Detecting Pre-Symptomatic Cognitive Impairment through Epigenetic Biomarkers and Machine Learning for Early Intervention

By:

William Guo

March 1, 2021

Northwood High School

Advisor - Angie Olivares

Abstract

The purpose of this project was to test the hypothesis of whether DNA methylation can be used as a biomarker to predict Alzheimer's disease and to create a method of detecting pre-symptomatic cognitive impairment that is simple, inexpensive, and minimally invasive. This was accomplished through the use of data analysis programs to discover specific CpG site biomarkers from case-control datasets based on probability (p) values and machine learning to create static and progressive models. These models utilize the biomarkers to distinguish mild cognitive impairment patients from controls as well as predict one's status (No Disease, Mild Cognitive Impairment, Alzheimer's Disease) in two years based on their current status respectively. Hundreds of sites with a p-value under 1E-5 from the GSE156984, GSE153712, and ADNI datasets were found to be significant regarding the development of Alzheimer's. These sites were then used along with machine learning algorithms to create the static and progressive models. The best static model utilized 300 sites (GSE156984, GSE153712) found using the lasso algorithm and the SVC algorithm, with a testing accuracy of 80.3%. Furthermore, the ND to MCI conversion progressive model built using 79 sites (ADNI) and SVC had a testing accuracy of 86.1% while the MCI to AD conversion model which used 80 sites (ADNI) and SVC had a testing accuracy of 90.9%. In conclusion, it can be reasonably assumed that the hypothesis regarding the feasibility of using DNA methylation as a biomarker to predict Alzheimer's disease was supported.

Table of Contents

Introduction

In a world tasked with addressing a multitude of prevalent issues ranging from climate change to racial inequality, it is common to lose sight of many significant problems that see less public and media attention. One such example is the dramatic increase in cognitive decline, specifically Alzheimer’s disease. Alzheimer’s disease is a progressive type of dementia that damages and eventually destroys brain cells, affecting memory and other cognitive functions. In addition to the human suffering caused by the disease, Alzheimer’s creates a massive strain on not just the victim’s families, but also on the healthcare system and federal budget (Alzheimer’s Association). According to the same source, the number of people in the U.S who lived with Alzheimer’s disease in 2020 totaled 5 million, which is projected to nearly triple to 14 million by 2050. Though a cure currently does not exist for the disease, early and accurate diagnosis is vital in regulating its progression while providing the patient a continued sense of normalcy. Despite this, many Alzheimer’s patients are only diagnosed upon experiencing symptoms, by which point irreversible damage has already been done. Additionally, the methods used for confirmatory diagnosis such as positron emission tomography (PET) scans and cerebrospinal fluid (CSF) procedures can be expensive as well as invasive. These major downsides discourage people from even taking measures to examine their mental health, further decreasing the likelihood of detecting any signs of pre-symptomatic cognitive impairment.

However, recent studies that have reported an association between DNA methylation and Alzheimer's disease may hold the key to the creation of prediction-based models. DNA methylation is a vital epigenetic mechanism where a methyl group (CH_3) is added to DNA/CpG (Cytosine - Phosphate - Guanine) sites, typically affecting gene expression. The article “DNA methylation alterations in Alzheimer’s disease” featured in Oxford Academic states that aberrant methylation in the peripheral blood is correlated with AD disease status. It can then be speculated that the methylation of specific CpG sites may subsequently result in the development of the disease.

Therefore, the objective of this project was to test whether DNA methylation can be used as a biomarker to diagnose Alzheimer's disease and create a method of presymptomatic diagnosis of cognitive impairment that is accessible, inexpensive, and minimally invasive. This was accomplished through the creation of machine learning models that aim to assist medical professions in their diagnosis decision-making by not only determining a patient's disease status but also predicting their future status in 2 years based on their current condition.

Materials and Methods

Materials:

Datasets:

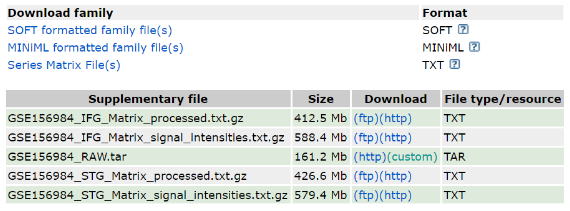
Static Models:

The two datasets utilized in the static models are obtainable in PubMed and GEO (Gene Expression Omnibus), which contain a large number of scientific articles, research, and publications that are available to the public. (Link: https://www.ncbi.nlm.nih.gov/pubmed/).

GSE156984 (Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE156984)
- 728,553 CpG Sites
- 127 Alzheimer's Disease
- 117 No Disease

*DNA methylation values obtained from IFG & STG, which are regions in the brain)

Figure 1: *GSE156984 Downloadable Files on GEO*



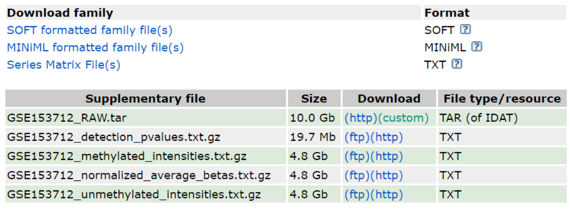
Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE156984_IFG_Matrix_processed.txt.gz	412.5 Mb	(ftp)(http)	TXT
GSE156984_IFG_Matrix_signal_intensities.txt.gz	588.4 Mb	(ftp)(http)	TXT
GSE156984_RAW.tar	161.2 Mb	(http)(custom)	TAR
GSE156984_STG_Matrix_processed.txt.gz	426.6 Mb	(ftp)(http)	TXT
GSE156984_STG_Matrix_signal_intensities.txt.gz	579.4 Mb	(ftp)(http)	TXT

GSE153712 (Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE153712)
- 832,602 CpG Sites
- 161 Alzheimer's Disease
- 94 Mild Cognitive Impairment
- 471 No Disease

*DNA methylation values obtained from peripheral blood)

Figure 2: *GSE153712 Downloadable Files on GEO*



Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE153712_RAW.tar	10.0 Gb	(http)(custom)	TAR (of IDAT)
GSE153712_detection_pvalues.txt.gz	19.7 Mb	(ftp)(http)	TXT
GSE153712_methylated_intensities.txt.gz	4.8 Gb	(ftp)(http)	TXT
GSE153712_normalized_average_betas.txt.gz	4.8 Gb	(ftp)(http)	TXT
GSE153712_unmethylated_intensities.txt.gz	4.8 Gb	(ftp)(http)	TXT

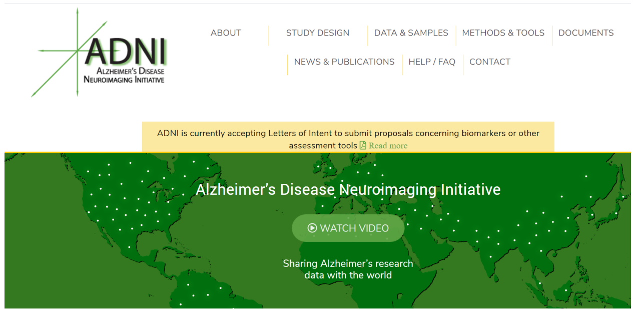
Progressive Models:

The dataset used in the progressive models is the ADNI DNA methylation dataset. ADNI (Alzheimer's Disease Neuroimaging Initiative) "is a longitudinal multicenter study designed to develop clinical, imaging, genetic, and biochemical biomarkers for early detection and tracking of Alzheimer's disease". In order to gain access to the data, I had to submit a Data Use Agreement form, a general application, and be granted access by the IDA collaboration, which is part of the Laboratory of Neuroimaging (LONI - University of Southern California). (Link: http://adni.loni.usc.edu/)

ADNI DNA Methylation Dataset:
- 865,859 CpG Sites
- 400 Alzheimer's Disease
- 895 Mild Cognitive Impairment
- 610 No Disease

  *Please note that most statuses listed in the dataset come from repeating patients/samples in order to track disease progression.

Figure 3: *ADNI Website Home Page*


ADNI
Alzheimer's Disease Neuroimaging Initiative
ABOUT
STUDY DESIGN
DATA & SAMPLES
METHODS & TOOLS
DOCUMENTS
NEWS & PUBLICATIONS
HELP / FAQ
CONTACT
Read more
Alzheimer's Disease Neuroimaging Initiative
WATCH VIDEO
Sharing Alzheimer's research data with the world

ADNI is currently accepting Letters of Intent to submit proposals concerning biomarkers or other assessment tools

Physical Computer Hardware:

1 Personal Computer: 1.8 GHz processor, 8 GB RAM, Windows 10

Websites, Programming Languages, and Computer Software:

Anaconda 3 (Jupyter Notebook - Python Language)	RStudio (R Language)	Linux (Amazon Web Service)
WinSCP	Microsoft Excel	DAVID Bioinformatics

Figure 4: *Anaconda 3*

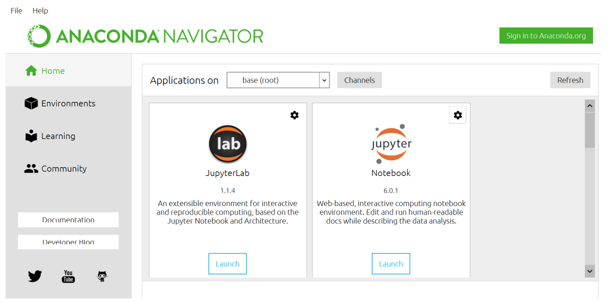

File Help
ANACONDA NAVIGATOR
Sign in to Anaconda.org
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Figure 5: *Python 3 in Jupyter Notebook*

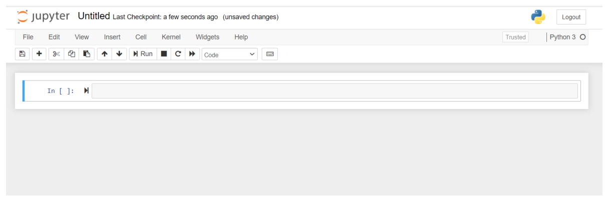
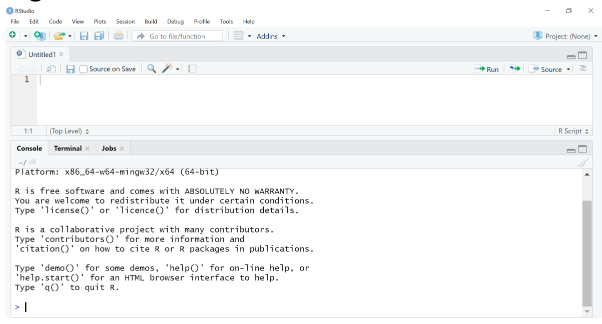
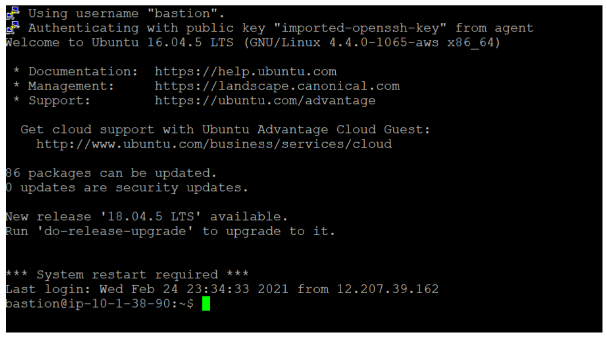

jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [ ]:

Figure 6: *RStudio*



```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Figure 7: *Linux*



```
   Using username "bastion".
   Authenticating with public key "imported-openssh-key" from agent
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1065-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

86 packages can be updated.
0 updates are security updates.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Wed Feb 24 23:34:33 2021 from 12.207.39.162
bastion@ip-10-1-38-90:~$ 
```

Figure 8: *WinSCP*

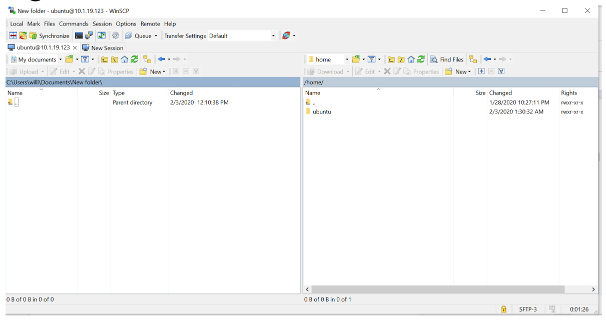


Figure 9: *Microsoft Excel*

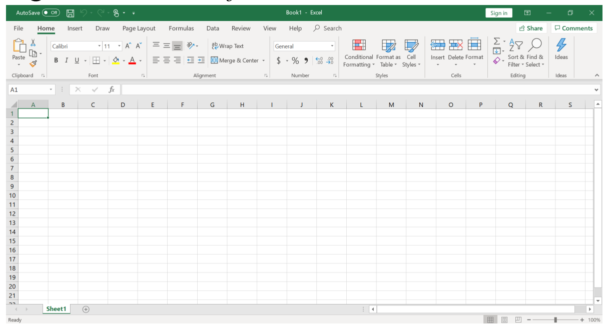
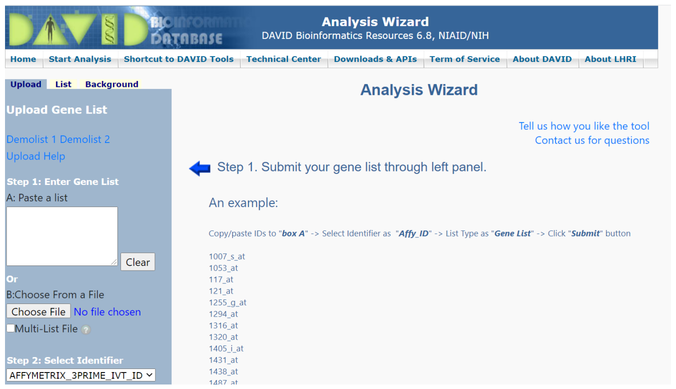


Figure 10: *DAVID Bioinformatics*



Step 1. Submit your gene list through left panel.

An example:

Copy/paste IDs to "*box A*" -> Select Identifier as "*Affy_ID*" -> List Type as "*Gene List*" -> Click "*Submit*" button

1007_s_at
1053_at
117_at
121_at
1255_g_at
1294_at
1316_at
1320_at
1405_i_at
1431_at
1438_at
1487_at

Methods:

Static Models:

1) Download Datasets and other Important Files

- Use wget in Linux to download GSE156984 dataset and Series Matrix File by copy and pasting the files' address link.
- Use wget to also download the GSE153712 Series Matrix File and load() in R for the dataset's normalized value file.

2) Process and Format Data

- Unzip files in Linux using the command gunzip and format each file in R.
- Extract sample ID's and disease status from Series Matrix file for each dataset.
- Calculate cell composition of the blood samples from GSE153712 using the R library cellDeconMeth.

3) Transfer Files to Personal Computer

- Use WinSCP to transfer all files needed from Linux to personal computer.
- Attach .csv to the end of the filenames, which allows it to be opened in Excel.

4) Find Important CpG Sites

- Attach respective status and samples to datasets in Excel.
- Sort files by status and perform t_test. For GSE153712, do t_test in terms of different status combinations (AD vs. ND, AD vs. MCI, MCI vs. ND).
- Record sites with a p-value under 1E-5.
- Compare and find similar sites between GSE156984, AD vs. ND, AD vs. MIC, and MCI vs. ND site list.

5) Create Machine Learning Models

- In Jupyter Notebook, create 3 models using 4 machine learning algorithms (SVC, LogisticRegression, RandomForest, GradientBoosting) with only data from MCI and ND samples.
    - 1) 6 specific sites
    - 2) Sites with p-value < 1E-6 (594 sites)
    - 3) Collection of 5 models with different amounts of sites (1E-6) found using the lasso algorithm
- Create Area Under the Curve graphs for each model.

Progressive Models:

1) Download Datasets and other Important Files

- Use R program minfi to download and convert original .idat file from ADNI to beta (methylation) file.
- Get annotation and merge Excel file from ADNI.

2) Process and Format Data

- Use script in Linux to split the beta file for ease of processing, format each file in R, and calculate cell composition of the samples using the library cellDeconMeth.

3) Format and Evaluate Information Excel File

- Combine annotation and merge file using a script that matches RID # and sample examination date, which helps identify each sample's statuses, as repeat samples collected from same patients in order to record disease progression.

- Find and put together 4 different groups of samples based on status:
    - 1) ND → ND (in 2 years)
    - 2) ND → MCI (in 2 years)
    - 3) MCI → MCI (in 2 years)
    - 4) MCI → AD (in 2 years)

4) Discover Significant CpG Sites

- Find DNA methylation values present in the beta files of each sample in every status group identified in the previous step.
- Use the R library matrixTests to perform t_test: (ND → ND vs. ND → MCI) and (MCI → MCI vs. MCI → AD).
- Create a list of sites that have a p-value under 1E-5 for ND → ND vs. ND → MCI and MCI → MCI vs. MCI → AD and find/record the methylation values of these specific sites.

5) Create Machine Learning Models

- Using the same 4 machine learning algorithms as the static models, create 2 models with one having data from (ND → ND vs. ND → MCI) and the other from (MCI → MCI vs. MCI → AD) in Jupyter Notebook.
    - 1) List of 79 sites (1E-5: ND → ND vs. ND → MCI)
    - 2) List of 80 sites (1E-5: MCI → MCI vs. MCI → AD)
- Create Area Under the Curve graphs for each model.

DAVID Functional Analysis:

1) Find the gene names of the Top 20 Sites from ND to MCI Conversion Model
    - Use an Illumia site information file along with the match function to identify the gene names associated with the top 20 selected sites used in the ND to MCI Conversion Model.
2) Perform functional enrichment analysis of the genes
    - In the DAVID program, click the Functional Annotation icon.
    - Paste the gene names, select OFFICIAL_GENE_SYMBOL and specify the species as Homo sapiens (Human), check the “Gene List” box, and then click “Submit List”.
    - Save Annotation Summary Results, including UP_KEYWORDS and UP_SEQ_FEATURE in “Functional_Categories” and KEGG_PATHWAY in “Pathways”.

Data and Discussion/Analysis

Data:

Significant CpG Sites (GSE156984 + GSE153712):

In order to find significant CpG sites, t_test in Excel for each dataset was performed, which would provide a probability (p) value to a specific site. The lower the value, the more significant. *A general baseline of p-value < 1E-5* was used to determine whether a site was important or not.

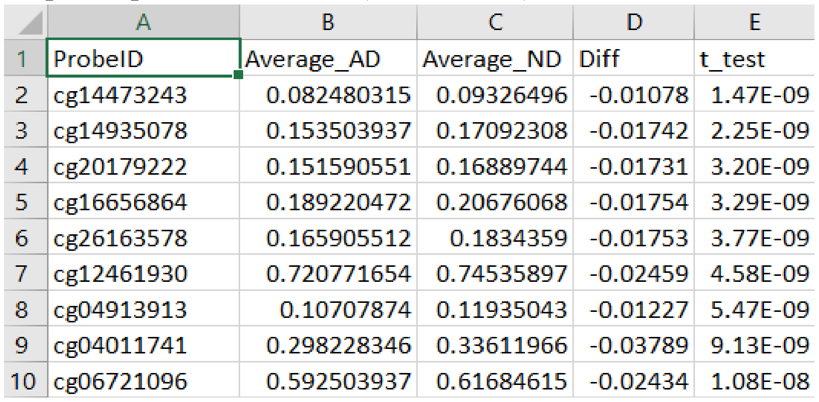
For the GSE156984, since there are only two statuses, No Disease and Alzheimer's disease, the lower the p-value was, the more significant the site was in regards to determining whether a sample should be classified as "No Disease" or "Alzheimer's disease".

On the other hand, since GSE153712 had three general statutes, No Disease, Mild Cognitive Impairment, and Alzheimer's disease, t_test was performed for all possible combinations including: AD vs. ND, AD vs. MCI, and MCI vs. ND. The significance of the sites depends on which status combination the t_test was based off of.

<u>Significant CpG Sites (GSE156984: p-value < 1E-5):</u>

- **1114 CpG sites**

Figure 11: *Top 10 CpG Sites in Excel (GSE156984)*

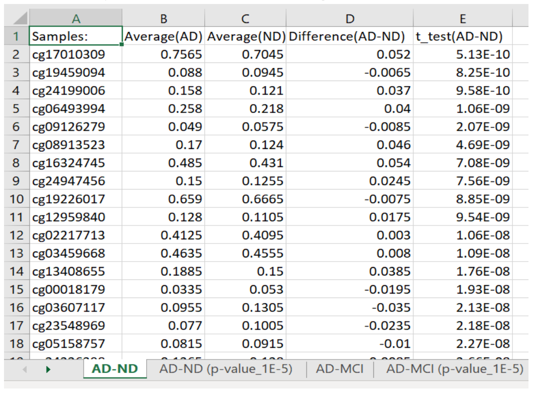


	A	B	C	D	E
1	ProbeID	Average_AD	Average_ND	Diff	t_test
2	cg14473243	0.082480315	0.09326496	-0.01078	1.47E-09
3	cg14935078	0.153503937	0.17092308	-0.01742	2.25E-09
4	cg20179222	0.151590551	0.16889744	-0.01731	3.20E-09
5	cg16656864	0.189220472	0.20676068	-0.01754	3.29E-09
6	cg26163578	0.165905512	0.1834359	-0.01753	3.77E-09
7	cg12461930	0.720771654	0.74535897	-0.02459	4.58E-09
8	cg04913913	0.10707874	0.11935043	-0.01227	5.47E-09
9	cg04011741	0.298228346	0.33611966	-0.03789	9.13E-09
10	cg06721096	0.592503937	0.61684615	-0.02434	1.08E-08

<u>Significant CpG Sites (GSE153712: p-value < 1E-5):</u>

- **AD vs. ND: 3034 CpG sites**
- **AD vs. MCI: 22 CpG sites**
- **MCI vs. ND: 248 CpG sites**

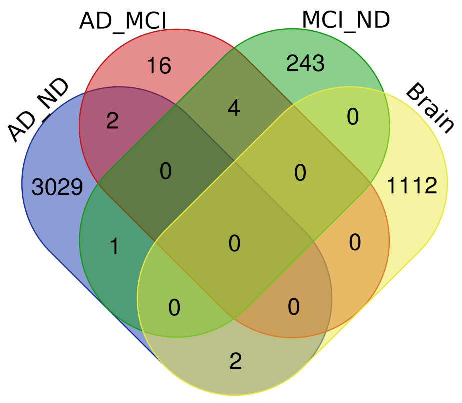
Figure 12: *GSE153712 Methylation Value Organization Sheet*



	A	B	C	D	E
1	Samples:	Average(AD)	Average(ND)	Difference(AD-ND)	t_test(AD-ND)
2	cg17010309	0.7565	0.7045	0.052	5.13E-10
3	cg19459094	0.088	0.0945	-0.0065	8.25E-10
4	cg24199006	0.158	0.121	0.037	9.58E-10
5	cg06493994	0.258	0.218	0.04	1.06E-09
6	cg09126279	0.049	0.0575	-0.0085	2.07E-09
7	cg08913523	0.17	0.124	0.046	4.69E-09
8	cg16324745	0.485	0.431	0.054	7.08E-09
9	cg24947456	0.15	0.1255	0.0245	7.56E-09
10	cg19226017	0.659	0.6665	-0.0075	8.85E-09
11	cg12959840	0.128	0.1105	0.0175	9.54E-09
12	cg02217713	0.4125	0.4095	0.003	1.06E-08
13	cg03459668	0.4635	0.4555	0.008	1.09E-08
14	cg13408655	0.1885	0.15	0.0385	1.76E-08
15	cg00018179	0.0335	0.053	-0.0195	1.93E-08
16	cg03607117	0.0955	0.1305	-0.035	2.13E-08
17	cg23548969	0.077	0.1005	-0.0235	2.18E-08
18	cg05158757	0.0815	0.0915	-0.01	2.27E-08

<u>Similarities between GSE156984 + GSE153712 sites (p < 1E-5):</u>

Figure 13: *Venn Diagram Comparison between GSE156984 + GSE153712*



<u>AD-MCI + AD-ND = cg07347869, cg05234135</u>
- AD-MCI (cg07347869) = 6.28E-06
- AD-ND (cg07347869) = 7.19E-05
- AD-MCI (cg05234135) = 1.10878E-07
- AD-ND (cg05234135) = 6.97E-05

<u>AD-ND + MCI-ND = cg04876500</u>
- AD-ND (cg04876500) = 9.5E-05
- MCI-ND (cg04876500) = 4.67E-05

<u>AD-MCI + MCI-ND = cg14706655, cg17422516, cg09234764, cg09044631</u>
- AD-MCI (cg14706655) = 9.02E-05
- MCI-ND (cg14706655) = 1.28E-05
- AD-MCI (cg17422516) = 7.82E-06
- MCI-ND (cg17422516) = 8.18E-05
- AD-MCI (cg09234764) = 7.58E-05
- MCI-ND (cg09234764) = 1.46E-05
- AD-MCI (cg09044631) = 1.01E-05
- MCI-ND (cg09044631) = 5.71E-05

<u>AD-ND + Brain: cg09559780, cg06532212</u>
- AD-ND (cg09559780) = 1.62E-05
- Brain (cg09559780) = 1.95E-05
- AD-ND (cg06532212) = 9.12E-05
- Brain (cg06532212) = 6.09E-05

6 Select Sites used for Static Model #1

The 9 sites found from the comparison between GSE156984 and GSE153712 along with the top 10 sites from MCI-ND (GSE153712) and another site found to be also significant across research papers including "Harnessing peripheral DNA methylation differences in the Alzheimer's Disease Neuroimaging Initiative (ADNI) to reveal novel biomarkers of disease" from Clin Epigenetic were put together and evaluated. A list of 6 sites was selected based on positive and negative DNA methylation correlation. That is, the average methylation values of the probes must either increase from ND → MCI and AD or decrease.

Figure 14: *20 Important CpG Sites from GSE156984 and GSE153712 (6 Selected)*

AD-MCI + AD-ND = cg07347869, cg05234135

AD-ND + MCI-ND = cg04876500

AD-MCI + MCI-ND = cg14706655, cg17422516, cg09234764, cg09044631

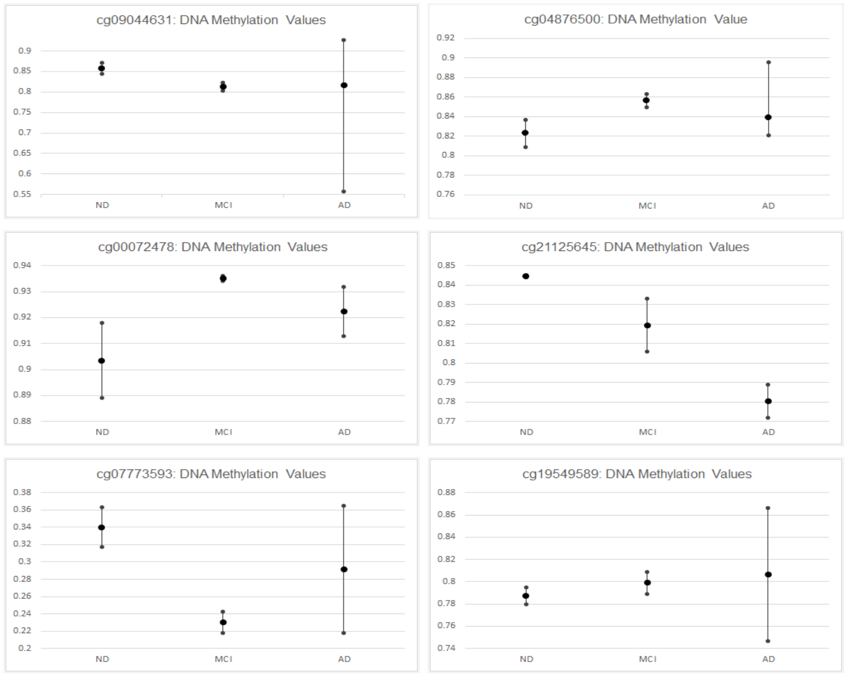
AD-ND + Brain: cg09559780, cg06532212

Another Probe (Featured in other studies): cg07773593

MCI-ND (Top 10 Sites)

- cg07241675, cg00072478, cg21125645, cg22471641, cg20630239, cg10948751, cg03700990, cg01747278, cg00862028, cg19549589

Figure 15: *Average-Min-Max graphs of the 6 Selected Sites*



Significant CpG Sites (ADNI DNA Methylation Dataset):

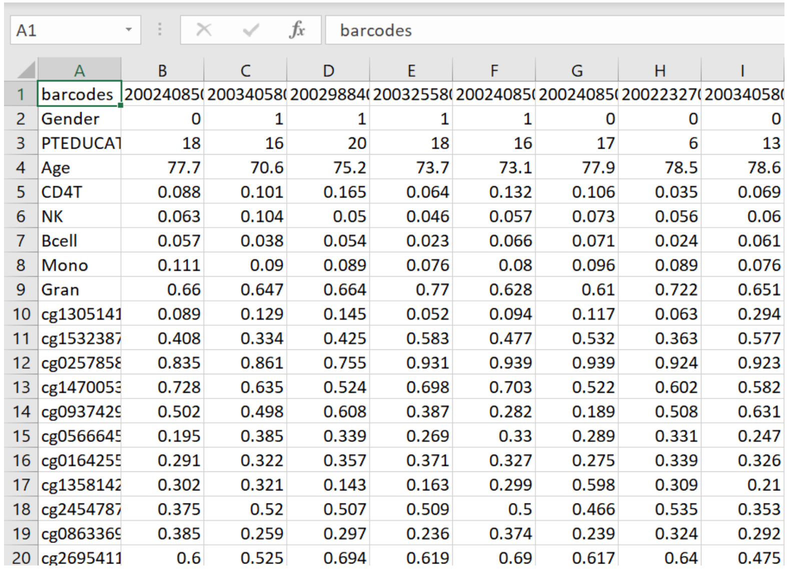
For the ADNI DNA methylation dataset, two lists of significant sites with a p-value < 1E-5 were selected. These were made using t_test in Linux (matrixTests) based on two different status groups. The first one involves ND → ND vs. ND → MCI, which in other words means samples whose disease status stayed "No Disease" in 2 years vs. samples whose disease status changed from "No Disease" to "Mild Cognitive Impairment" in 2 years. On the other hand, the second group involves MCI → MCI vs. MCI → AD, which follows the same general premise but with different status combinations.

This was done in order to select sites that were significant in terms of whether a sample would develop "Mild Cognitive Impairment"/"Alzheimer's Disease" or not, which were then used to construct their respective machine learning models.

<u>Significant Sites (ND vs. ND → ND vs. MCI: p-value < 1E-5):</u>

- **79 CpG sites**

Figure 16: *Top 10 CpG Sites in Excel including Cell Composition, Age, Gender, and PTEDUCATION Score (ND → ND vs. ND → MCI)*

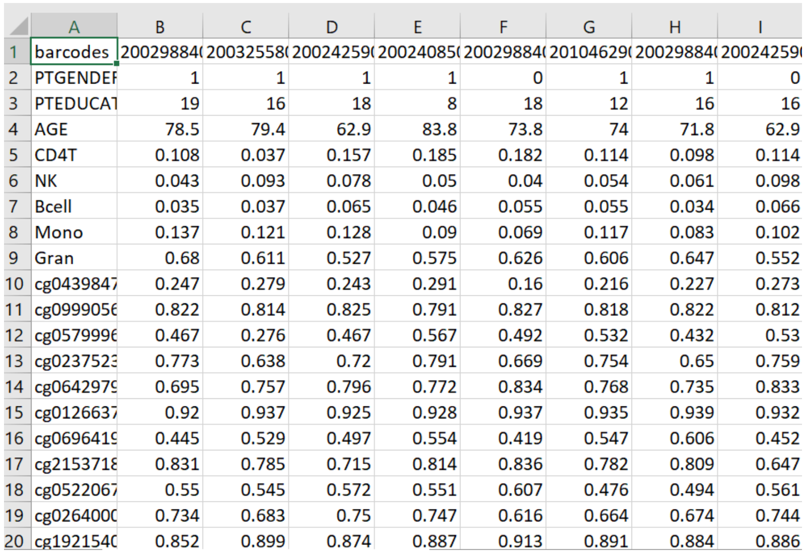

A1 | barcodes

	A	B	C	D	E	F	G	H	I
1	barcodes	20024085(	20034058(	20029884(	20032558(	20024085(	20024085(	20022327(	20034058(
2	Gender	0	1	1	1	1	0	0	0
3	PTEDUCAT	18	16	20	18	16	17	6	13
4	Age	77.7	70.6	75.2	73.7	73.1	77.9	78.5	78.6
5	CD4T	0.088	0.101	0.165	0.064	0.132	0.106	0.035	0.069
6	NK	0.063	0.104	0.05	0.046	0.057	0.073	0.056	0.06
7	Bcell	0.057	0.038	0.054	0.023	0.066	0.071	0.024	0.061
8	Mono	0.111	0.09	0.089	0.076	0.08	0.096	0.089	0.076
9	Gran	0.66	0.647	0.664	0.77	0.628	0.61	0.722	0.651
10	cg1305141	0.089	0.129	0.145	0.052	0.094	0.117	0.063	0.294
11	cg1532387	0.408	0.334	0.425	0.583	0.477	0.532	0.363	0.577
12	cg0257858	0.835	0.861	0.755	0.931	0.939	0.939	0.924	0.923
13	cg1470053	0.728	0.635	0.524	0.698	0.703	0.522	0.602	0.582
14	cg0937429	0.502	0.498	0.608	0.387	0.282	0.189	0.508	0.631
15	cg0566645	0.195	0.385	0.339	0.269	0.33	0.289	0.331	0.247
16	cg0164255	0.291	0.322	0.357	0.371	0.327	0.275	0.339	0.326
17	cg1358142	0.302	0.321	0.143	0.163	0.299	0.598	0.309	0.21
18	cg2454787	0.375	0.52	0.507	0.509	0.5	0.466	0.535	0.353
19	cg0863369	0.385	0.259	0.297	0.236	0.374	0.239	0.324	0.292
20	cg2695411	0.6	0.525	0.694	0.619	0.69	0.617	0.64	0.475

<u>Significant Sites (MCI → MCI vs. MCI → AD: p-value <1E-5)</u>

- **80 CpG sites**

Figure 17: *Top 10 CpG Sites in Excel including Cell Composition, Age, Gender, and PTEDUCATION Score (MCI → MCI vs. MCI → AD)*



	A	B	C	D	E	F	G	H	I
1	barcodes	20029884(	20032558(	20024259(	20024085(	20029884(	20104629(	20029884(	20024259(
2	PTGENDEF	1	1	1	1	0	1	1	0
3	PTEDUCAT	19	16	18	8	18	12	16	16
4	AGE	78.5	79.4	62.9	83.8	73.8	74	71.8	62.9
5	CD4T	0.108	0.037	0.157	0.185	0.182	0.114	0.098	0.114
6	NK	0.043	0.093	0.078	0.05	0.04	0.054	0.061	0.098
7	Bcell	0.035	0.037	0.065	0.046	0.055	0.055	0.034	0.066
8	Mono	0.137	0.121	0.128	0.09	0.069	0.117	0.083	0.102
9	Gran	0.68	0.611	0.527	0.575	0.626	0.606	0.647	0.552
10	cg0439847	0.247	0.279	0.243	0.291	0.16	0.216	0.227	0.273
11	cg0999056	0.822	0.814	0.825	0.791	0.827	0.818	0.822	0.812
12	cg0579996	0.467	0.276	0.467	0.567	0.492	0.532	0.432	0.53
13	cg0237523	0.773	0.638	0.72	0.791	0.669	0.754	0.65	0.759
14	cg0642979	0.695	0.757	0.796	0.772	0.834	0.768	0.735	0.833
15	cg0126637	0.92	0.937	0.925	0.928	0.937	0.935	0.939	0.932
16	cg0696419	0.445	0.529	0.497	0.554	0.419	0.547	0.606	0.452
17	cg2153718	0.831	0.785	0.715	0.814	0.836	0.782	0.809	0.647
18	cg0522067	0.55	0.545	0.572	0.551	0.607	0.476	0.494	0.561
19	cg0264000	0.734	0.683	0.75	0.747	0.616	0.664	0.674	0.744
20	cg1921540	0.852	0.899	0.874	0.887	0.913	0.891	0.884	0.886

Static Models Accuracy Results:

Once again, three models which each used a list of different significant CpG sites were created for the purpose of determining a sample's status as either "No Disease" or "Mild Cognitive Impairment". As such, only methylation data from MCI & ND samples from the GSE153712 were used.

1. <u>6 Selected Sites</u>

Table 1: 6 Selected Sites

Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	95.7%	86.6%	76.1%
LogisticRegression	67.3%	69.5%	77.5%
RandomForest	94.3%	82.3%	73.2%
GradientBoosting	94.9%	83.5%	76.8%

2. <u>Sites with p-value < 1E-6 (594: GSE153712)</u>

Table 2: Sites with p-value < 1E-6 (GSE153712)			
Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	92.0%	88.4%	65.5%
LogisticRegression	95.9%	83.5%	71.8%
RandomForest	96.9%	81.1%	70.4%
GradientBoosting	88.5%	77.4%	69.7%

3. <u>Collection of 5 models using varying amounts of sites (p-value < 1E-6: GSE153712) found using lasso</u>

By using the lasso algorithm in Jupyter Notebook that determines which sites are most significant, 5 lists of different amount of sites were created based on a pool of sites with p-value < 1E-6 (594: GSE153712), including 10, 45, 100, 200, and 300 sites.

Table 3: Lasso_10 Sites			
Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	98.6%	84.8%	76.1%
LogisticRegression	67.5%	58.5%	56.3%
RandomForest	93.3%	72.6%	66.9%
GradientBoosting	95.7%	73.8%	62.0%

Table 4: Lasso_45 Sites			
Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	96.7%	89.0%	73.9%
LogisticRegression	88.1%	76.8%	66.9%
RandomForest	93.0%	78.0%	64.1%
GradientBoosting	93.9%	81.1%	62.0%

Table 5: Lasso_100 Sites			
Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	94.3%	89.6%	73.9%
LogisticRegression	97.1%	87.8%	71.8%
RandomForest	91.8%	78.0%	64.1%
GradientBoosting	96.3%	80.5%	69.0%

Table 6: Lasso_200 Sites			
Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	90.6%	84.1%	78.2%
LogisticRegression	96.1%	86.0%	70.4%
RandomForest	95.5%	80.5%	73.9%
GradientBoosting	91.8%	81.7%	70.4%

Table 7: Lasso_300 Sites			
Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	**93.5%**	**90.9%**	**80.3%**
LogisticRegression	96.1%	83.5%	70.4%
RandomForest	95.1%	82.3%	70.4%
GradientBoosting	95.5%	81.7%	66.2%

Progressive Models Accuracy Results:

Two progressive models were selected that could predict a sample's disease status in 2 years. This depends on their current status, as the ND to MCI conversion model predicts whether a person's status will stay "No Disease" or change to "Mild Cognitive Impairment". On the other hand, the MCI to AD conversion model predicts whether a person's status will stay "Mild Cognitive Impairment" or change to "Alzheimer's Disease".

Table 8: ND → ND vs. ND → MCI Model

Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	**98.7%**	**96.3%**	**86.1%**
LogisticRegression	98.7%	96.3%	83.3%
RandomForest	97.5%	81.5%	77.8%
GradientBoosting	94.9%	81.5%	66.7%

Table 9: MCI → MCI vs. MCI → AD Model

Model (Algorithm):	Training Accuracy:	Validation Accuracy:	Testing Accuracy:
SVC	**97.6%**	**88.1%**	**90.9%**
LogisticRegression	93.5%	88.1%	90.0%
RandomForest	97.6%	76.2%	74.8%
GradientBoosting	97.6%	69.0%	60.0%

DAVID Functional Analysis:

Figure 18: *Top 20 Sites for the ND to MCI Conversion (DAVID Analysis)*

Probe ID	Methylation MCI in 2 yrs	Methylation ND in 2 yrs	Difference	P value	Chromo-some	Position	Gene	Biological Pathway	Disease Class
cg13051418	0.143	0.084	0.059	2.25E-07	12	115139149			
cg15323871	0.420	0.350	0.070	5.29E-07	14	60043906	C14orf38		
cg02578584	0.866	0.913	-0.047	1.43E-06	8	8325700			
cg14700531	0.598	0.540	0.059	1.68E-06	8	733412			
cg09374293	0.448	0.558	-0.110	1.92E-06	21	48081242	PRMT2	Transcription regulation	**Neurological**
cg05666456	0.358	0.273	0.085	2.57E-06	4	170214340			
cg01642550	0.336	0.296	0.040	5.6E-06	16	89098327			
cg13581422	0.254	0.332	-0.078	5.97E-06	3	130236522			
cg24547873	0.488	0.541	-0.053	6.32E-06	1	17086558	MST1P9	Macrophage stimulation	
cg26954114	0.573	0.479	0.094	7.55E-06	15	96838120			
cg17024257	0.706	0.653	0.053	7.94E-06	3	171528758	PLD1	Signal transduction	Metabolic
cg20737388	0.533	0.648	-0.115	8E-06	11	73668626	DNAJB13	HSP40 co-chaperone	**Neurological**
cg03812172	0.531	0.707	-0.176	1E-05	7	44184403	GCK	Glycogen biosynthesis	**Neurological.** Immune
cg14168690	0.317	0.219	0.098	1.1E-05	21	32819053	TIAM1	Protein localization	**Neurological.** Metabloic
cg13455439	0.399	0.487	-0.088	1.53E-05	11	69934128	ANO1	Chloride channel	
cg15865243	0.703	0.632	0.072	1.62E-05	12	34496342			
cg23692114	0.447	0.413	0.034	1.69E-05	2	75154873	LINC01291		
cg13649415	0.715	0.755	-0.039	1.74E-05	3	46621737	TDGF1	Cell differentiation	Metabolic
cg00240732	0.432	0.375	0.057	1.76E-05	7	70923396	WBSCR17	Membrane trafficking	**Neurological.** Immune
cg08707819	0.351	0.311	0.039	1.83E-05	14	103059391	RCOR1	Neural cell differentiation	

Discussion/Analysis

The results obtained from this experiment supported both the background research and my hypothesis of whether DNA methylation can be used as a biomarker to predict Alzheimer's disease. In the study "DNA methylation alterations in Alzheimer's disease" featured in NCBI, it states that aberrant methylation in peripheral blood is correlated with AD disease status (Yokoyama, Rutledge, Medici 2017). This is supported here by the finding of hundreds of significant CpG sites across the GSE156984, GSE153712, and ADNI dataset through the use of t_test to find probability (p) values. 4577 total sites across all datasets were found to have a p-value < 1E-5, which demonstrates a clear correlation between these specific methylated sites and the development/existence of Alzheimer's disease. Analysis of the top 20 sites from the ND to MCI conversion model shown in Figure 18 found that the genes (PRMT2, DNAJB13, GCK, TIAM1, WBSCR17) of some of the sites have a connection with neurological disease, strengthening the general notion that DNA methylation has an association with Alzheimer's.

The creation of multiple successful machine learning models that utilized these various sites also served to support my hypothesis. Tables 7-9 display the results from the Lasso 300 static model, the ND to MCI conversion model, and the MCI to AD conversion model, with all three having a testing accuracy of at least 80%. As such, rather than a regular 50/50 percent chance of guessing a patient's status, these models can accurately predict both their current and future status at least 80% of the time, which is a large improvement. This provides further supporting evidence that DNA methylation is not only associated with Alzheimer's, but that it can also be used to more reliably predict the disease itself.

Conclusion

Alzheimer's disease is a progressive form of dementia that damages memory and other cognitive functions. Despite the large and growing threat it poses to our collective society, there is still no form of early diagnosis, which is crucial to provide proactive medical treatment to slow the disease's progression. To address this issue, this experiment was conducted to test whether DNA methylation can be used as a biomarker to predict Alzheimer's disease and create a method of reliably detecting pre-symptomatic cognitive impairment for early intervention. I discovered 4577 total significant CpG sites associated with the development and existence of Alzheimer's disease through the use of probability (p) values. These sites were then used to construct multiple successful machine learning models that could both determine a patient's current status and future status in 2 years with an accuracy of at least 80%. The apparent ability of these models to accurately determine/predict a patient's status, no matter whether it is in the present or future, serves to once again demonstrate the notion that these specific methylated sites have an innate correlation to Alzheimer's and can therefore be used as a means of predicting the disease. In conclusion, because of the experiment's findings, it can be reasonably concluded that the hypothesis regarding the feasibility of using DNA methylation to diagnose Alzheimer's disease was supported and the objective of creating a method of detecting pre-symptomatic cognitive impairment was met as well.

Literature Cited/Reference List

Alzheimer’s Association. Facts and Figures (Alzheimer’s Disease and Dementia). www.alz.org/alzheimers-dementia/facts-figures

Alzheimer's Disease Neuroimaging Initiative. *ADNI*. adni.loni.usc.edu/

Johns, Harry. Testimony of Harry Johns, President and CEO of the Alzheimer’s Association Fiscal Year 2014 Appropriations for Alzheimer’s-Related Activities at the U.S.Department of Health and Human Services.” 13 Mar. 2013. https://www.alz.org/documents/national/submitted-testimony-050113.pdf

Li QS, Sun Y, Wang T. (2020) Epigenome-wide association study of Alzheimer's disease replicates 22 differentially methylated positions and 30 differentially methylated regions. *Clin Epigenetics*. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE156984

Müller, A. C., and S. Guido. (2016). Introduction to Machine Learning with Python: a Guide for Data Scientists. *O’reilly Et Associates In.* Nabais, MF., Laws, SM., Wray, NR. *et al.* (2020). Meta-analysis of genome-wide DNA methylation identifies shared associations across neurodegenerative disorders. *NCBI Gene Expression Omnibus.* https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE153712

Nabais, MF., Laws, SM., Wray, NR. *et al.* (2020). Meta-analysis of genome- wide DNA methylation identifies shared associations across neurodegenerative disorders. *NCBI Gene Expression Omnibus.* https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE153712

Vasanthakumar, A., Davis, J.W., Idler, K. *et al.* (2020). Harnessing peripheral DNA methylation differences in the Alzheimer’s Disease Neuroimaging Initiative (ADNI) to reveal novel biomarkers of disease. *Clin Epigenet* 12, 84. https://clinicalepigeneticsjournal.biomedcentral.com/articles/10.1186/s13148-020-00864-y#availability-of-data-and-materials

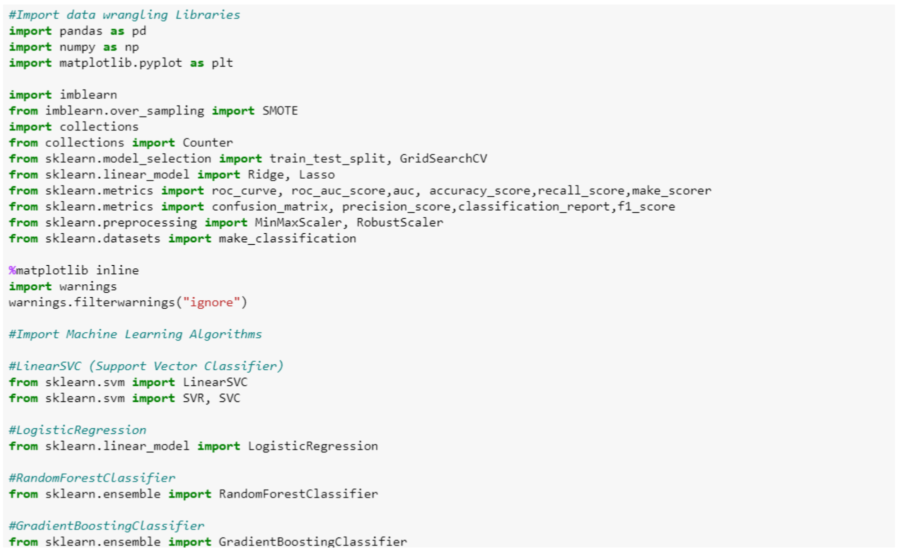
Yokoyama, A. S., Rutledge, J. C., & Medici, V. (2017). DNA methylation alterations in Alzheimer's disease. *Environmental epigenetics*, *3*(2), dvx008. https://doi.org/10.1093/eep/dvx008

Appendix A

Code for Machine Learning Models

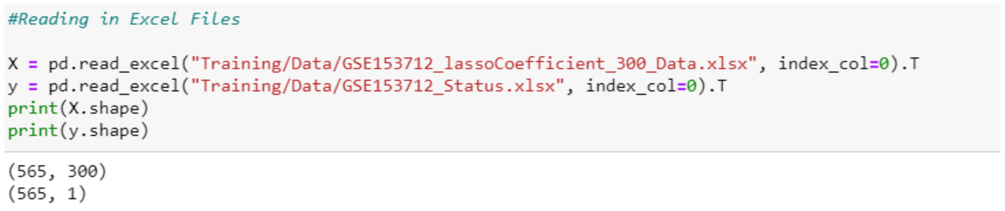
<u>Lasso_3oo (Static Model):</u>

Figure 19: *Essential Libraries and Tools (Same for All Models)*



```
#Import data wrangling Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import imblearn
from imblearn.over_sampling import SMOTE
import collections
from collections import Counter
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import roc_curve, roc_auc_score,auc, accuracy_score,recall_score,make_scorer
from sklearn.metrics import confusion_matrix, precision_score,classification_report,f1_score
from sklearn.preprocessing import MinMaxScaler, RobustScaler
from sklearn.datasets import make_classification

%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

#Import Machine Learning Algorithms

#LinearSVC (Support Vector Classifier)
from sklearn.svm import LinearSVC
from sklearn.svm import SVR, SVC

#LogisticRegression
from sklearn.linear_model import LogisticRegression

#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

#GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

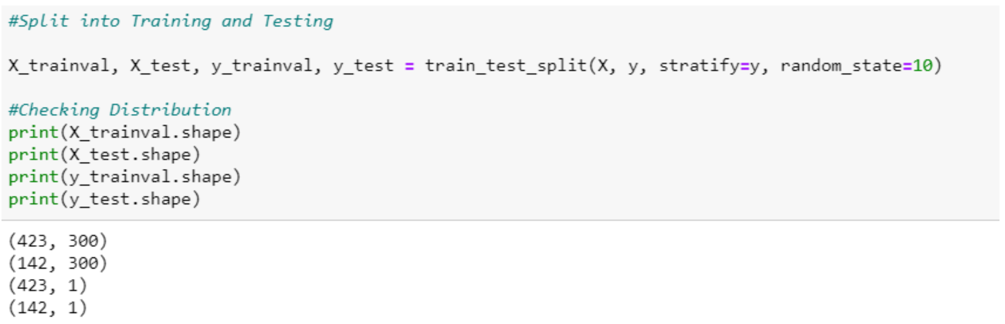
Figure 20: *Reading in Excel Files*



```
#Reading in Excel Files

X = pd.read_excel("Training/Data/GSE153712_lassoCoefficient_300_Data.xlsx", index_col=0).T
y = pd.read_excel("Training/Data/GSE153712_Status.xlsx", index_col=0).T
print(X.shape)
print(y.shape)
```

```
(565, 300)
(565, 1)
```

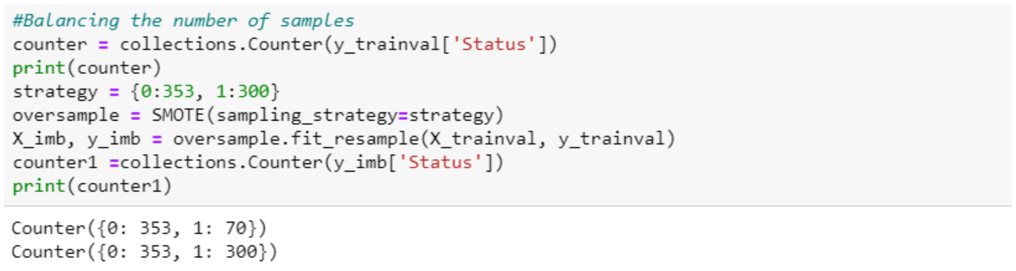
Figure 21: *Splitting into Training and Testing*



```
#Split into Training and Testing

X_trainval, X_test, y_trainval, y_test = train_test_split(X, y, stratify=y, random_state=10)

#Checking Distribution
print(X_trainval.shape)
print(X_test.shape)
print(y_trainval.shape)
print(y_test.shape)
```

```
(423, 300)
(142, 300)
(423, 1)
(142, 1)
```

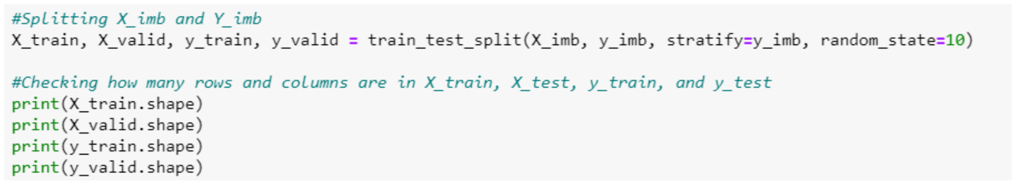
Figure 22: *Balancing the Number of Samples*



```
#Balancing the number of samples
counter = collections.Counter(y_trainval['Status'])
print(counter)
strategy = {0:353, 1:300}
oversample = SMOTE(sampling_strategy=strategy)
X_imb, y_imb = oversample.fit_resample(X_trainval, y_trainval)
counter1 =collections.Counter(y_imb['Status'])
print(counter1)
```

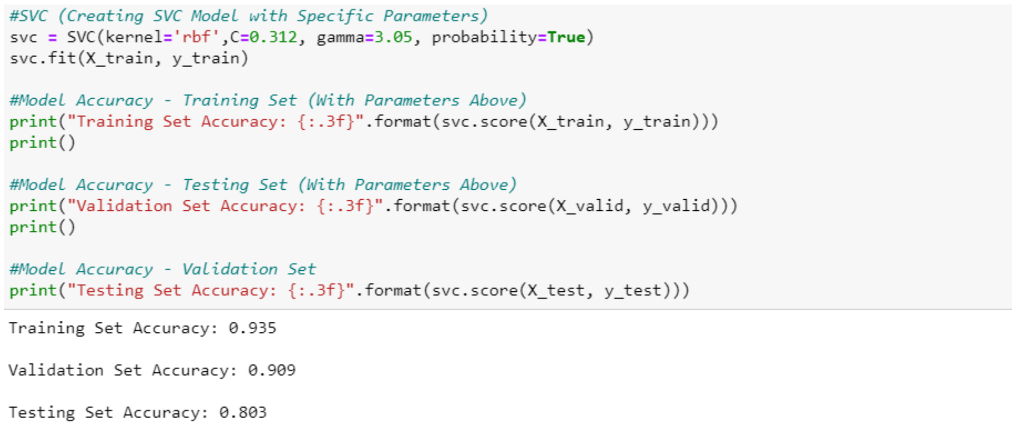
```
Counter({0: 353, 1: 70})
Counter({0: 353, 1: 300})
```

Figure 23: *Splitting X_imb and y_imb (New Balanced Variables)*



```
#Splitting X_imb and Y_imb
X_train, X_valid, y_train, y_valid = train_test_split(X_imb, y_imb, stratify=y_imb, random_state=10)

#Checking how many rows and columns are in X_train, X_test, y_train, and y_test
print(X_train.shape)
print(X_valid.shape)
print(y_train.shape)
print(y_valid.shape)
```

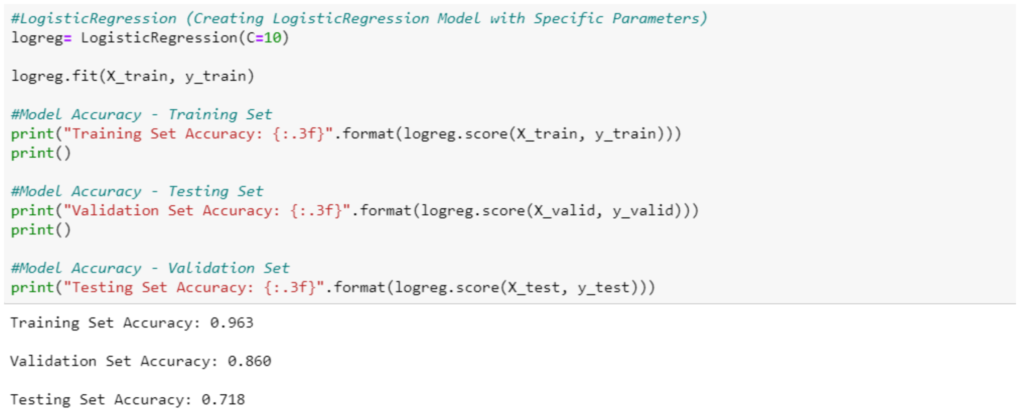
Figure 24: *SVC (Support Vector Classifier) Algorithm Results*



```
#SVC (Creating SVC Model with Specific Parameters)
svc = SVC(kernel='rbf',C=0.312, gamma=3.05, probability=True)
svc.fit(X_train, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(svc.score(X_train, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(svc.score(X_valid, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(svc.score(X_test, y_test)))
```

```
Training Set Accuracy: 0.935

Validation Set Accuracy: 0.909

Testing Set Accuracy: 0.803
```

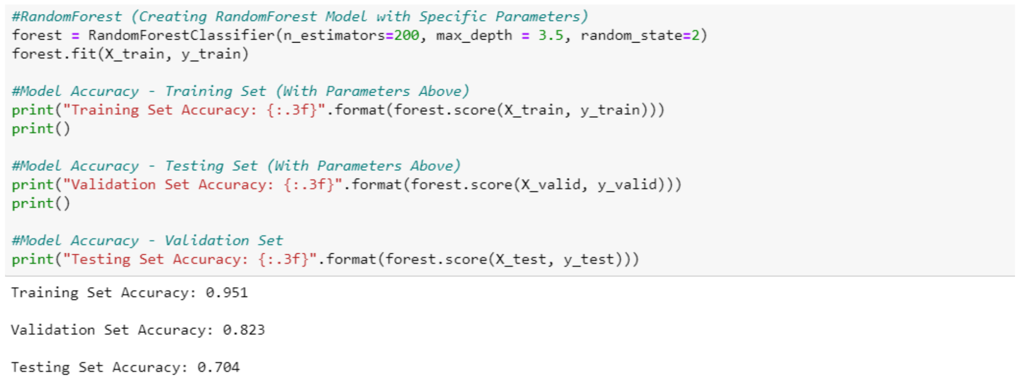
Figure 25: *LogisticRegression Algorithm Results*



```
#LogisticRegression (Creating LogisticRegression Model with Specific Parameters)
logreg= LogisticRegression(C=10)

logreg.fit(X_train, y_train)

#Model Accuracy - Training Set
print("Training Set Accuracy: {:.3f}".format(logreg.score(X_train, y_train)))
print()

#Model Accuracy - Testing Set
print("Validation Set Accuracy: {:.3f}".format(logreg.score(X_valid, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(logreg.score(X_test, y_test)))
```

```
Training Set Accuracy: 0.963

Validation Set Accuracy: 0.860

Testing Set Accuracy: 0.718
```

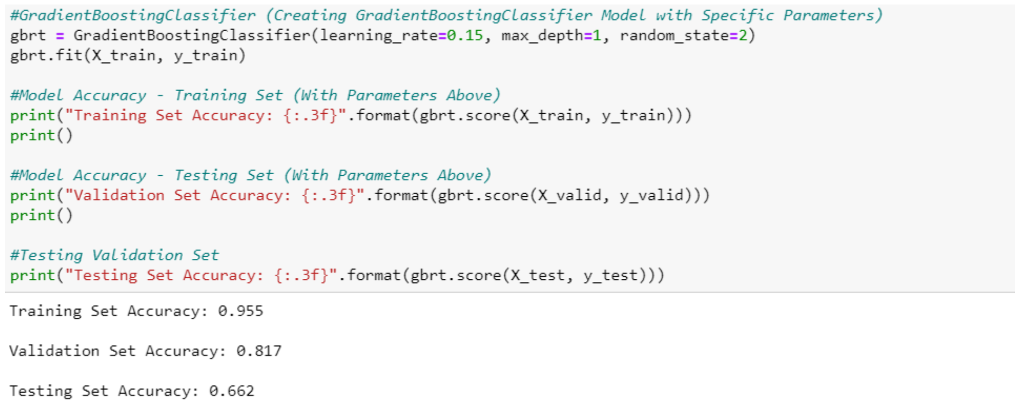
Figure 26: *RandomForest Algorithm Results*



```
#RandomForest (Creating RandomForest Model with Specific Parameters)
forest = RandomForestClassifier(n_estimators=200, max_depth = 3.5, random_state=2)
forest.fit(X_train, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(forest.score(X_train, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(forest.score(X_valid, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(forest.score(X_test, y_test)))
```

```
Training Set Accuracy: 0.951

Validation Set Accuracy: 0.823

Testing Set Accuracy: 0.704
```

Figure 27: *GradientBoostingClassifier Algorithm Results*

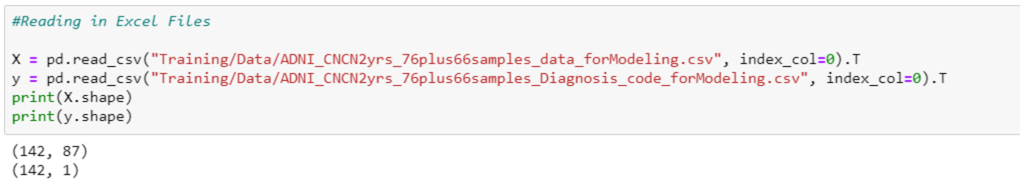


```
#GradientBoostingClassifier (Creating GradientBoostingClassifier Model with Specific Parameters)
gbrt = GradientBoostingClassifier(learning_rate=0.15, max_depth=1, random_state=2)
gbrt.fit(X_train, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(gbrt.score(X_train, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(gbrt.score(X_valid, y_valid)))
print()

#Testing Validation Set
print("Testing Set Accuracy: {:.3f}".format(gbrt.score(X_test, y_test)))
```

```
Training Set Accuracy: 0.955

Validation Set Accuracy: 0.817

Testing Set Accuracy: 0.662
```

ND → ND vs. ND → MCI (Progressive Model):

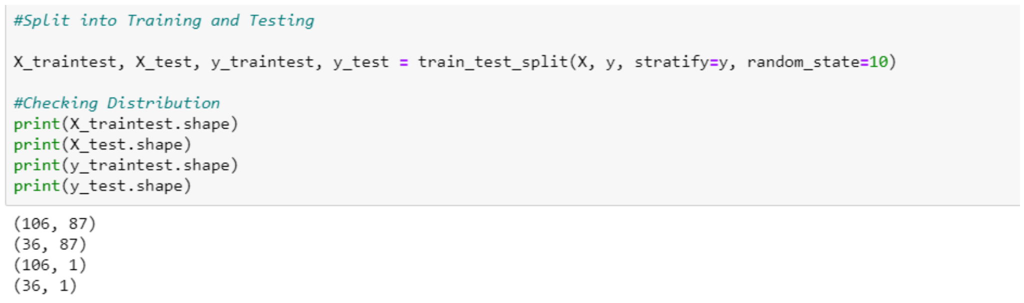
Figure 28: *Reading in Excel Files*



```
#Reading in Excel Files

X = pd.read_csv("Training/Data/ADNI_CNCN2yrs_76plus66samples_data_forModeling.csv", index_col=0).T
y = pd.read_csv("Training/Data/ADNI_CNCN2yrs_76plus66samples_Diagnosis_code_forModeling.csv", index_col=0).T
print(X.shape)
print(y.shape)
```

```
(142, 87)
(142, 1)
```

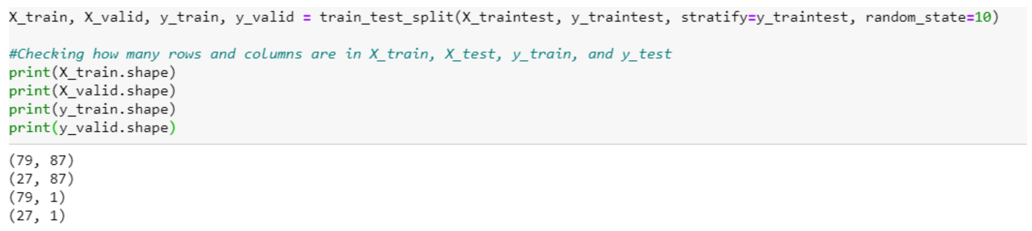
Figure 29: *Splitting into Training and Testing*



```
#Split into Training and Testing

X_traintest, X_test, y_traintest, y_test = train_test_split(X, y, stratify=y, random_state=10)

#Checking Distribution
print(X_traintest.shape)
print(X_test.shape)
print(y_traintest.shape)
print(y_test.shape)
```

```
(106, 87)
(36, 87)
(106, 1)
(36, 1)
```

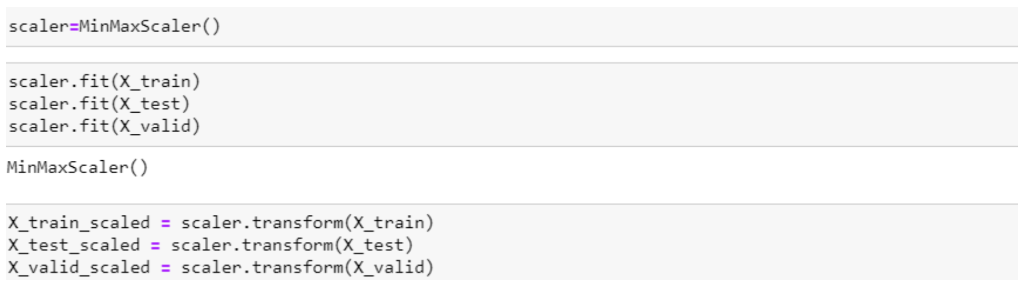
Figure 30: *Splitting X_traintest into X_train & X_valid and y_traintest into y_train & y_valid*



```
X_train, X_valid, y_train, y_valid = train_test_split(X_traintest, y_traintest, stratify=y_traintest, random_state=10)

#Checking how many rows and columns are in X_train, X_test, y_train, and y_test
print(X_train.shape)
print(X_valid.shape)
print(y_train.shape)
print(y_valid.shape)
```

```
(79, 87)
(27, 87)
(79, 1)
(27, 1)
```

Figure 31: *Scaling X_train, X_test, and X_valid (Same for MCI → MCI vs. MCI → AD Model)*



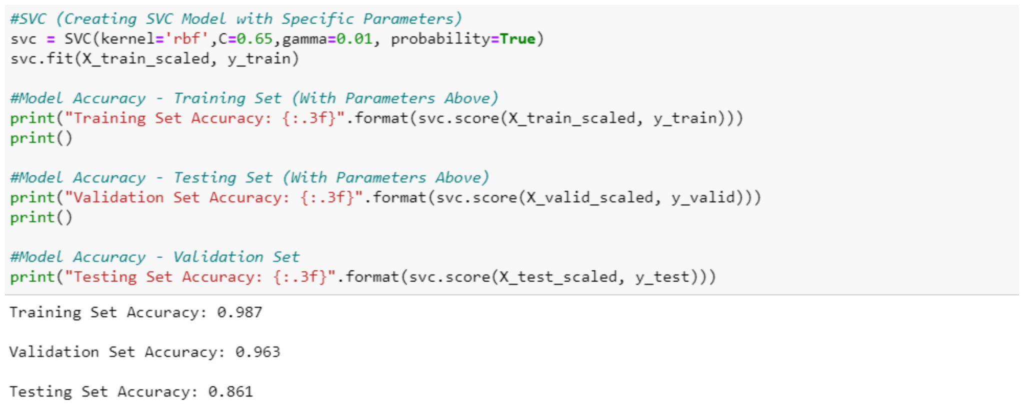
```
scaler=MinMaxScaler()
```

```
scaler.fit(X_train)
scaler.fit(X_test)
scaler.fit(X_valid)
```

```
MinMaxScaler()
```

```
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_valid_scaled = scaler.transform(X_valid)
```

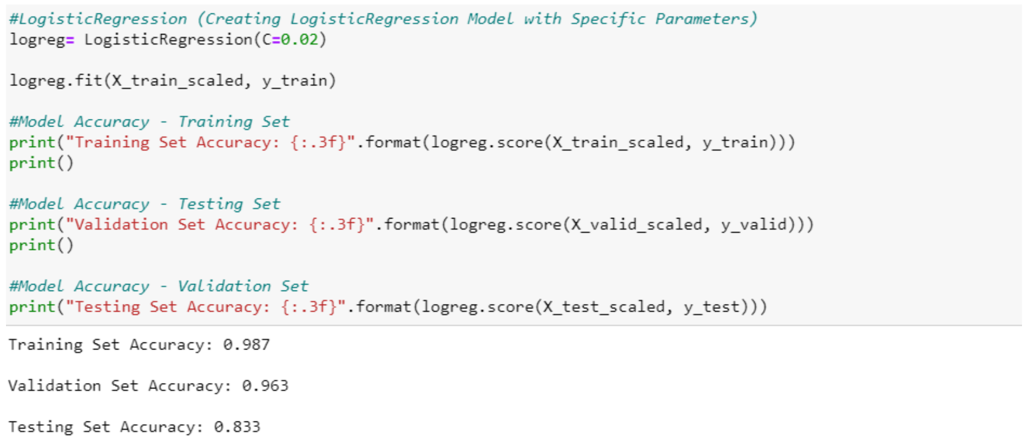
Figure 32: *SVC Algorithm Model Results*



```
#SVC (Creating SVC Model with Specific Parameters)
svc = SVC(kernel='rbf',C=0.65,gamma=0.01, probability=True)
svc.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(svc.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(svc.score(X_valid_scaled, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.987

Validation Set Accuracy: 0.963

Testing Set Accuracy: 0.861
```

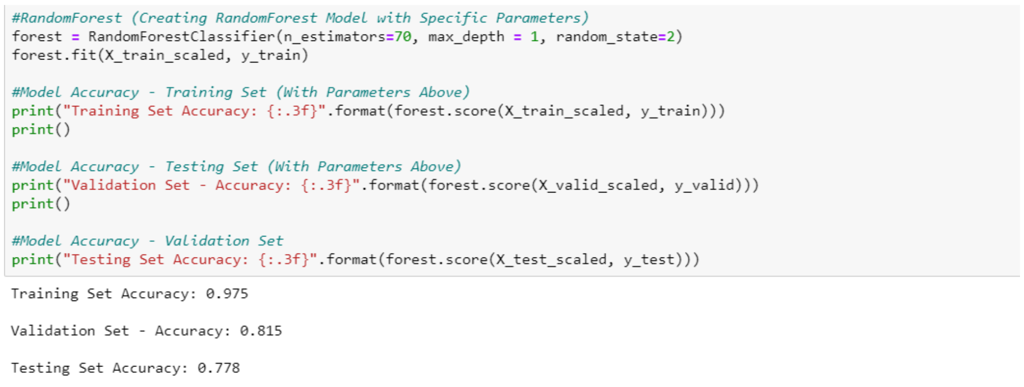
Figure 33: *LogisticRegression Algorithm Model Results*



```
#LogisticRegression (Creating LogisticRegression Model with Specific Parameters)
logreg= LogisticRegression(C=0.02)

logreg.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set
print("Training Set Accuracy: {:.3f}".format(logreg.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set
print("Validation Set Accuracy: {:.3f}".format(logreg.score(X_valid_scaled, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(logreg.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.987

Validation Set Accuracy: 0.963

Testing Set Accuracy: 0.833
```

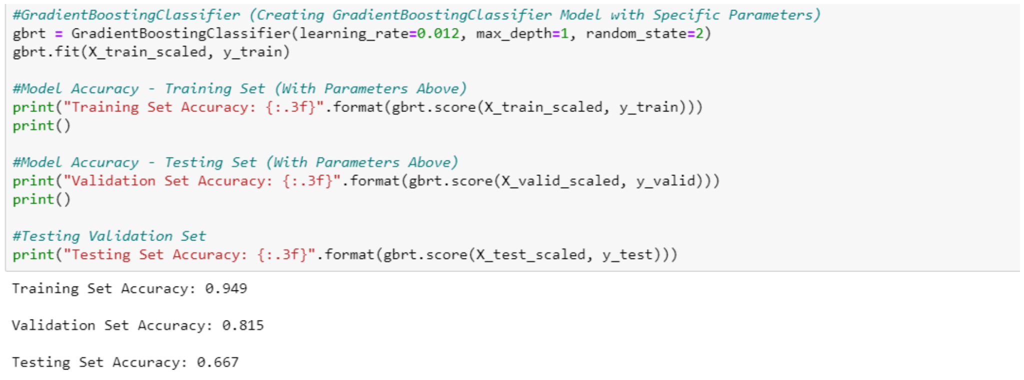
Figure 34: *RandomForest Algorithm Model Results*



```
#RandomForest (Creating RandomForest Model with Specific Parameters)
forest = RandomForestClassifier(n_estimators=70, max_depth = 1, random_state=2)
forest.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(forest.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set - Accuracy: {:.3f}".format(forest.score(X_valid_scaled, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(forest.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.975

Validation Set - Accuracy: 0.815

Testing Set Accuracy: 0.778
```

Figure 35: *GradientBoostingClassifier Algorithm Model Results*

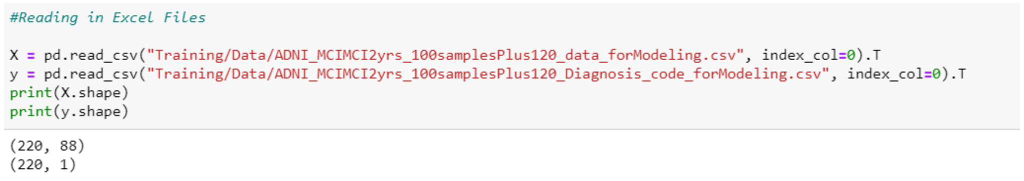


```
#GradientBoostingClassifier (Creating GradientBoostingClassifier Model with Specific Parameters)
gbrt = GradientBoostingClassifier(learning_rate=0.012, max_depth=1, random_state=2)
gbrt.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(gbrt.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(gbrt.score(X_valid_scaled, y_valid)))
print()

#Testing Validation Set
print("Testing Set Accuracy: {:.3f}".format(gbrt.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.949

Validation Set Accuracy: 0.815

Testing Set Accuracy: 0.667
```

<u>MCI → MCI vs. MCI → AD (Progressive Model):</u>

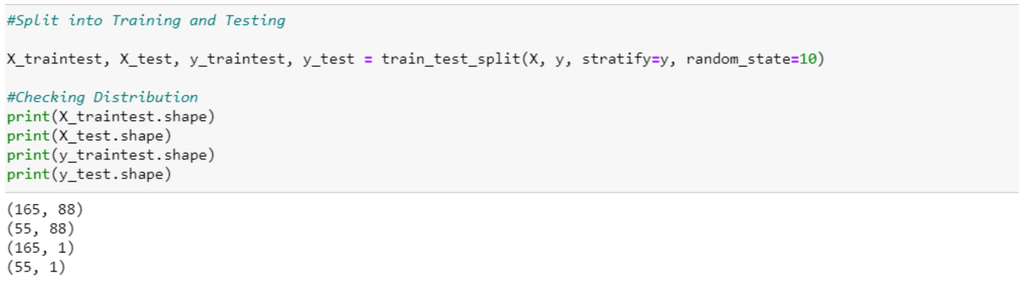
Figure 36: *Reading in Excel Files*



```
#Reading in Excel Files

X = pd.read_csv("Training/Data/ADNI_MCIMCI2yrs_100samplesPlus120_data_forModeling.csv", index_col=0).T
y = pd.read_csv("Training/Data/ADNI_MCIMCI2yrs_100samplesPlus120_Diagnosis_code_forModeling.csv", index_col=0).T
print(X.shape)
print(y.shape)
```

```
(220, 88)
(220, 1)
```

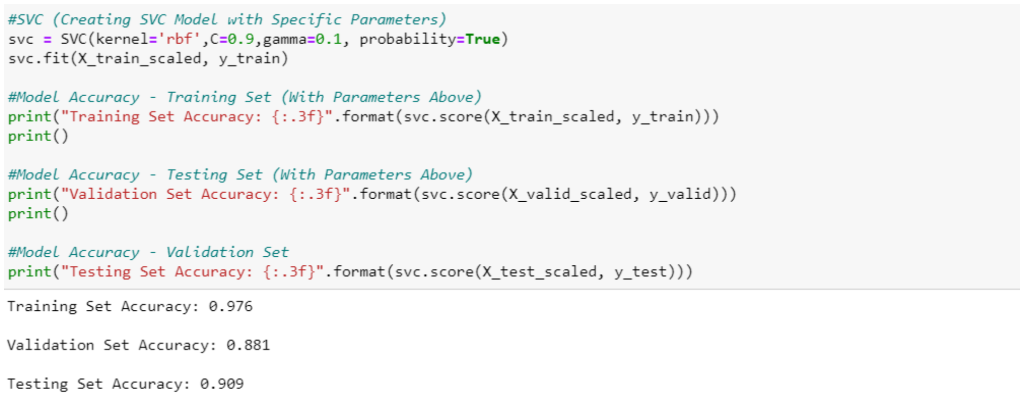
Figure 37: *Splitting into Training and Test*



```
#Split into Training and Testing

X_traintest, X_test, y_traintest, y_test = train_test_split(X, y, stratify=y, random_state=10)

#Checking Distribution
print(X_traintest.shape)
print(X_test.shape)
print(y_traintest.shape)
print(y_test.shape)
```

```
(165, 88)
(55, 88)
(165, 1)
(55, 1)
```

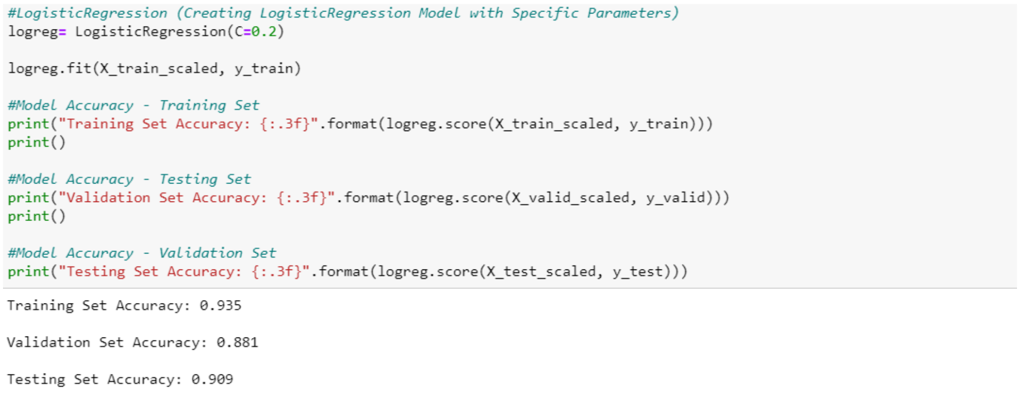
Figure 38: *SVC Algorithm Model Results*



```
#SVC (Creating SVC Model with Specific Parameters)
svc = SVC(kernel='rbf',C=0.9,gamma=0.1, probability=True)
svc.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(svc.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(svc.score(X_valid_scaled, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.976

Validation Set Accuracy: 0.881

Testing Set Accuracy: 0.909
```

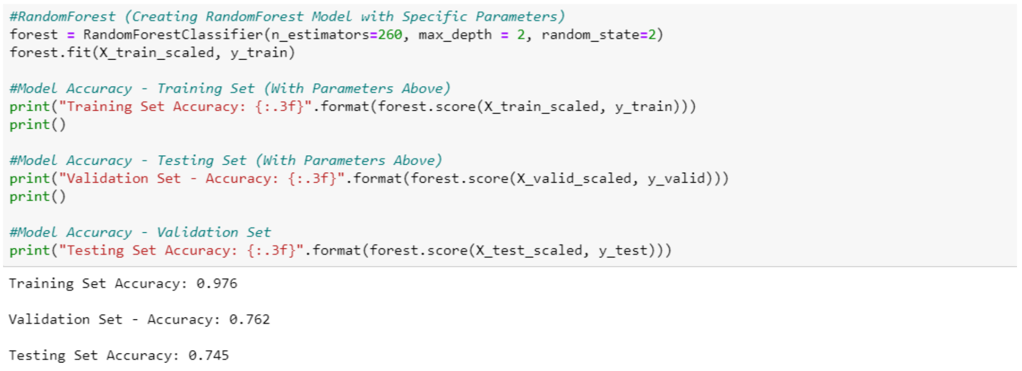
Figure 39: *LogisticRegression Algorithm Model Results*



```
#LogisticRegression (Creating LogisticRegression Model with Specific Parameters)
logreg= LogisticRegression(C=0.2)

logreg.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set
print("Training Set Accuracy: {:.3f}".format(logreg.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set
print("Validation Set Accuracy: {:.3f}".format(logreg.score(X_valid_scaled, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(logreg.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.935

Validation Set Accuracy: 0.881

Testing Set Accuracy: 0.909
```

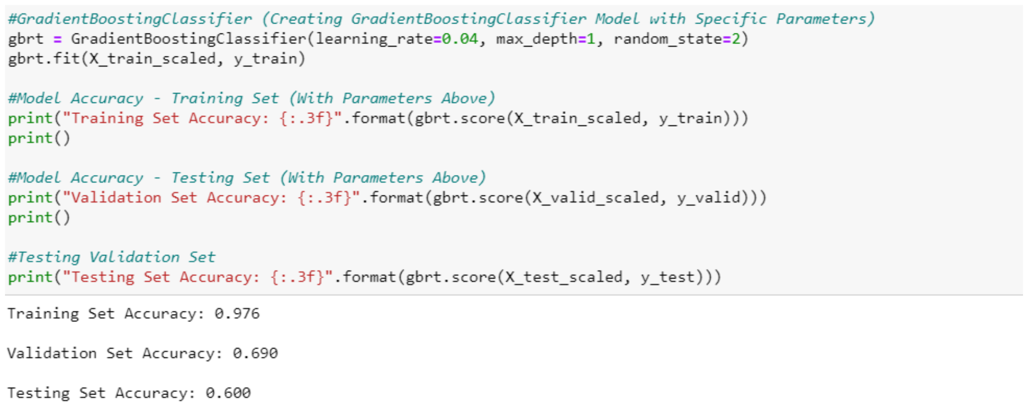
Figure 40: *RandomForest Algorithm Model Results*



```
#RandomForest (Creating RandomForest Model with Specific Parameters)
forest = RandomForestClassifier(n_estimators=260, max_depth = 2, random_state=2)
forest.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(forest.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set - Accuracy: {:.3f}".format(forest.score(X_valid_scaled, y_valid)))
print()

#Model Accuracy - Validation Set
print("Testing Set Accuracy: {:.3f}".format(forest.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.976

Validation Set - Accuracy: 0.762

Testing Set Accuracy: 0.745
```

Figure 41: GradientBoostingClassifier Algorithm Model Results



```
#GradientBoostingClassifier (Creating GradientBoostingClassifier Model with Specific Parameters)
gbrt = GradientBoostingClassifier(learning_rate=0.04, max_depth=1, random_state=2)
gbrt.fit(X_train_scaled, y_train)

#Model Accuracy - Training Set (With Parameters Above)
print("Training Set Accuracy: {:.3f}".format(gbrt.score(X_train_scaled, y_train)))
print()

#Model Accuracy - Testing Set (With Parameters Above)
print("Validation Set Accuracy: {:.3f}".format(gbrt.score(X_valid_scaled, y_valid)))
print()

#Testing Validation Set
print("Testing Set Accuracy: {:.3f}".format(gbrt.score(X_test_scaled, y_test)))
```

```
Training Set Accuracy: 0.976

Validation Set Accuracy: 0.690

Testing Set Accuracy: 0.600
```

Appendix B

Helpful Resources

1) ADNI (Alzheimer's Disease Neuroimaging Initiative)

2) "A Practical Guide to Linux Commands, Editors and Shell Programming" by Mark G. Sobell

3) DAVID (The Database for Annotation, Visulatzation, and Integrated Discovery) Bioinformatics Resources 6.8

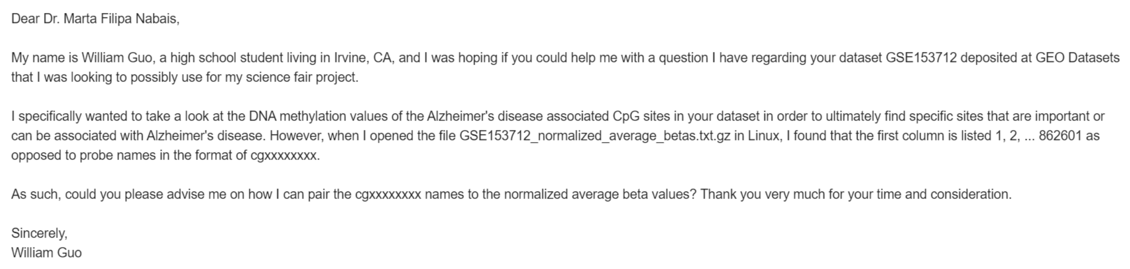
4) "Introduction to Machine Learning with Python: A Guide for Data Scientists" by Andreas C. Müller & Sarah Guido

5) Pubmed - National Center for Biotechnology Information

Appendix C

Communication with Dr. Marta Nabais

Figure 42: *Contacting Dr. Nabais regarding probe names missing in the GSE153712 dataset*



Dear Dr. Marta Filipa Nabais,

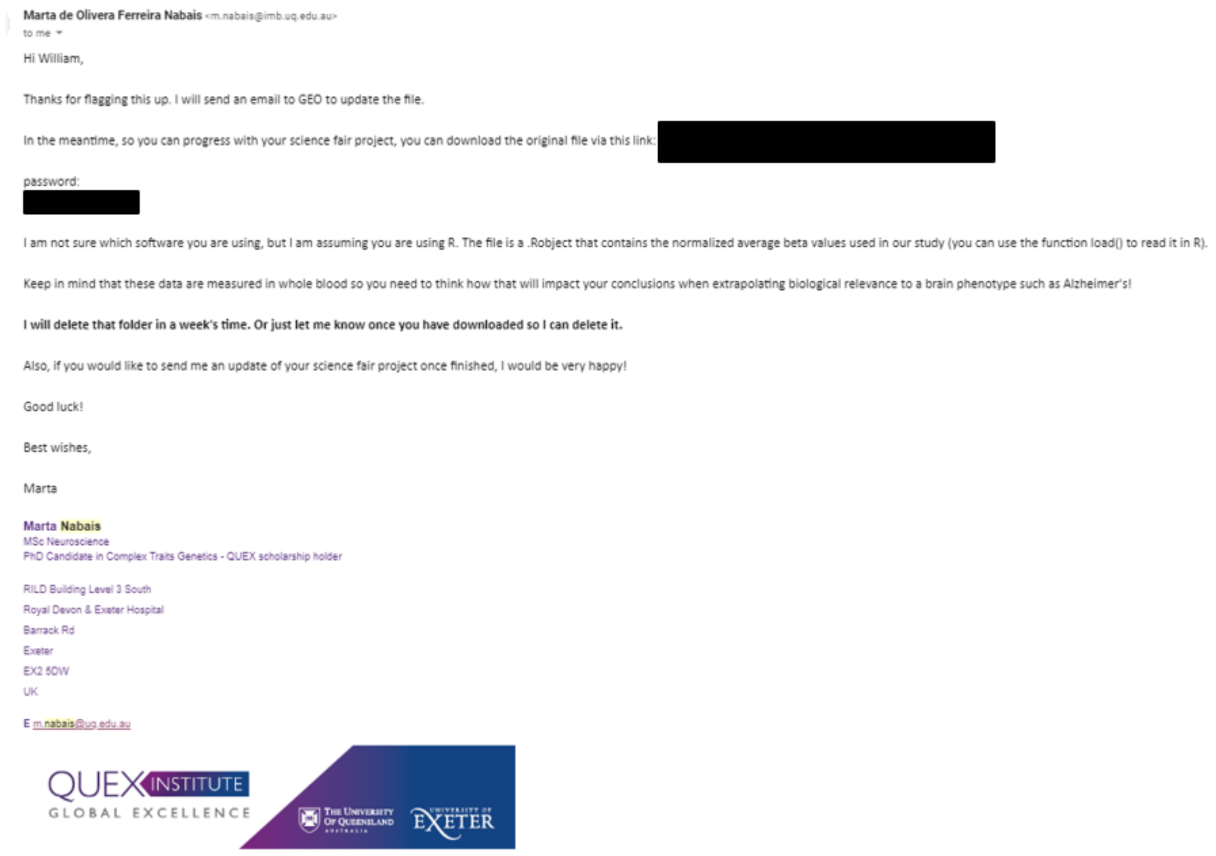
My name is William Guo, a high school student living in Irvine, CA, and I was hoping if you could help me with a question I have regarding your dataset GSE153712 deposited at GEO Datasets that I was looking to possibly use for my science fair project.

I specifically wanted to take a look at the DNA methylation values of the Alzheimer's disease associated CpG sites in your dataset in order to ultimately find specific sites that are important or can be associated with Alzheimer's disease. However, when I opened the file GSE153712_normalized_average_betas.txt.gz in Linux, I found that the first column is listed 1, 2, ... 862601 as opposed to probe names in the format of cgxxxxxxxx.

As such, could you please advise me on how I can pair the cgxxxxxxxx names to the normalized average beta values? Thank you very much for your time and consideration.

Sincerely,
William Guo

Figure 43: *Dr. Nabais's response and providing the original file*



Marta de Olivera Ferreira Nabais <m.nabais@imb.uq.edu.au>
to me

Hi William,

Thanks for flagging this up. I will send an email to GEO to update the file.

In the meantime, so you can progress with your science fair project, you can download the original file via this link: [redacted]

password:
[redacted]

I am not sure which software you are using, but I am assuming you are using R. The file is a .Robject that contains the normalized average beta values used in our study (you can use the function load() to read it in R).

Keep in mind that these data are measured in whole blood so you need to think how that will impact your conclusions when extrapolating biological relevance to a brain phenotype such as Alzheimer's!

I will delete that folder in a week's time. Or just let me know once you have downloaded so I can delete it.

Also, if you would like to send me an update of your science fair project once finished, I would be very happy!

Good luck!

Best wishes,

Marta

Marta Nabais
MSc Neuroscience
PhD Candidate in Complex Traits Genetics - QUEX scholarship holder

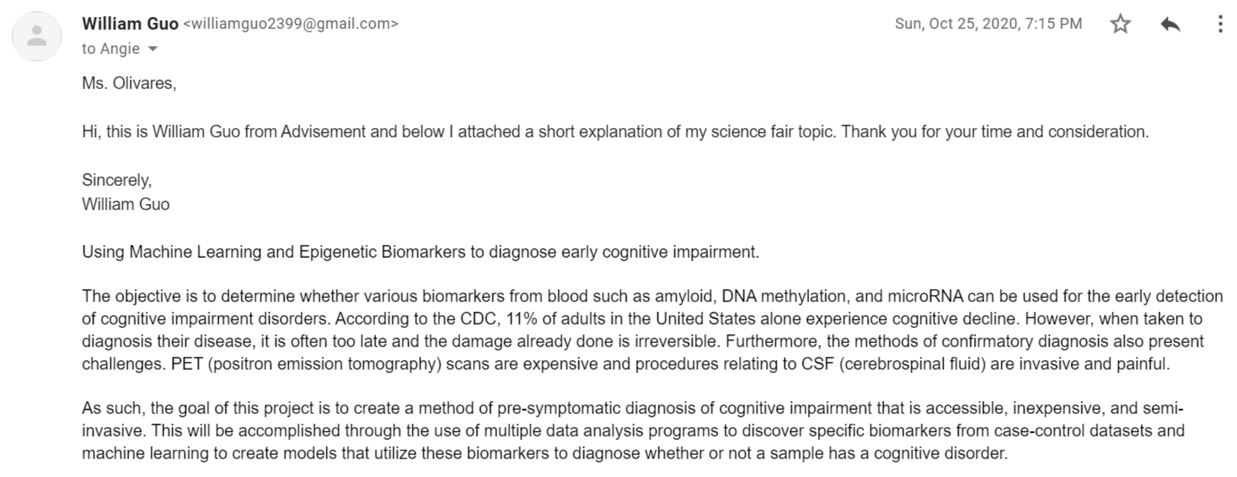
RILD Building Level 3 South
Royal Devon & Exeter Hospital
Barrack Rd
Exeter
EX2 5DW
UK

E m.nabais@uq.edu.au

Appendix D

Original Project Topic Proposal

Figure 44: *Emailing original science fair proposal to Advisor (Angie Olivaries)*



William Guo <williamguo2399@gmail.com>
to Angie

Sun, Oct 25, 2020, 7:15 PM

Ms. Olivares,

Hi, this is William Guo from Advisement and below I attached a short explanation of my science fair topic. Thank you for your time and consideration.

Sincerely,
William Guo

Using Machine Learning and Epigenetic Biomarkers to diagnose early cognitive impairment.

The objective is to determine whether various biomarkers from blood such as amyloid, DNA methylation, and microRNA can be used for the early detection of cognitive impairment disorders. According to the CDC, 11% of adults in the United States alone experience cognitive decline. However, when taken to diagnosis their disease, it is often too late and the damage already done is irreversible. Furthermore, the methods of confirmatory diagnosis also present challenges. PET (positron emission tomography) scans are expensive and procedures relating to CSF (cerebrospinal fluid) are invasive and painful.

As such, the goal of this project is to create a method of pre-symptomatic diagnosis of cognitive impairment that is accessible, inexpensive, and semi-invasive. This will be accomplished through the use of multiple data analysis programs to discover specific biomarkers from case-control datasets and machine learning to create models that utilize these biomarkers to diagnose whether or not a sample has a cognitive disorder.

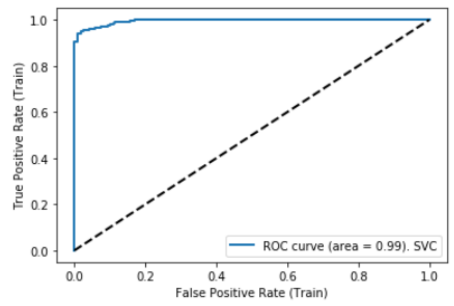
Appendix E

Code for AUC (Area Under the Curve) Graphs

Figure 45: *Lasso_3oo Model AUC graphs (SVC)*

```
# Draw AUC
svc_auc = roc_auc_score(y_train, svc.decision_function(X_train))
fpr, tpr, thresholds = roc_curve(y_train, svc.decision_function(X_train))
lw = 2
plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f). SVC' % svc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=lw, linestyle='--')
plt.xlabel("False Positive Rate (Train)")
plt.ylabel("True Positive Rate (Train)")
plt.legend(loc=4)
#plt.savefig('Training/Output/AUC_SVC_2.0_Training.png', figsize=(10, 10), dpi=300)
#plt.show()
```

```
<matplotlib.legend.Legend at 0x1e6d9b331c8>
```



```
# Draw AUC
svc_auc = roc_auc_score(y_valid, svc.decision_function(X_valid))
fpr, tpr, thresholds = roc_curve(y_valid, svc.decision_function(X_valid))
lw = 2
plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f). SVC' % svc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=lw, linestyle='--')
plt.xlabel("False Positive Rate (Validation)")
plt.ylabel("True Positive Rate (Validation)")
plt.legend(loc=4)
#plt.savefig('Training/Output/AUC_SVC_2.0_Training.png', figsize=(10, 10), dpi=300)
#plt.show()
```

```
<matplotlib.legend.Legend at 0x1e6d6a8ee08>
```

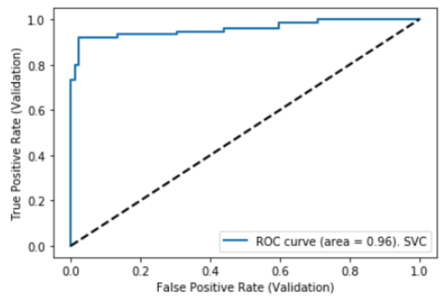
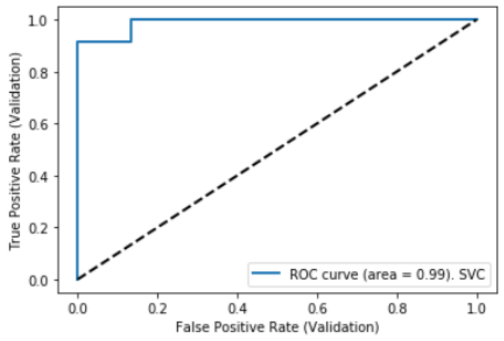


Figure 46: *ND → ND vs. ND → MCI Model AUC graphs (SVC)*

```
# Draw AUC
svc_auc = roc_auc_score(y_valid, svc.decision_function(X_valid_scaled))
fpr, tpr, thresholds = roc_curve(y_valid, svc.decision_function(X_valid_scaled))
lw = 2
plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f). SVC' % svc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=lw, linestyle='--')
plt.xlabel("False Positive Rate (Validation)")
plt.ylabel("True Positive Rate (Validation)")
plt.legend(loc=4)
#plt.savefig('Training/Output/AUC_SVC_2.0_Training.png', figsize=(10, 10), dpi=300)
#plt.show()
```

```
<matplotlib.legend.Legend at 0x1b6f7467708>
```



```
# Draw AUC
svc_auc = roc_auc_score(y_test, svc.decision_function(X_test_scaled))
fpr, tpr, thresholds = roc_curve(y_test, svc.decision_function(X_test_scaled))
lw = 2
plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f). SVC' % svc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=lw, linestyle='--')
plt.xlabel("False Positive Rate (Testing)")
plt.ylabel("True Positive Rate (Testing)")
plt.legend(loc=4)
#plt.savefig('Training/Output/AUC_SVC_2.0_Validation.png', figsize=(10, 10), dpi=300)
#plt.show()
```

```
<matplotlib.legend.Legend at 0x1b6f78ab488>
```

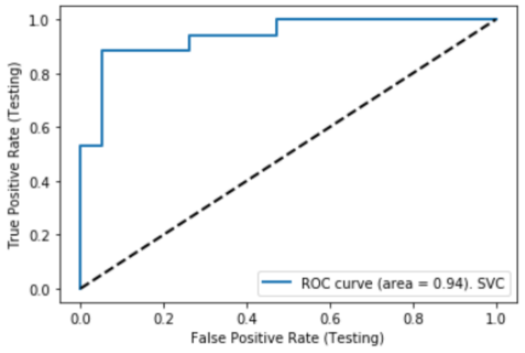
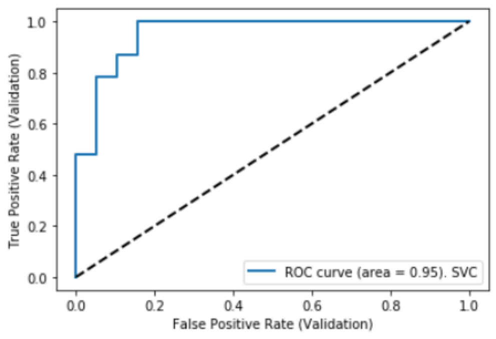


Figure 47: *MCI → MCI vs. MCI → AD Model AUC graphs (SVC)*

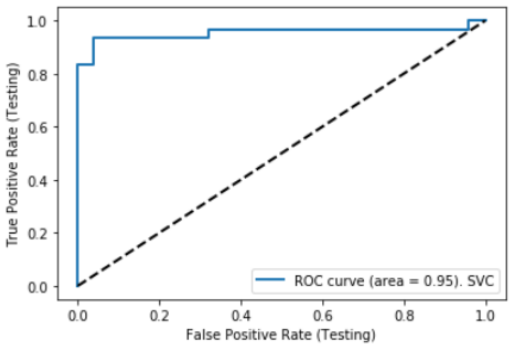
```
# Draw AUC
svc_auc = roc_auc_score(y_valid, svc.decision_function(X_valid_scaled))
fpr, tpr, thresholds = roc_curve(y_valid, svc.decision_function(X_valid_scaled))
lw = 2
plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f). SVC' % svc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=lw, linestyle='--')
plt.xlabel("False Positive Rate (Validation)")
plt.ylabel("True Positive Rate (Validation)")
plt.legend(loc=4)
#plt.savefig('Training/Output/AUC_SVC_2.0_Training.png', figsize=(10, 10), dpi=300)
#plt.show()
```

```
<matplotlib.legend.Legend at 0x24134280d48>
```



```
# Draw AUC
svc_auc = roc_auc_score(y_test, svc.decision_function(X_test_scaled))
fpr, tpr, thresholds = roc_curve(y_test, svc.decision_function(X_test_scaled))
lw = 2
plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f). SVC' % svc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=lw, linestyle='--')
plt.xlabel("False Positive Rate (Testing)")
plt.ylabel("True Positive Rate (Testing)")
plt.legend(loc=4)
#plt.savefig('Training/Output/AUC_SVC_2.0_Validation.png', figsize=(10, 10), dpi=300)
#plt.show()
```

```
<matplotlib.legend.Legend at 0x2413433eec8>
```



Appendix F

Acknowledgements

From William Guo,

I would like to conclude this presentation by expressing my utmost gratitude to Dr. Marta Nabais and the ADNI Data Sharing and Publications Committee (DPC). Both Dr. Nabais and the DPC provided me with access to the GSE153712 and the ADNI dataset, respectively, which made the creation of both the static and progressive models possible. In short, this project wouldn't have existed without their kind support.