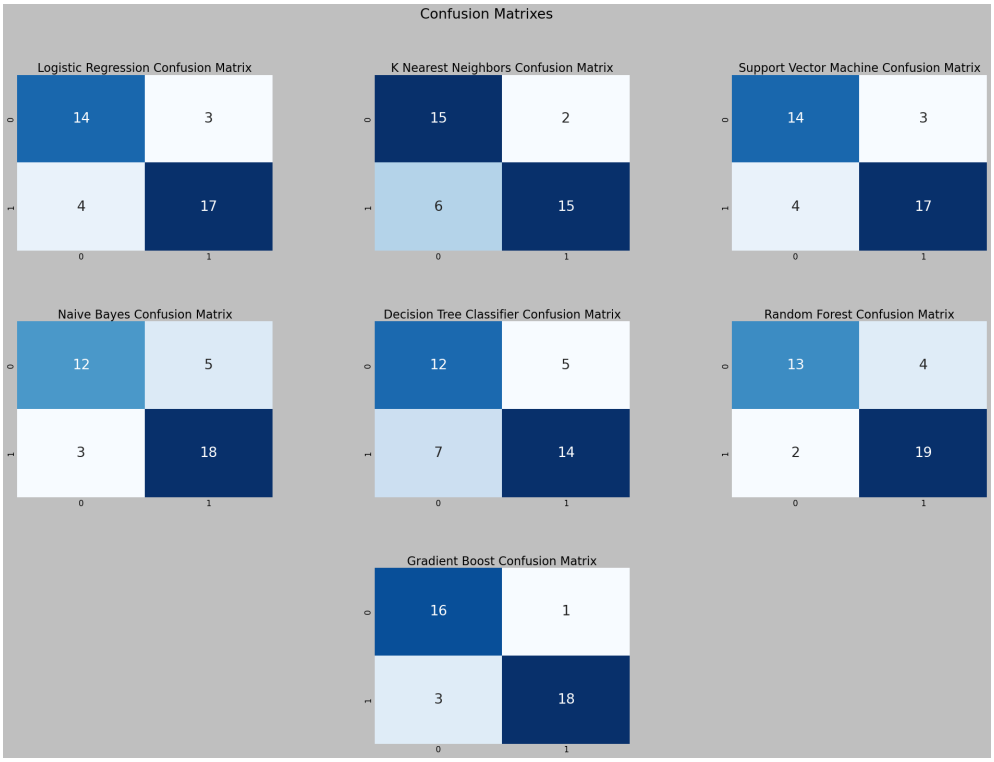


Figure 13

Figure 13 shows the confusion matrix for each of the 7 machine learning models. Confusion matrices evaluate model performance by showing the number of predictions of each class. The table below shows what each number means in each quadrant of the confusion matrix:

	Predicted: Alzheimer's Disease	**Predicted: No Alzheimer's Disease**
Actual: Alzheimer's Disease	True positive	False Positive
Actual: No Alzheimer's Disease	False Negative	True negative

In order to better visualize the confusion matrix, an ROC plot was created (shown below)

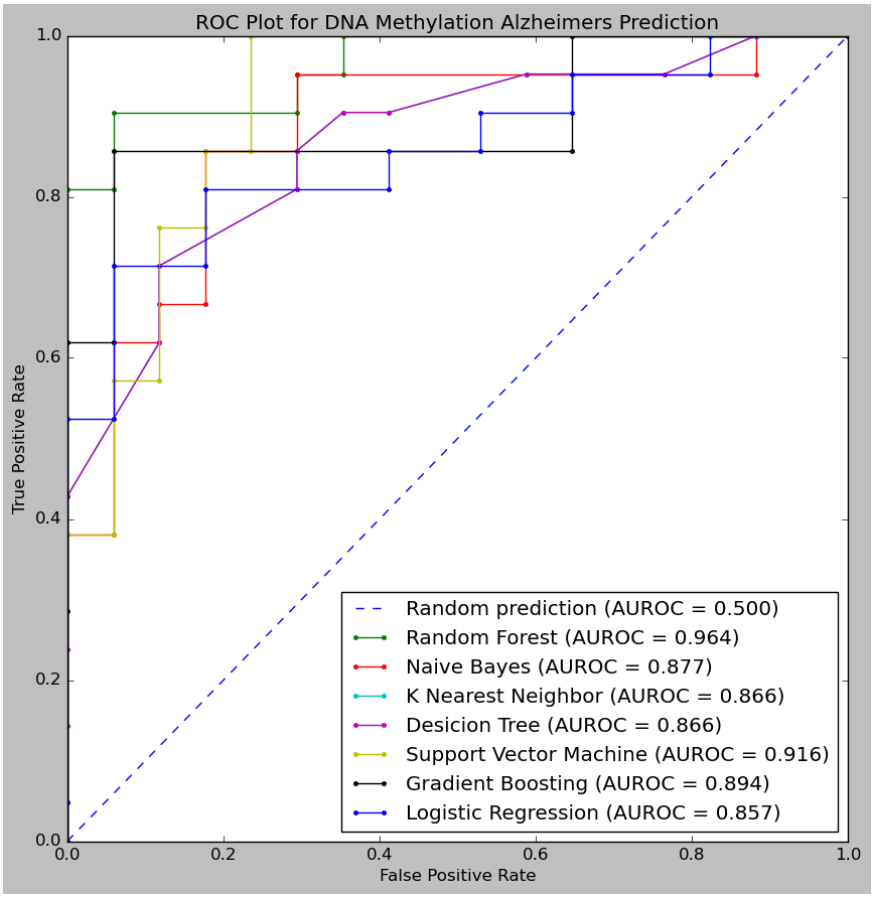


Figure 14

Model performance can also be analyzed using ROC curves (Receiver Operating Characteristic) and AUROC (Area under ROC curve). The ROC curve plots the True positive rate (TPR) vs False positive rate (FPR).

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

TPR = (# of True positives)/(True positives + False Negatives)

$$\text{FPR} = 1 - \text{Specificity} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

(False positive rate = (# of False Positives)/(True Negatives + False Positives)

The AUROC is the area under the ROC curve. A good AUROC is near 1, which shows a good measure of separability (Alzheimer's Disease vs Control). A poor model will have an AUROC near 0, which indicates the model gets every prediction incorrect. A model with AUROC of 0.5 cannot make any separation (cannot differentiate between Alzheimer's Disease and CL). **Figure 14 shows the ROC curve for all models. The random forest model had the highest AUROC of 0.964. The average AUROC was 0.891.**

Stratified K-Fold Cross Validation

Next, Stratified K-fold Cross validation was performed in Python. In regular machine learning, the data was split using the train test split (70% of the data used to train models, and the other 30% used to test the model accuracy). Splitting the data with a train-test split has more bias and can lead to overfitting, so Stratified K-fold splits the data in a different manner. K-fold cross validation involves splitting data into k equal folds (shown by Figure 15 below). The first k-1 folds are used for training, and the remaining are used for testing. This is repeated for all k-folds, and the mean of the accuracies of each k-fold is returned. Stratified k-fold is similar, but involves splitting the data into folds not randomly, but based on the number of each class (if fold 1 has 15 tumor samples and 15 non-tumor samples, then fold 2 should have a roughly equal amount). This helps eliminate biases that come with randomly splitting the data.

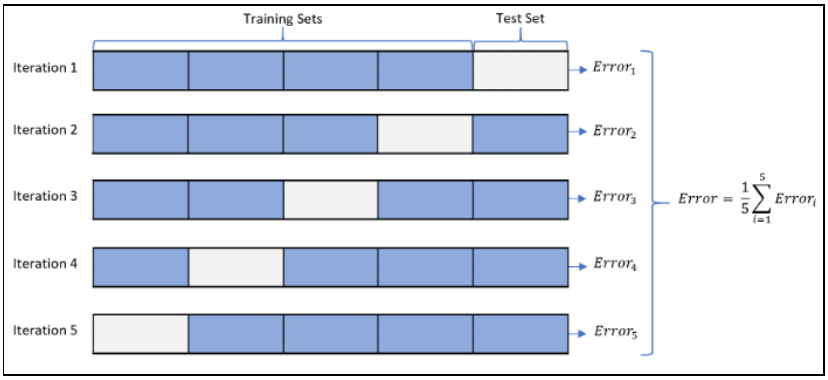


Figure 15 https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-e54df2fc179b

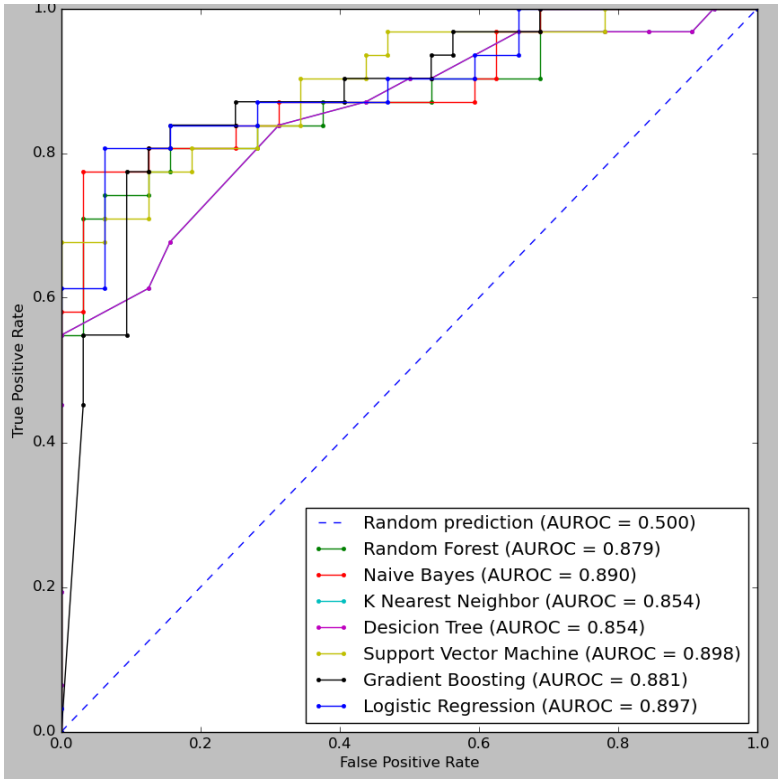


Figure 16

Figure 16 shows the AUROC scores for each of the 7 models after performing Stratified K-Fold Cross Validation. The model with the highest AUROC was the Support Vector machine with a score of 0.898. The average AUROC score was 0.879. This was a little lower than the average for regular machine learning, which was 0.891. This could mean that the models before were slightly overfitting. However, most of the AUROC scores are still fairly high, and random forest

had an AUROC of 0.96 - which has promising potential to predict Alzheimer's Disease in a medical setting.

Machine Learning Hierarchical Clustering Results

Hierarchical Clustering with Alzheimer's Data

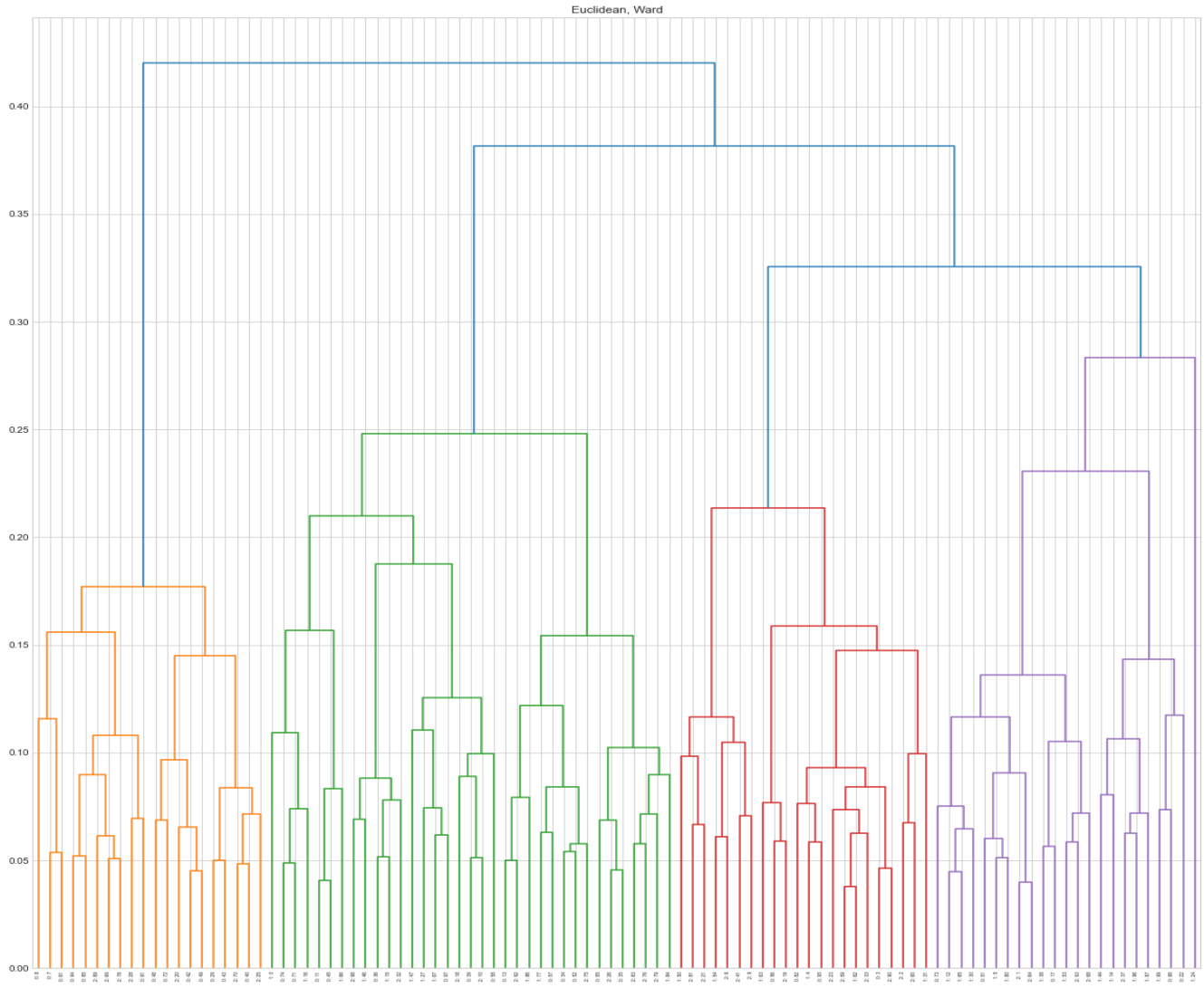
When creating a hierarchical clustering model, the three most important parameters are the following:

- The distance function: which function/formula to use to compute the distance between points (Eg. euclidean, manhattan distance, etc.)
- The joining algorithm: the method used to join/group two points
- The distance threshold OR number of clusters: the distance threshold determines when to cut off the clusters (the distance is the y-axis of the dendrogram), and the number of clusters is the target number of clusters to make

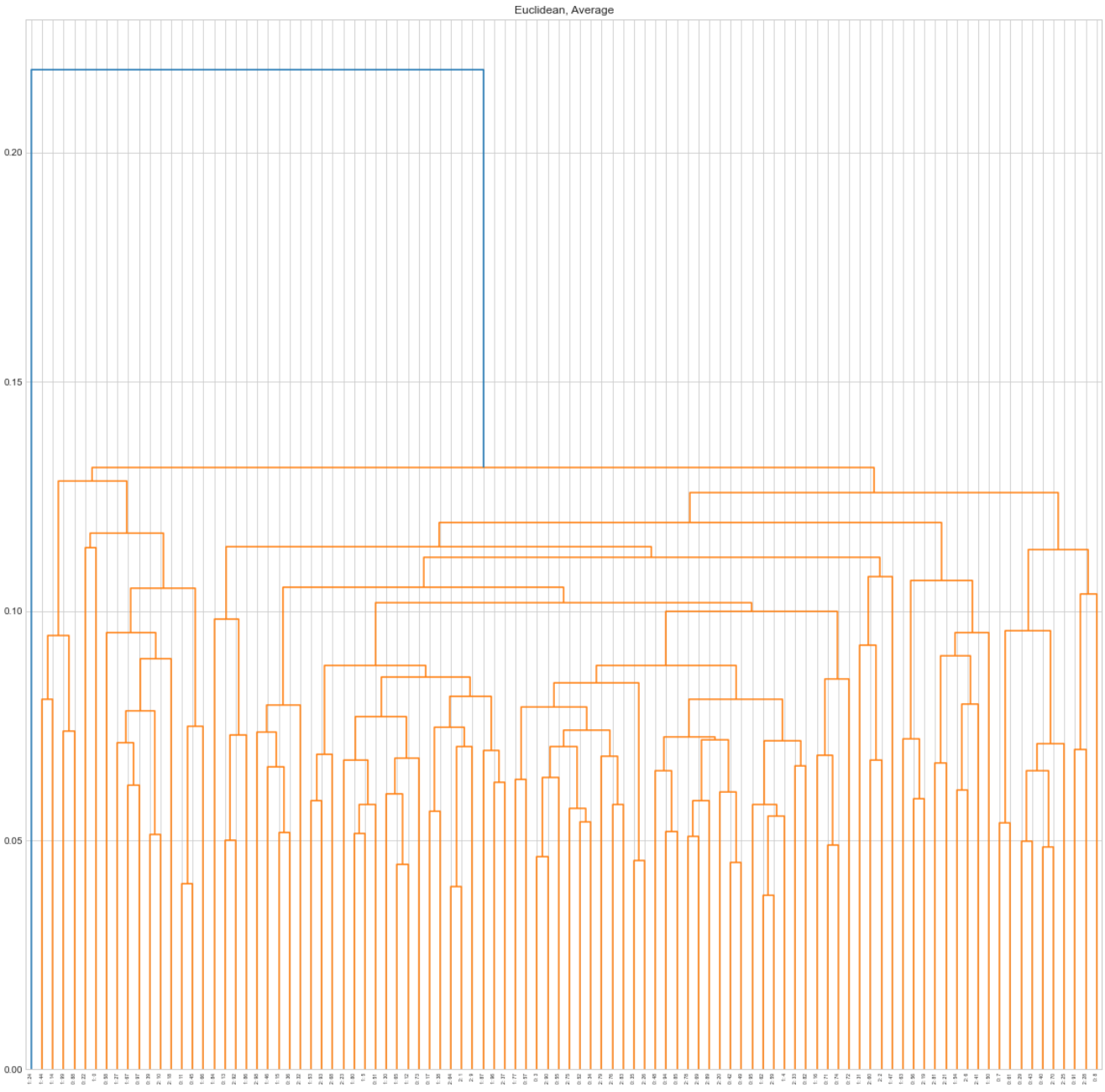
We assigned hierarchical clustering to the task of predicting between the control case, mild imperative impairment, and Alzheimer's (unlike in other models where we let it predict between control and Alzheimer's).

First thing we did with the model was to graph the dendrogram with different joining methods and distance functions.

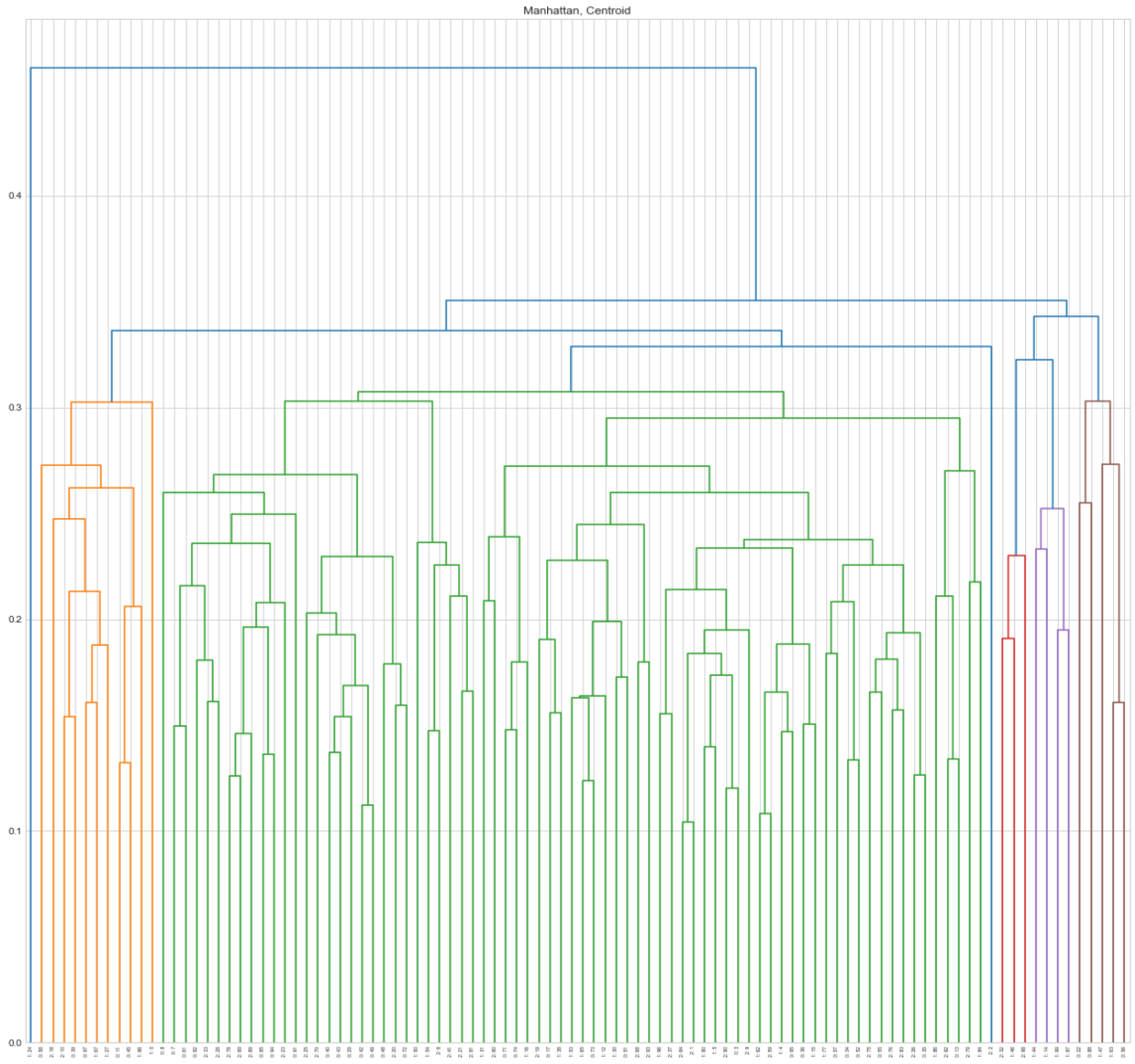
Distance: Euclidean, Joining: Ward



Dist: Euclidean, Join: Average



Dist: Manhattan, Join: Centroid



The dendrograms above were a couple of the dendrograms we collected before creating the final model that included the distance threshold and number of clusters. Based on observations, we decided to use the distance function of euclidean and the merge function of ward to create the final model. The observations were:

- With ward and euclidean, we had an even split between the clusters
- The first merges were consistent, meaning that most of the samples were able to find the first partner at a relatively similar distance

In the final model, we tried restricting the model to a distance threshold and a certain number of clusters (60), and after exploring different parameters with different values, we came to the conclusion that the distance threshold of 0.07 would create a model that created the minimal number of clusters while maintaining the highest accuracy.

```
xs, ys = createData(100)
cluster = AgglomerativeClustering(affinity='euclidean', linkage='ward', n_clusters=None, distance_threshold=0.07)
y_pred = cluster.fit(xs)
def calc_score(y_pred, y):
    arr = {};
    for i in range(0, len(y_pred)):
        arr[y_pred[i]] = [0, 0, 0]
    for i in range(0, len(y_pred)):
        arr[y_pred[i]][ys[i]] += 1;
    avg = 0;
    right = 0;
    for i in range(0, len(y_pred)):
        tot = arr[y_pred[i]][0] + arr[y_pred[i]][1];
        avg += tot;
        if (arr[y_pred[i]][ys[i]] > tot-arr[y_pred[i]][ys[i]]):
            right+=1;
    return right/(len(y_pred))

print("Accuracy: " + str(100 * calc_score(y_pred.labels_, ys)) + "%");
```

```
Accuracy: 86.0%
```

Overall, the accuracy of our hierarchical clustering model in predicting Mild Cognitive Impair, Control, and Alzheimer's Disease was 86%.

]

Gene Predictions

The next step of our project was to predict the top 2000 DMRs using Mutual Information (we only wanted the methylated regions that cause Alzheimer's, so no Mild Cognitive Impair). We found which gene each of the 2000 DMRs correspond to using https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL13534. Because the methylation data is profiled in the whole human genome, we wanted to find which genes our 2000 DMRs are located in. Next, we obtained the name of the genes that the DMRs are located in and graphed the gene interactions in the Cytoscape software. We graphed a gene interaction network since that would allow us to predict genes that cause Alzheimer's Disease and can be used for a potential pharmacogenomic therapeutic for Alzheimer's Disease. It is important to identify new genes that can cause Alzheimer's Disease since that can provide more options for therapeutics. Our objectives to predict genes were twofold:

Objective 1: Predict genes that are known to cause Alzheimer's Disease (using MCODE, Python)
Objective 2: Predict genes that have not yet been explored (using MCODE, Python)

Gene Interaction Network

Before either objective could be completed, first a gene interaction network had to be constructed in cytoscape. The interaction network would show the interactions of genes that the top 2000 DMRs are located in. It is important to look at gene interactions since 1 gene may not have the ability to cause Alzheimer's Disease, but a series of many genes that interact with each other can have the potential to cause Alzheimer's Disease. Additionally, the MCODE algorithm needs a gene interaction network built to make predictions. Below is the gene interaction network for the genes of the top 2000 DMRs. This shows a small glimpse of the center of the network (most important) since the network was too large to attach.

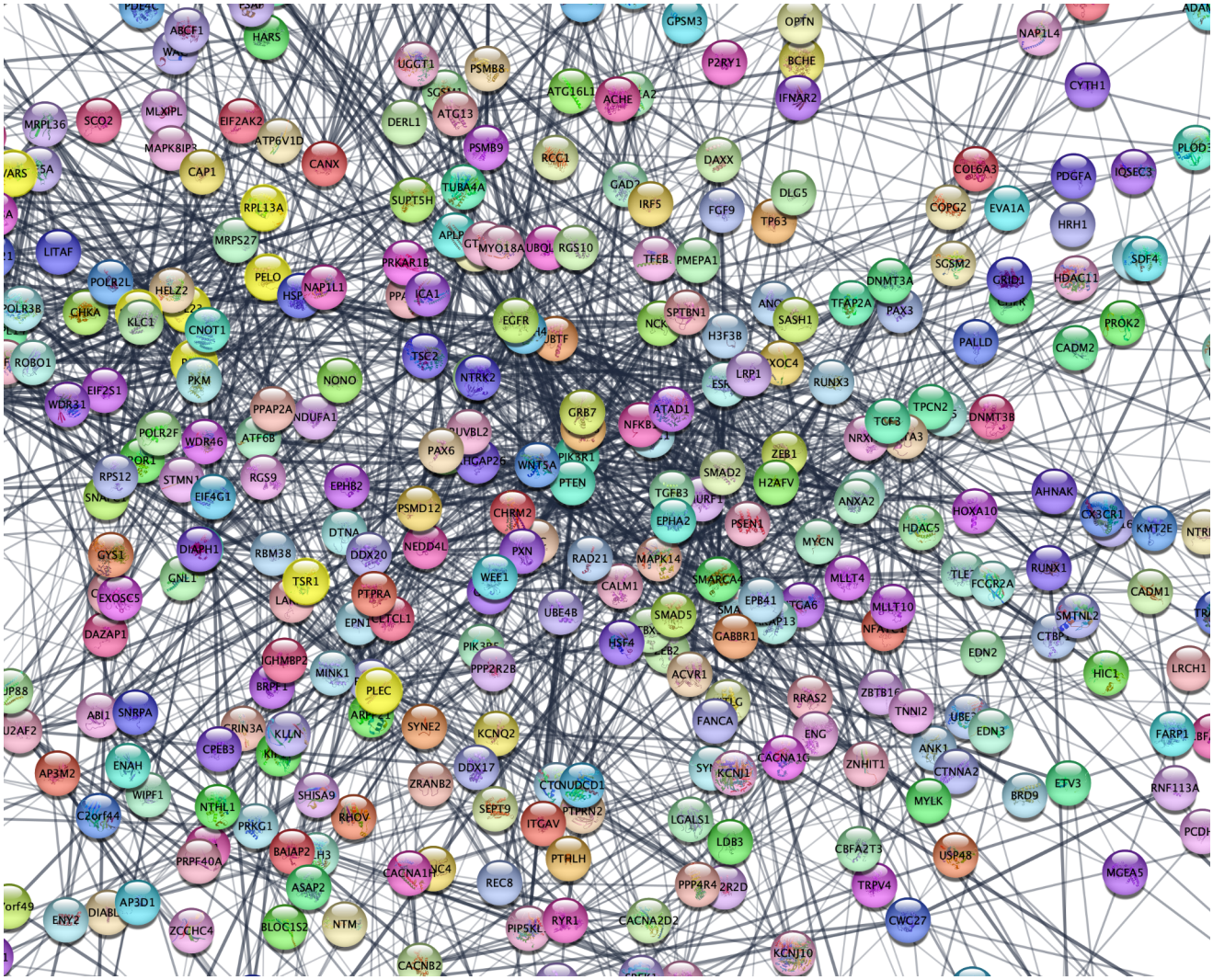


Figure 17

As we mentioned, many genes interacting with each other have a higher probability of causing Alzheimer's Disease. MCODE is an algorithm that finds the top scoring cluster of gene interactions in the network above. These clusters of genes can interact with each other and therefore cause Alzheimer's disease. The top scoring cluster of genes predicted by MCODE are highlighted in a yellow color above. These genes are **PLEC, TSR1, RPL5, RPL23, SERBP1, PELO, RPL13A, RSP13, ATP5A1, VARS**. In addition to MCODE, we built a python program to see if a lot of the 2000 DMRs are located in a certain gene. If they are, that gene could be another potential Alzheimer's-causing gene (if a lot of DMRs are located in it, it

can result in excessive methylation which can cause AD). We are essentially finding the most frequently occurring gene in the list of genes of the 2000 DMRs. The findings are shown below:

[('PTPRN2', 13), ('MINK1', 13), ('IQCE', 12), ('MADL1', 11), ('HPGD', 11)]

The number shows the number of DMRs located in that gene. For example, the PTPRN2 gene has 13 DMRs located in the gene. If a lot of DMRs are found in a certain gene, it has a high probability to cause Alzheimer's Disease. In total, we predicted 15 genes (from MCODE and the Python program). In order to find out which of these 15 genes cause Alzheimer's Disease and which ones cause other disorders, the Gene Cards website was used. The results are summarized in the table below.

Blue Highlight: MCODE genes
Green Highlight: Python Predicted genes

Predicted Genes	Gene Cards Validation (What disorder is the gene known to play a role in)
PLEC	Muscular Dystrophy, Limb-Girdle, Autosomal Recessive, Epidermolysis
TSR1	Epithelial Malignant Thymoma, Dendritic Cell Thymoma, Geleophysic Dysplasia, Sick Building Syndrome
RPL5	Diamond-Blackfan Anemia
RPL23	Brain Glioblastoma Multiforme
SERPB1	Contagious Pustular Dermatitis, Ovarian Cancer
PELO	No disorders in Gene Cards
RPL13A	Sclerosteosis, Spermatogenic Failure

RPS13	Gaucher Disease, Neuroblastoma, Autonomic Nervous System Neoplasm
ATP5F1A	Combined Oxidative Phosphorylation Deficiency 22
VARS	Neurodevelopmental Disorder With Microcephaly, Seizures, And Cortical Atrophy
PTPRN2	Developmental Coordination Disorder
MINK1	**Alzheimer's Disease**
IQCE	Retinal Degeneration
MADL1	Prostate Cancer
HPGD	Digital Clubbing, Isolated Congenital

Some of these genes have no significant relation to Alzheimer's or any brain disorder. The most important genes (involved in Alzheimer's or other brain diseases) are **RPL23, RPS13, VARS, and MINK1.** MINK1 is proven to cause Alzheimer's, and we were able to predict it- which validated our methodology. RPL23, RPS13, and VARS aren't known to cause Alzheimer's Disease outside of a few published papers, but can still be novel genes that could be targeted via therapeutics. The most significant finding to this project however was that MINK1, one of the genes we predicted, is known to cause Alzheimer's Disease. The link to the gene card validation is here: https://www.genecards.org/cgi-bin/carddisp.pl?gene=MINK1&keywords=MINK1.

Conclusion

Our project featured various machine learning models to predict the severity of patients' Alzheimer's Disease with DNA methylation levels, and to predict genes that can be used as therapeutic targets for Alzheimer's Disease. We initially cleaned the data to only include the top 10 most differentially methylated regions, in order to ensure efficiency during our process. This later proved to be useful because some of our models, such as hierarchical clustering, proved to be time inefficient, and by trimming the data down, we were able to save time and remain efficient with our process. Throughout model building, we experimented with various methods to create models with high accuracy (without overfitting). Such methods included hyperparameter tuning, which finds the best parameter for the models to use to achieve the best accuracy. We also decided to make the models predict between control (no Alzheimer's Disease) and Alzheimer's cases rather than control, mild cognitive impairment, and Alzheimer's disease. We did so because during our process, we learned that classification models could not distinguish between mild cognitive impairment and Alzheimer's. Knowing the difficulties of distinguishing between the three states, we explored hierarchical clustering to see if it was possible, and it was proved that hierarchical clustering can distinguish between the three states with 86%.

The main findings of this project is 1) Machine Learning Classification can predict Alzheimer's/No Alzheimer's with 82% accuracy on average; 2) The highest scoring classification model was Gradient Boosting with 90% accuracy; 3) Hierarchical Clustering can predict Alzhiemer's, Control (No Alzheimer's), and Mild Cognitive Impair with 86% accuracy; 4) RPL23, RPS13, VARS were predicted using the MCODE algorithm and are known to cause many brain disorders. **More importantly, the MINK1 gene contains the most DMRs upon Python analysis and is proven to cause Alzheimer's Disease, which makes it the best target for an Alzheimer's therapeutic.** In the future, we would like to make more machine learning models such as neural networks to see how that might affect model accuracy and maybe try to improve the model accuracy of hierarchical clustering. We would also like to look into larger datasets and DNA methylation profiles in more diseases. The main limitation to our project is with the model accuracies, since they are still slightly low, and we would like to try to improve them in the future. Another limitation is the data. Our dataset initially showed fairly low mutual information scores which doesn't show a strong correlation of DMRs with Alzheimer's disease,

even though it is a biologically known fact. This could be from substandard data which was downloaded, so in the future we would like to find other sources of data.

This project is important because it allowed us to explore the non-hereditary causes of Alzheimer's Disease such as epigenetics processes, which can be changed by environmental, lifestyle, and aging factors, and can play such a large role in causing diseases. It showed us that we have some control over the occurrences of diseases such as Alzheimer's, and how we have control over many aspects of our health and body. Additionally, our machine learning models can offer an alternative to current diagnostic methods such as MRI or CSF tests for a safer, cheaper method in predicting Alzheimers. Finally the genes (especially MINK1) we predicted in this project can be used as a pharmacogenomic (drug targeting of genes) therapy for Alzheimer's Disease.

Most Significant Achievement

In this project, we were able to use machine learning classification to predict Alzheimer's Disease with 90% accuracy. Additionally, we were able to successfully use hierarchical clustering to predict Mild Cognitive Impair and Alzheimer's disease with 86% accuracy. Finally, we predicted the MINK1 gene which is known to cause Alzheimer's Disease using Python analysis of DMRs, and we also predicted the genes RPL23, RPS13, VARS which could be used for a novel AD therapeutic.

Acknowledgements

We would like to thank Ms. Yolanda Lozano, our teacher, for supporting and guiding us through the project and our parents for their encouragement and support. We would also like to thank the Supercomputing Challenge Judges for giving us feedback during the midterm interview.

Bibliography

1. Centers for Disease Control and Prevention. (2020, August 3). *What is epigenetics?* Centers for Disease Control and Prevention. Retrieved April 5, 2022, from https://www.cdc.gov/genomics/disease/epigenetics.htm
2. Choudhury, A. (2020, December 24). *What is gradient boosting? how is it different from Ada Boost?* Medium. Retrieved April 6, 2022, from https://medium.com/analytics-vidhya/what-is-gradient-boosting-how-is-it-different-from-ada-boost-2d5ff5767cb2
3. *DNA methylation*. What is Epigenetics? (2019, September 5). Retrieved April 5, 2022, from https://www.whatisepigenetics.com/dna-methylation/
4. KNN classification. (n.d.). Retrieved April 6, 2022, from https://www.saedsayad.com/k_nearest_neighbors.htm
5. Latham, P. E., & Roudi, Y. (2009, January 21). *Mutual information*. Scholarpedia. Retrieved April 6, 2022, from http://www.scholarpedia.org/article/Mutual_information
6. Pal, D. S. K. (2019, July 15). *Epigenetic processes and human health*. Skyline University Nigeria. Retrieved April 6, 2022, from https://www.sun.edu.ng/knowledge-update/epigenetic-processes-and-human-health
7. Patro, R. (2021, February 1). *Cross validation: K Fold vs Monte Carlo*. Medium. Retrieved April 6, 2022, from https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-e54df2fc179b
8. U.S. National Library of Medicine. (n.d.). *Geo accession viewer*. National Center for Biotechnology Information. Retrieved April 6, 2022, from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL13534
9. U.S. National Library of Medicine. (n.d.). *Geo accession viewer*. National Center for Biotechnology Information. Retrieved April 6, 2022, from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE144858
10. U.S. National Library of Medicine. (n.d.). *Geo accession viewer*. National Center for Biotechnology Information. Retrieved April 6, 2022, from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE144858
11. Vassallo, D. (2019, August 6). *3 uses for Random decision trees / forests you (maybe) didn't know about*. David Vassallo's Blog. Retrieved April 6, 2022, from http://blog.davidvassallo.me/2019/08/06/3-uses-for-random-decision-trees-forests-you-maybe-didnt-know-about/
12. z_ai. (2021, September 26). *Logistic regression explained*. Medium. Retrieved April 6, 2022, from https://towardsdatascience.com/logistic-regression-explained-9ee73cede081