FreeSurfer Segmentation Quality Assessment Using Machine Learning Classifiers

The GORDON ENGINEERING LEADERSHIP PROGRAM at Northeastern University

MGH/HST Athinoula A. Martinos Center for Biomedical Imaging

A Challenge Project Final Report presented

by

Matthew Goh

of

MGH/HST Martinos Center for Biomedical Imaging

to

The Gordon Engineering Leadership Program

in progress towards the requirements

for

The Graduate Certificate in Engineering Leadership

Northeastern University

Boston, Massachusetts

August 2016

1 SPONSOR APPROVED STATEMENT OF PROJECT IMPACT

CHALLENGE PROJECT STATEMENT OF IMPACT
Martinos Center for Biomedical Imaging – Goh

Project Impact:
The current state-of-art in magnetic resonance (MR) brain image segmentation, including those generated by the FreeSurfer (FS) software package, requires users to evaluate the quality of segmentation results "by eye" in order to determine whether manual edits are needed. This is feasible in low sample-size studies but extremely impractical with larger population. For this project, the candidate developed a tool which implemented machine-learning algorithms to classify FS segmentation results into "needs editing" or "does not need editing" categories. The deployment of this tool mitigated dependence on manual inspection of MRI segmentation quality, while also providing a novel automated approach to segmentation quality assessment.

Project Achievements: The project deliverable is a tool categorizing FS segmentations into "needs edit" or "does not need edit" categories. The inability to identify problematic segmentation volumes in a consistent and automated manner remained a significant barrier to widespread adoption of quantitative image analysis in the clinical setting. Therefore, a major milestone achieved was in driving progress in image segmentation state-of-the art. A secondary impact is that our tool can be leveraged to trace common sources of segmentation errors, which can be extrapolated to highlight the critical need for standardized image acquisition protocols across research centers. This would benefit both clinical and research settings.

Project Value:
Return on investment is quantified through labor costs. Inspecting each segmentation volume "by eye" is associated with approximately $11.70 in labor costs. In this information age, research efforts have ranged from analyzing ~400 volumes per imaging project to over 50,000 volumes per project. This translates to $4600 and $585,000 in labor costs, respectively. By contrast, this tool is capable of evaluating over 1000 segmentation volumes in less than 30 seconds; therefore, evaluating 50,000 volumes with our tool would cost ~$9.75. The savings this tool provides cannot be understated. Indirectly, ROI is quantified through MR image acquisition costs. Laboratories without the resources to both inspect and edit segmentation volumes have, in some cases, simply discarded low quality MRI data, choosing instead to only segment higher quality data. This is a net negative on ROI for that organization. Conversely, our tool would provide timesavings with respect to the inspection process, thereby enabling organizations to also consider segmenting low quality MRI data.

Candidate Impact:
The candidate developed technical skills in Python, machine learning, and MR image analysis. Interfacing machine-learning algorithms with MR data required developing programming skills, as well as technical acuity in order to understand and draw conclusions from the results of machine-learning experiments. Because the candidate did not possess many of these technical skills prior to the project, the candidate needed to advocate for his needs with technical guidance and understanding the developmental

methodology. In order to achieve this, the candidate needed to exercise effective communication and decision-making capabilities

Gordon Engineering Leadership Candidate	Date
Industry Sponsor/Advocate	Date

2 ABSTRACT

FreeSurfer is a software package developed at the Martinos Center for Biomedical Imaging that provides tools for neuroimaging data analysis. The goal of FreeSurfer is to allow users to generate computerized models of the human brain from *T1*-weighted MRI scans. FS performs cortical and subcortical segmentation, labeling, surface reconstruction, and other extensive analyses on brain morphometry.

The focus of this project was FreeSurfer's brain segmentation capability. In general, image segmentation is the task of dividing a computer image into a set of homogenous and non-overlapping regions based on specific criteria, for example, image intensity, depth, color, or texture. The result is a label of images identifying each homogenous region with some particular classification. In the case of FreeSurfer and structural (anatomical) analyses, FreeSurfer segments MRI scans of the brain into gray matter (GM), white matter (WM), cerebrospinal fluid (CSF), and a number of macroscopically visible brain structures, e.g. the thalamus, caudate, putamen, hippocampus. The resulting output is a set of labels that can be reconstructed into 3D surfaces for brain visualization.

Due to variations in image intensity and other image-related problems, the resulting segmentation may have errors that require manual correction. Therefore, it is crucial that users review each segmentation volume via visual inspection. If corrections are needed, the user marks edits either by erasing or filling GM or WM voxels, or by placing 'control points' which identify the marked pixel as WM. Because each segmentation volume may have well over 200 slices, this quality assessment (QA) process is extremely laborious and time consuming, such that QA is not realistic in many large sample-sized studies.

The purpose of this project was therefore to develop an automated and quantitative method to performing FreeSurfer segmentation QA. The approach employed was using a machine learning approach with the algorithms available at www.scikit-learn.org. The strategy is to test existing machine learning algorithms and refine the features used to make the classification. The ultimate purpose is to mitigate the reliance on evaluating segmentation quality "by eye", which is feasible in low sample-size studies, but extremely laborious and impractical in larger population studies. The algorithm would address the needs of providing some quantitative measure of quality, improving data consistency, and increasing throughput.

3 ACKNOWLEDGEMENTS

I would like to thank the following people for their time, dedication, patience, and mentorship.

Martinos Center for Biomedical Imaging

[redacted]
Assistant Professor in Radiology, Harvard Medical School
Assistant in Neuroscience, Massachusetts General Hospital

[redacted]

Instructor in Radiology, Harvard Medical School
Assistant in Neuroscience, Massachusetts General Hospital

[redacted]
Principal Research Scientist, McGovern Institute for Brain Research, MIT
Assistant Professor in Otology and Laryngology, Harvard Medical School

[redacted]
Professor in Radiology, Harvard Medical School
Neuroscientist, Massachusetts General Hospital
Director, Computational Core, Martinos Center, MGH

Gordon Institute of Engineering Leadership, Northeastern University

[redacted]

Northeastern University
[redacted], Associate Professor, Department of Electrical and Computer Engineering

4 TABLE OF CONTENTS

5 LIST OF FIGURES AND TABLES

6 INTRODUCTION

Noninvasive brain imaging technologies such as magnetic resonance imaging (MRI) have opened new dimensions for the analysis of brain function and anatomy. MRI has become invaluable to clinicians for medical diagnostics, whereas MRI allows researchers to perform in-depth examinations of relationships between anatomy, diseases, treatments, and underlying genetics. While continual improvements in MRI technology provide large amounts of data with increasing levels of quality, it is nevertheless the interpretation of details in these images that remain key in fully appreciating their informational complexity.

Image segmentation has become an essential step in extracting meaningful information from these images. Generally speaking, image segmentation is the task of dividing a computer image into a set of homogenous and non-overlapping regions based on some criteria, for example, image intensity, depth, color, or texture. The result is a label of images identifying each homogenous region with some particular classification.

In structural (anatomical) MRI analysis, each voxel (3D pixel) element in an MRI scan is assigned a tissue class (tissue classification) following the segmentation process. In the healthy brain, these elements can be classified as white matter (WM), gray matter (GM), or cerebrospinal fluid (CSF). Segmentations of higher specificity may include structures beneath the outermost layer (the cortex), such as the thalamus, caudate, putamen, or hippocampus. In an injured brain, pathological elements may be classified as tumors, edematous (fluid-filled), hemorrhagic (bleeding), or necrotic tissue. The resulting label map (a brain mask) can be reconstructed into a 3D surface for brain visualization, quantifying morphometry, formation of realistic tissue models, or for pre-surgical planning.

For a long time, this segmentation process has depended on manual tracing by trained experts in neuroanatomy. However, as the size of datasets have increased, the amount of time required to perform these segmentations have become immensely prohibitive. Furthermore, manual tracing is prone to inter- and intra-user variability, such that resulting quantitative metrics can vary widely among experts. For these reasons, the advent of reliable, automatic segmentation algorithms has been critical to the evolution of the state-of-the-art.

To invoke the FS segmentation and reconstruction processing stream, a user inputs a single DICOM (Digital Imaging and Communications in Medicine) or NIFTI (Neuroimaging Informatics Technology Initiative) formatted file from a T1-weighted MRI scan. After several hours of running time (depending upon computer hardware), the output is a set of folders containing the resulting segmentation, 3D surface representations, and computed data on brain morphometry.

Three main types of image-related problems still unfortunately exist in image segmentation, however. The first is *noise* which may exist in images, altering the intensity of a pixel such that its classification becomes uncertain[1]. The second is *intensity inhomogeneity*, where the intensity level of a tissue type varies over the extent of the image. The third is *partial volume averaging* – because images have a finite pixel size, individual pixel volumes may contain a mixture of tissue classes such that the intensity of a pixel in the image may not be consistent with any single tissue class. The combination of these problems, in addition to variability in brain size and shape, may lead to segmentation errors that require manual user correction, depending on the severity of the error.

For FreeSurfer, the manual correction process requires the user to first manually inspect each segmentation volume in a slice-by-slice fashion. If manual editing were necessary, the user would make corrections in the FS editor by hand. This is an incredibly time consuming process which can be compounded by the fact that there may be hundreds or thousands of volumes within a given dataset.

The purpose of this project is therefore to develop an automated and quantitative method to FreeSurfer segmentation quality assessment, using a machine learning approach with the algorithms available at www.scikit-learn.org. The strategy is to test existing machine learning algorithms and refine the features used to make the classification. The ultimate purpose is to mitigate the reliance on evaluating segmentation quality “by eye”, which is feasible in low sample-size studies, but extremely laborious and impractical in larger population studies. The algorithm would address the needs of providing some quantitative measure of quality, improving data consistency, and increasing throughput.

6.1 Product Mission Statement

Mission Statement: Create a QA tool that quantitatively assesses the quality of a FreeSurfer segmentation and indicating to the user a) the number of segmentations that are acceptable, and b) the expected % of false negative misclassifications	
Product Description	• A QA tool sorts an input FreeSurfer segmentation into a 'needs editing' or 'does not need editing' class
Benefit Proposition	• Reduce reliance on a 'human-in-the-loop' approach to QA • Minimize time spent on QA • Increase throughput of quality FreeSurfer segmentations
Key Organization Goals	• Augment benefit to users using FreeSurfer to evaluate MRIs
Primary Market	• FreeSurfer user base – FS is used worldwide (> 20,000 downloads)
Assumptions and Constraints	• QA tool can be developed using *existing* machine learning algorithms • Effective classification requires carefully researched and tested algorithms and input features
Stakeholders	• FreeSurfer developers and users

6.2 Project Definitions

brain mask – a binary overlay containing integer values, typically "1s" inside the labeled region and "0s" outside the labeled region (which is typically any area outside the brain). The resulting brain mask is an image labeling specific tissues that can be overlaid on an anatomical scan such as a T_1- weighted MRI scan

cerebrum - the principal and most anterior part of the brain in vertebrates, located in the front area of the skull and comprises two hemispheres, left and right, and is separated by a fissure. It is responsible for the integration of complex sensory and neural functions and for the initiation and coordination of voluntary activity in the body

contract research organizations - organizations that provide outsourced services (pharmaceutical, biotechnology, management, clinical research) on a contracted basis

cortical – of or relating to the cerebral cortex. The cerebral cortex is the outermost layer of the cerebrum, and comprises gray matter, which consists of neuronal cell bodies, glial cells, synapses (connections between neurons), and capillaries. The cortex contrasts with the underlying white matter, which consists of glial cells and the white myelinated axons of neurons

DICOM – Digital Imaging and Communications in Medicine (DICOM). It is a data standard that defines the formats for medical images with the quality necessary for clinical use. DICOM is implemented in almost every radiology, cardiology imaging, and radiotherapy device (X-ray, CT, MRI, ultrasound, etc.), and increasingly in devices in other medical domains such as ophthalmology and dentistry. A single DICOM file contains both a header (which stores information about the patient's name, the type of scan, image dimensions, and other scanner parameters), as well as all of the image data (which can contain information in three dimensions).

FreeSurfer – A suite of tools developed at the Martinos Center for Biomedical Imaging that performs neuroimaging analysis. Its segmentation function is the focus of this project. FreeSurfer provides anatomical analysis tools, including: representation of the cortical surface between white and gray matter, segmentation of white matter, skull stripping, B1 bias field correction, nonlinear registration of the cortical surface of an individual with a stereotaxic atlas, labeling of regions of the cortical surface, statistical analysis of group morphometry differences, and labeling of subcortical brain structures and much more

feature – for purposes of this proposal, 'feature' will be used within the context of machine learning terminology. A feature is an individual measurable property of a phenomenon being observed. It may be morphometric information (e.g. brain volumes, thicknesses, curvatures) or other statistics calculated from an MR image signal (e.g. intensities, contrasts, signal to noise ratio, contrast to noise ratio, etc.)

gray matter (GM) - composes the cerebral cortex; GM consists of neuronal cell bodies, glial cells, synapses (connections between neurons), and capillaries

HST - Division of Health Sciences & Technology at Massachusetts Institute of Technology (MIT)

LCN – Laboratory for Computational Neuroimaging; the LCN is one of the many labs at the Martinos Center for Biomedical Imaging

machine learning – subfield of computer science based upon pattern recognition and computational learning in artificial intelligence. The machine learning philosophy is to enable computers to act without being explicitly programmed, i.e. to learn and make predictions based on data

manual editing – refers to the process by which a user corrects a segmentation by hand, i.e. using a mouse and the FreeSurfer editing tool

Martinos Center for Biomedical Imaging – The parent organization of the LCN located at the MGH East Campust in Charlestown. The A.A. Martinos Center is a research center committed to the development and applied use of advanced biomedical imaging technologies, and to the advancement of translational research and education

MGH - Massachusetts General Hospital

MRI – magnetic resonance imaging, a medical imaging technique used in radiology based on magnetic fields and radio waves. Images are formed to investigate anatomy and physiology of the body.

segmentation – Segmentation is the process of delineating the different tissue types within an MRI scan

subcortical – relating to or denoting the region of the brain below the cortex

T_1-weighted – one of the basic pulse sequences in MRI that provide gray matter/white matter contrast

volume – a ‘volume’ (3D) refers to a complete series of MRI slices (2D)

voxel – a 3D pixel

6.3 Company/industry background

The project will be completed in affiliation with the *Laboratory for Computational Neuroimaging* (LCN) of the Athinoula A. Martinos Center for Biomedical Imaging, located at Massachusetts General Hospital (MGH). The A.A. Martinos Center is a research center committed to the development and applied use of advanced biomedical imaging technologies, and to the advancement of translational research and education. The Center is comprised of many laboratories and programs, one of which is the *Laboratory for Computational Neuroimaging*. The goals of the LCN are to build computational tools and algorithms that optimize neuroimaging data analysis within the domains of structural, functional, and diffusion magnetic resonance imaging. The LCN spends substantial resources investigating the labeling of fiber pathways and subcortical structures, and on the development of MR scanner pulse sequences and image reconstruction methods that enhance image tissue contrast, reduce motion artifacts, and improve the reliability of scans within and across individuals.

The A.A. Martinos Center began in 1999 when Thanassis and Marina Martinos of Athens, Greece, presented a gift of $20 million to the Harvard-Massachusetts Institute of Technology (MIT) Division of Health Sciences & Technology (HST) to honor the memory of their daughter, Athinoula. The purpose of the gift was to establish a biomedical imaging center dedicated to research and the medical application of new technologies. HST then invited MGH to participate in founding the Athinoula A. Martinos Center for Biomedical Imaging, thereby forming a partnership. This formed a synergistic alliance uniting the clinical and imaging expertise of the MGH Nuclear Magnetic Resonance (NMR) Center with the engineering and neuroscience expertise of HST. The A.A. Martinos Center launched in 2000 under the directorship of Bruce R. Rosen, MD, PhD, with a faculty of approximately forty investigators and over $23 million in existing biomedical imaging equipment. The Center is located on the MGH research campus in the Charlestown Navy Yard, with a satellite facility on the MIT campus.

7 MARKET AND IMPACT ASSESSMENT

7.1 External Market

The external market was considered all organizations beyond the Martinos Center, for example FreeSurfer users, researchers, clinicians, developers, or contract research organizations who may all utilize automatic segmentation tools. Segmentation tools are valuable not only for exploring brain anatomy, but also for formulating novel therapeutic treatment strategies. Taking Alzheimer’s disease (AD) as an example, AD has long been thought to be a disease of the gray matter due to severe neurodegeneration and atrophy of many gray matter structures. Consequently, volumetric studies (by means of MRI segmentation) on GM structures such as the hippocampus, on one hand, have provided high diagnostic value. On the other hand, these same types of volumetric studies have also indicated that *white matter* abnormalities are important components of the disease. Thus, segmentation can ultimately benefit patients by first shedding knowledge on how clinicians can optimize therapy regimens.

The state-of-the-art in brain segmentation software consisted of very few tools with similar capabilities to that of FreeSurfer. In the research space, some of the more popular tools were FreeSurfer, FSL, and BrainVoyager. FreeSurfer, however, was capable of performing both cortical and subcortical segmentation, whereas FSL was limited to GM, WM, and CSF segmentation. BrainVoyager performed only cortical segmentation and cost €5300 ($5770USD) per license, whereas FreeSurfer was freely available. Thus, FreeSurfer possessed several advantages over its competition.

In the commercial space, there were several segmentation tools available for segmenting CT datasets, but disappointingly few for that of MRI. Based on market research, the only notable automatic segmentation software available for MRI was Neuroreader, which segments only the hippocampus, and has only recently received FDA approval. The paucity of clinically approved segmentation software is in part a result of the overall reluctance of clinicians to employ quantitative approaches to image analysis. This is largely due to the lack of standardized methods in a) image acquisition, b) information extraction from these images, and c) in correlating quantitative metrics with internationally recognized criteria.

These circumstances provided FreeSurfer with a unique opportunity to expand its functionality. For developers, the creation of an automated, quantitative, and dependable means to perform FreeSurfer quality assessment (QA) can aid the identification of common sources of segmentation errors. If these sources were found to result from certain image acquisition parameters, it would highlight the case that standardizing image acquisition protocols is critical for revolutionizing medical image analysis. This would bring benefit to both clinical and research settings.

For users, this project addressed the time-consuming methodology of visual slice-by-slice inspection. This process is illustrated by the sample decision tree below:

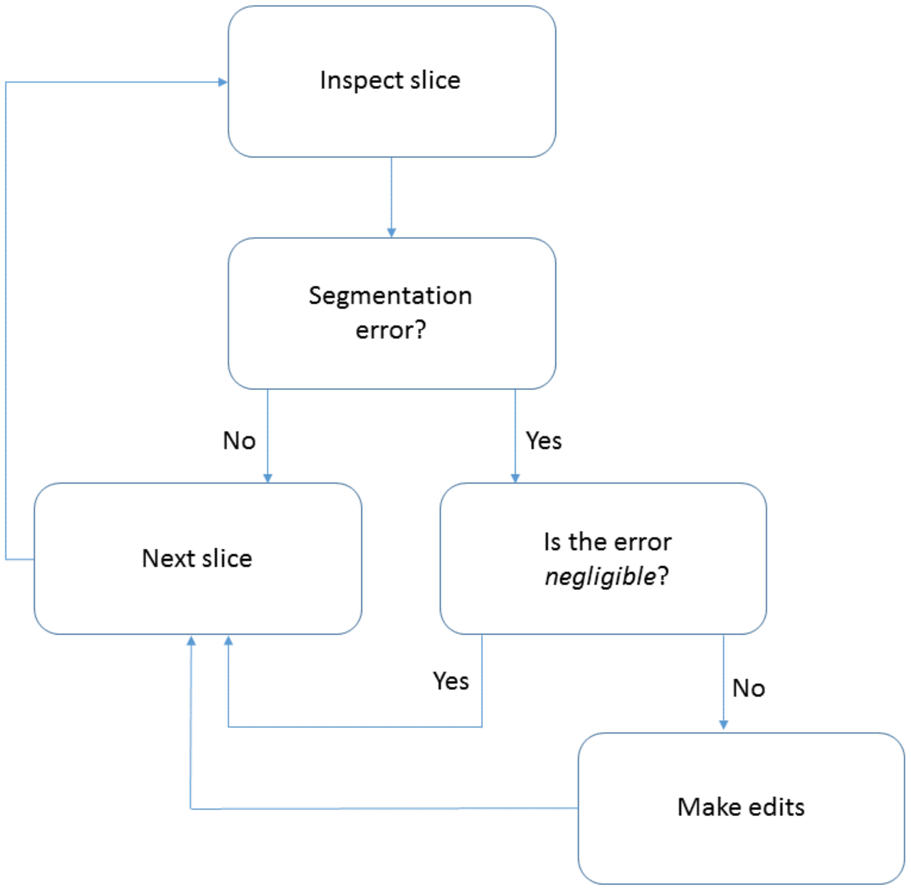


Figure 8.1. Decision tree for the 'visual inspection' approach

For each slice in the segmentation volume (see Fig. 2), a user needed to determine whether the segmented result met some criteria of acceptability. This decision depended upon on the users knowledge of neuroanatomy and judgement in whether an error could be safely ignored (a study interested only in white matter, for example, might not need accurate segmentation of gray matter).

This approach was problematic for several reasons. In time costs, performing visual inspection ranged from 20 seconds to a few minutes for each slice. A given segmentation volume may contain well over 200 slices, translating into a minimum time cost of 33 minutes per volume. Because many neuroimaging studies now contain cohort sizes numbering in the hundreds and thousands, it was clear that manual inspection was not a viable approach to QA [2 3 4 5].

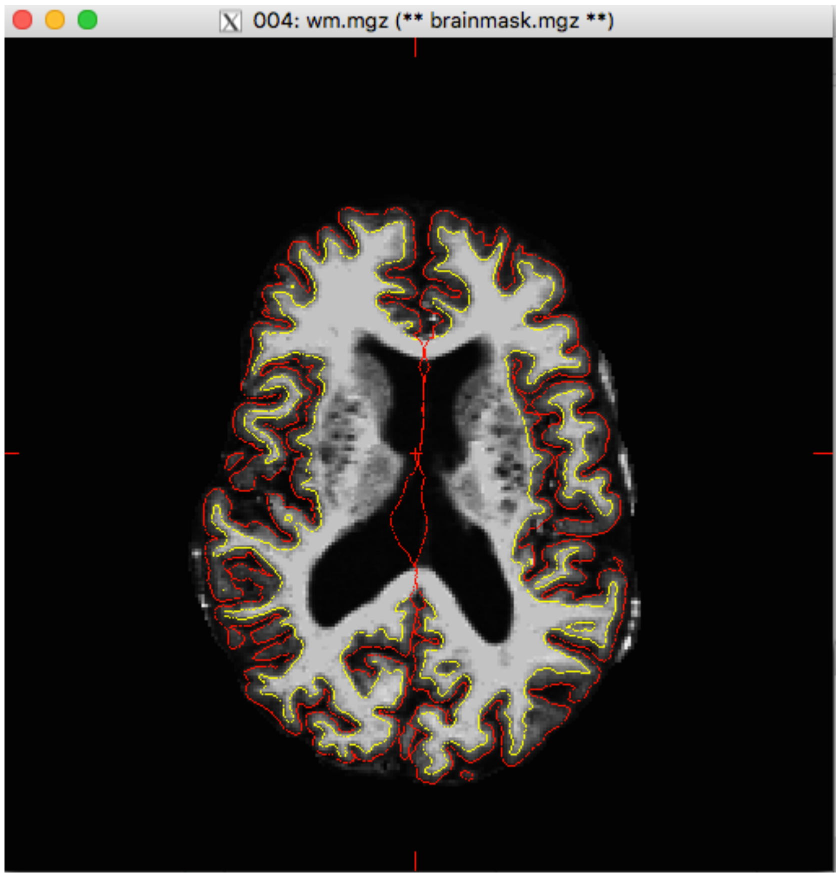


Figure 8.2. A sample slice as seen by a user during FS QA. A segmentation volume typically contains well over 200 slices.

Time costs also translated into labor costs. Based on salary averages from two separate sources, the national annual salary estimates of a Staff Research Associate I (the position most likely to perform segmentation QA) were $46,774 and $48,384 respectively[67]. Taking the average of the two ($47,579 /yr), this equated to an hourly rate of $22.87, or a cost of $12.58 per segmentation volume. Large neuroimaging studies utilizing ~400 volumes per imaging project to over 50,000 volumes per project would thus consume $4,600 to $585,000 in labor costs, respectively. By contrast, this tool is capable of evaluating roughly 1000 segmentation volumes in less than 30 seconds, or 66,000 volumes in 30 minutes. The cost savings this tool provides cannot be understated. See Table 1 for a summary of comparative costs.

A final issue with the visual inspection methodology was its susceptibility to intra- and inter-user variability. Based on user feedback, it was hardest to decide whether edits were needed when the segmentation was almost entirely accurate. Small errors required an attentive and experienced eye to recognize them. This was thus a highly qualitative means of QA that depended on the user's experience and personal criteria of acceptability.

	Visual Inspection	QA Tool
Time cost per volume (s)	1980 s	0.03 s
Dollar cost per volume ($)	$12.58	$1.91e-4
Number volumes assessed per 33 minutes (1980 seconds)	1	65,340

Table 8.1. Summary of comparative costs. Visual inspection vs QA tool

7.2 Internal Market

The internal market was considered FreeSurfer developers at the Martinos Center. The development of a QA tool could provide future benefit to FreeSurfer development; a central theme to the project was identify specific causes of segmentation errors, which can ultimately aid developers in fine-tuning the segmentation algorithms. These findings could then be applied to a future tool that directly searches for and fixes segmentation errors, eliminating altogether the need for manual user input in the segmentation correction process.

7.3 Customer and Customer Needs Assessment

The customer needs were assessed through an in-depth examination of the decision tree presented in Figure 8.1.

The first decision point (Is there a segmentation error?) depended upon the user's expertise in neuroanatomy. A user would inspect the segmentation in the FreeSurfer viewing and editing tool called Tkmedit, where the gray matter/cerebrospinofluid (CSF) and white matter/gray matter boundaries were delineated by the red and yellow lines, respectively (refer back to Figure 8.2).

The first decision point would thus be a question of accuracy, and, by and large, a binary decision of "yes, needs edit" or "no, does not need edit" response. However, an issue of 'ground truth' is presented in that it is impossible to know with 100% confidence the exact boundaries between the respective tissues without histological examination. The important point is that in regions where the user was not confident of its accuracy, more time would be required to determine whether editing was needed.

If the user decided that an error occurred, then the next decision point was reached, which is concerned with whether the error is negligible (i.e. can safely ignore the problem). A negligible error is one that would likely yield a negligible change in the computed metrics (e.g. volume, curvature, or thickness). This again was reliant on qualitative observation, and the user's past experience with segmentation correction.
This decision point raised yet another issue to consider: a segmentation error appearing significant at the slice level ('microscale') may in actuality be negligible when considered at the volume level (macroscale). Suppose an MRI segmentation containing errors in slices 1, 27, and 123 in a 256-slice volume. The user may or may not conclude that errors occuring only 3 slices out of 256 are negligible. In either case, this conclusion could not be reached until after the user inspected every slice. This scenario highlights two major problems:

1) The decision is qualitative
2) The process is inefficient in time costs

In larger population studies, the issues with manual inspection are magnified: suppose in a cohort of 100 subjects, 10% of the segmentation volumes were determined to need editing. The user may decide that the remaining 90 “good” volumes (90% of 100) are sufficient to perform the analysis, and discards the 10% that needed to be edited. In this scenario, the user could not reach that conclusion without first visually inspecting all 100 volumes to determine the 90%/10% breakdown.

In summary, the customers’ needs were a QA tool which:
1. Provided a binary “edit”/”do not edit” response
2. In the case of “edit”, quantified the degree of error to guide the user in determining whether the error is negligible

7.4 Economic Impact and Return on Investment

8.4.1 Estimated Impact

The main groups that benefit from this project are clinicians, researchers, and FreeSurfer users. For FS users, the estimated impacts are:

1. Automation of quality assessment
2. Reduced reliance on ‘human-in-the-loop’ when performing QA
3. Reduced inter- and intra-user variability in the quality assessment process

The direct benefit to users is that they no longer need to spend significant time visually inspecting the segmentation results. This would reduce analyses times, and can potentially drive down research costs by a) reducing the number of research assistants needed, and b) reducing (paid) time spent on recruiting and training new lab members.

In certain cases, labs without the resource to edit segmentation volumes will simply discard data rather than make corrections. For example, the Alzheimer’s Disease Neuroimaging Initiative (ADNI), funded with $67 million in 2004 and another $67 million in 2011, discards approximately 15% of FS-segmented data as a result of poor quality. No attempt is made at editing these volumes due to the sheer amount of data. Consequently, there is a cost associated with discarding bad data in terms of money spent on data acquisition. An automated QC tool would therefore allow such organizations to edit the data more quickly, thereby maximizing their return on investment.

Clinicians and researchers will also receive benefit due to the reasons discussed in Section 8.1.

8.4.2 Return on Investment

Return on investment (ROI) to the organization was not quantified in dollar value because FreeSurfer is freely available and not in direct competition with commercial software. ROI was instead expressed in terms of added potential for FreeSurfer to become FDA-approved (Food and Drug Administration). FDA approval would enable the Martinos Center to bill insurance companies when FreeSurfer is used in the clinical setting. This would bring additional revenue to the organization. Receiving FDA approval includes an evaluation process that would examine a) the technology behind FreeSurfer, and b) the process used to build the technology. Both elements would examine FreeSurfer segmentation reliability and accuracy rates. User-reported accuracy rates (based on visual QA), even if low, would provide a weaker argument for reliability compared to an objective and consistent means of performing QA by way of this tool. Therefore, there is significant potential for ROI due to the creation of this tool.

External organizations also have potential to receive ROI from use of this tool. was quantified in labor and data acquisition costs. As stated in the previous section, the national average salary for a Staff Research Associate I was estimated to be $46,774 and $48,384 from two separate sources. Taking the average of the two ($47,579 /yr), this equates to an hourly rate of $22.87, or a cost of $12.58 per segmentation volume based on a time estimate of 33 minutes per volume. There are no reported numbers on total labor hours spent annually evaluating segmentation volumes. However, Table 8.4 below contains data on major brain imaging projects that utilize FreeSurfer, thereby highlighting the scale at which many neuroimaging studies take place. The QA tool developed here would benefit organizations worldwide by reducing labor costs.

Major Brain Imaging Projects that use FreeSurfer	
Project	**Number of Cases**
Alzheimer's Disease Neuroimaging Initiative (ADNI)	>5000
Framingham Heart Study (FHS)	>5000
Human Connectome Project (HCP)	1200
Superstruct Project	1500
Open Access Series of Imaging Studies (OASIS)	416
Enhanced NeuroImaging Genetics Meta Analysis (EN	>50000
UK Bio Bank	~100000
Rotterdamn Study	12000

Table 8.4. Cohort sizes of major neuroimaging projects

ROI to external organizations was also quantified in terms of data acquisition costs. In many cases, organizations without the resources to review and edit segmentation volumes have discarded low quality MRI scans rather than attempting to segment and QA these volumes, preferring instead to devote attention only to higher quality scans. For example, the Alzheimer's Disease Neuroimaging Initiative (ADNI), funded with $67 million in 2004 and another $67 million in 2011, discards approximately 15% of data due to poor quality. For context, a typical research scan costs approximately $500-$600 per hour ($1500 and over for clinical scans), and a typical scan without contrast agent requires 35 minutes to complete[8]. This translate to costs between $292 and $350 per scan that is discarded. This QA tool therefore provides savings in terms of enabling organizations to leverage all data since the time needed to QA segmentation data is significantly reduced. Furthermore, organizations that *overcollect* data in order to accommodate turnover rates will no longer need to do so.

7.5 Market Challenges and Risks

The market risks were anticipated to be minimal. Segmentation QA is not an area developers often invest time on, therefore there was low risk that another organization would be the first produce an automated QA tool. In the research space, free segmentation software is also often preferred over commercial tools in order to minimize research costs.

8 TECHNOLOGY DESCRIPTION

8.1 Overview of the Technical Challenge

This project sought to improve the status quo of assessing FreeSurfer segmentation volumes. Visual inspection of segmentations had been standard practice; for reasons described in above sections, this approach suffered from many problems. Consequently, the aim of this project was to create a tool which could perform FreeSurfer segmentation quality assessment automatically. The two most important pieces of information we wished to present to users with this tool were a) the number of segmentations that passed inspection, and b) of those that passed, the proportion of those that were incorrectly classified (false negative).

The project incorporated cross-disciplinary knowledge in machine learning, MRI segmentation analysis, and computer science. Interfacing MRI segmentation data with machine learning algorithms required computer programming in Python; MATLAB and shell scripting were needed on occasion.

Prior to the project, the candidate's programming experience resided primarily with MATLAB, so becoming familiar with the Python language was a substantial technical challenge.

Because implementing machine learning classifiers was the central basis to developing this tool, it was critical for the candidate to obtain a thorough understanding of relevant topics.

An additional significant challenge was in determining which types of predictions could be made with the most accuracy. Early experimental runs demonstrated that binary classifications ("edit" or "don't edit") could not be achieved with desirable accuracy using the initial feature set. Thus, other "forks" needed to be explored. These include:

a) Experimenting with other types of predictions
b) Experimenting with different feature sets
c) Experimenting with different machine-learning algorithms

Every experimental run required high amounts of technical acuity in order to understand implications of that experiment, to draw conclusions, and to decide which next steps would most likely to lead to success. Determination of which methods or steps to explore also drew from 'soft' skills such as creativity and intuition.

8.2 Product Specifications

Primary Function	1) Provide “need edit” or “does not need edit” output for each segmentation 2) Output false negative discovery rate
Secondary Goals	Guide user to focus on problematic regions or voxels of the segmentation
Software Requirements	Python Anaconda
Software Input	Information extracted from FreeSurfer segmentation results. These constitute the feature set upon which the machine learning classifier was trained on
Software Output	a) Number of segmentations that can be ignored (passed inspection) b) False negative discovery rate (of those rated negative, how many of these can be expected to be false negatives)

8.3 Scientific Principles Applied

The primary scientific principles applied were:

1) MRI segmentation
2) Machine learning and statistics
3) Computer science

The following section provides background information on each of these principles and explains how they pertain to the project.

1. **MRI segmentation**

   In brain MRI analysis, image segmentation is an essential step to extracting meaningful information. It is used for visualizing anatomical structures, analyzing brain changes, delineating pathological regions, and for surgical planning.

   Generally speaking, image segmentation is the task of dividing a computer image into a set of homogenous and non-overlapping regions based on some criteria, for example, image intensity, depth, color, or texture. The result is a label of images identifying each homogenous region with some particular classification.

   In structural (anatomical) MRI analysis, each voxel (3D pixel) element in an MRI scan is assigned a tissue class (tissue classification) following the segmentation process. In the healthy brain, these elements can be classified as white matter (WM), gray matter (GM), or cerebrospinal fluid (CSF). Segmentations of higher specificity may include structures beneath the outermost layer (the cortex), such as the thalamus, caudate, putamen, or hippocampus. In an injured brain, pathological elements may be classified as tumors, edematous (fluid-filled), hemorrhagic (bleeding), or necrotic tissue. The resulting label map (a brain mask) can be reconstructed into a 3D surface for brain visualization, quantifying morphometry, formation of realistic tissue models, or for pre-surgical planning.

For a long time, this segmentation process has depended on manual tracing by trained experts in neuroanatomy. However, as the size of datasets have increased, the amount of time required to perform these segmentations have become immensely prohibitive. Furthermore, manual tracing is prone to inter- and intra-user variability, such that resulting quantitative metrics can vary widely among experts. For these reasons, the advent of reliable, automatic segmentation algorithms has been critical to the evolution of the state-of-the-art.

FreeSurfer (FS) is a widely used automatic segmentation software application developed that provides tools for brain MRI analysis, specifically cortical and subcortical segmentation, labeling, surface reconstruction, and statistical analyses on brain morphometry. To invoke the FS segmentation and reconstruction processing stream, a user inputs a single DICOM (Digital Imaging and Communications in Medicine) or NIFTI (Neuroimaging Informatics Technology Initiative) formatted file from a T1-weighted MRI scan. After several hours of running time (depending upon computer hardware), the output is a set of folders containing the resulting segmentation, 3D surface representations, and computed data on brain morphometry.

Three main types of image-related problems still unfortunately exist in image segmentation, however. The first is noise which may exist in images, altering the intensity of a pixel such that its classification becomes uncertain. The second is intensity inhomogeneity, where the intensity level of a tissue type varies over the extent of the image. The third is partial volume averaging – because images have a finite pixel size, individual pixel volumes may contain a mixture of tissue classes such that the intensity of a pixel in the image may not be consistent with any single tissue class. The combination of these problems, in addition to variability in brain size and shape, may lead to segmentation errors that require manual user correction, depending on the severity of the error.

For FreeSurfer, the manual correction process requires the user to first manually inspect each segmentation volume in a slice-by-slice fashion. If manual editing were necessary, the user would make corrections in the FS editor by hand. This is an incredibly time consuming process which can be compounded by the fact that there may be hundreds or thousands of volumes within a given dataset

2. Machine learning and statistics

Machine learning (ML) is a subfield of computer science that deals with constructing algorithms that learn from some properties (features) of a dataset and apply these properties to make predictions on new, unseen data. ML tasks can be generalized into supervised learning and unsupervised learning categories.

A supervised machine learning problem is one in which training data is provided. The training data set consists of training examples, where each example is a pair of an input object and a desired output "target" value. Stated another way, supervised learning algorithms infer a mapping between features and labels from labeled training data. Supervised learning problems can further be categorized into classification or regression problems. In a classification problem, each sample belongs to two or more classes, and the objective is to predict the class of unlabeled data based on existing labeled data. Thus, classification problems are a discrete (as opposed to continuous) form of supervised learning, where the number of output categories are finite. Conversely, a supervised learning problem is a regression problem if the desired output value consists of one or more continuous variables.

In unsupervised learning, the training data consists of input objects without corresponding target values. The objective is to infer a function that describes some hidden structure to the unlabeled data. The problems here can be similar to clustering, where the goal may be to discover groups of similar examples within the data, or similar to density estimation, where the goal may be to determine the distribution of data within input space.

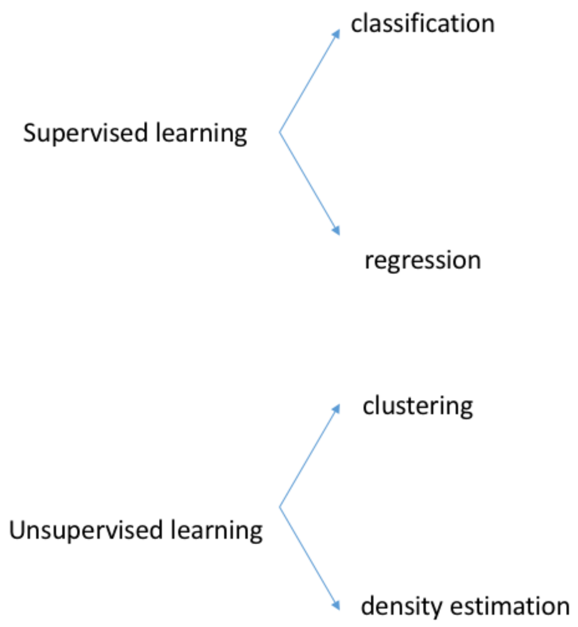


Figure 9.3.1. Breakdown of machine learning categories

This project consisted of a classification problem. Therefore, only supervised learning algorithms were employed.

Feature selection is a machine learning practice that was employed to select out the most useful features in making the predictions. The principal idea behind how feature selectors work is also grounded in statistics: feature selection algorithms analyze the distribution of a given metric, compare the distribution to the distributions across all other metrics, then computes the probability that each metric was drawn from the same distribution.

Other statistical concepts applied include data normalization, receiver operating characteristics (ROC) curves, and bootstrapping. Understanding normalization methods was crucial for properly accounting for the heterogeneity of a given metric across subjects.

The application of data normalization and feature selection are described in greater detail in the next section.

A number of critical data pre-processing steps were also performed; these are described in section 10.1.

3. **Computer science**

   This project was written in Python. The programming aspects include interfacing segmentation data with machine learning algorithms, implementing the algorithms and relevant functions, and outputting the results of machine-learning experiments. As with learning any programming language, the primary challenges were becoming familiar with the language's syntax, keywords, data structures, and functions. It was also important to be cognizant of how these properties differ from other languages that the candidate had experience with.

9 TECHNOLOGY APPROACH AND RESULTS

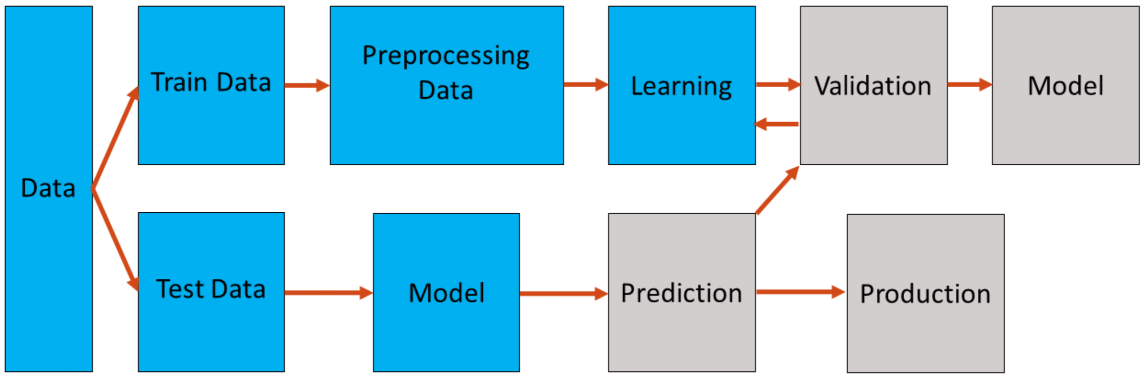
9.1 Development Approach and Methods

The developmental approach of the project was projected to include several phases or layers. The first layer was to first test machine learning algorithms available at http://www.scikit-learn.org/stable, using the standard FreeSurfer segmentation output metrics as the input features. These inputs are the components of the input matrix used to make the classification. An example of these metrics is shown in Table 10.1.1.

The second layer was to devise new metrics that are estimated to be more effective features compared to those used in the first layer; the project team had anticipated that the standard FreeSurfer metrics would not contain the types of information needed to accurately make the desired classifications. Nonetheless, in-depth analyses of the standard output metrics were necessary in order to fully understand how the algorithms behave.

The third layer was to devise a component of the quality assessment tool that is capable of guiding users to the specific problematic locations within the segmentation volume. This also constituted a 'stretch goal' if there was remaining time available. Due to time constraints, the third layer's functionality was not pursued. This section will therefore focus on developmental methodologies of the first two layers.

The flowchart below illustrates the project's developmental components in the big picture. All machine learning processes, which include data pre-processing and validation techniques, were incorporated into a robust Python script.



The blue boxes are aspects that will be addressed here. The remaining boxes will be discussed in sections 10.2 and 10.3.

Data and input vector

As described in Section 9.3, the inputs to supervised learning algorithms are input vectors which consist of input features and output target values. The output metrics of FreeSurfer segmentations from 3695 subjects were formatted into a data table where each row comprised a different subject, and each column comprised a metric extracted from the segmentation volume of that subject. In machine learning terminology, each column was a *feature* to be used for training, and the aggregate of all columns constituted a *feature set*. The target values were binarized values (0s and 1s) stored in a separate data table. The semantic definition of a '0' or '1' label varied depending on the experiment that was performed. The 3695 subjects constituted data from the Framingham Heart Study[9] and the MIT data set. Table 10.1.1 displays an input data tables for five subjects and four sample features. Table 10.1.2 displays the corresponding target values.

	Left-Lateral-Ventricle	Left-Inf-Lat-Vent	Left-Cerebellum-White-Matter	Left-Cerebellum-Cortex
B0	11420	334	12226	45489
B1	9173	366	14669	51042
B2	2089	751	14456	59771
B3	27703	506	13610	61274
B4	12727	568	15316	55417
B5	8664	155	12763	51243

Table 10.1.1. Sample input data table for five subjects. Each row corresponds to the segmentation result of a subject, and each column is the metric extracted from the segmentation result. In ML language, each column constitutes a feature.

B0	0
B1	0
B2	1
B3	0
B4	1
B5	1

Table 10.1.2. Sample binarized data table for five subjects. Each row corresponds to the same subjects as in the rows of Table 10.1.1. The second column outputs a '1' or '0' target value based on some criteria, for

example whether the subject received any edits. A ‘1’ or ‘0’ would constitute ‘received edits’ or ‘did not receive edits’ classes, respectively.

Train data and test data
For every experiment performed, the data set was divided into training and test data sets. The fundamental goal of machine learning is to generalize the learned model *beyond* training data. in order to achieve this, one must be able to project the quality of the model’s pattern generalization for data it was not trained on. However, because future samples have unknown target values, it would not be possible to extrapolate the model’s accuracy from future data. Thus, a common strategy is to use existing labeled data as a proxy for future data. To ensure fairness, these tests must be carried out on data samples that are statistically independent from those used during training – evaluating the model with the same training data is not useful because it would reward models that can “remember” the training data (a consequence known as overfitting), as opposed to generalizing from it.

Data preprocessing
Once the data was split into training and test data sets, the following data preprocessing steps were performed:

1) Data normalization
2) Data balancing
3) Feature selection

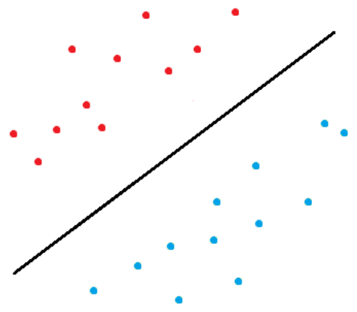
Z-score normalization was used in all cases to scale values across all input features. Data balancing refers to the task of ensuring that the number of positive and negative classes in the training set are equal. This was performed to ensure that the algorithm would not be biased by the weight of one class over the other.

Finally, feature selection was performed prior to training. Feature selection is the process of selecting a subset of the most relevant features for constructing the learned model. This removes features with little impact on making the prediction, allowing for a simpler model, and reducing the possibility of overfitting data. The learned production model would also incur less computational costs.

Jackknifing and cross-validation are additional preprocessing steps involved, but are more closely related to methodology validation, and so are discussed in section 10.2.

Learning
The learning phase involves passing of the pre-processed input features into an algorithm where training and testing are executed. The machine learning libraries used for this project were imported from www.scikit-learn.org/stable, and are written in the Python programming language. The support vector machine (SVM) classifier algorithm was rigorously tested. The input data tables were interfaced with SVM through Python scripts, and the results were written out to a text file for further analyses via bar graphs, line graphs, and ROC curves (ROC is discussed in section 10.2).

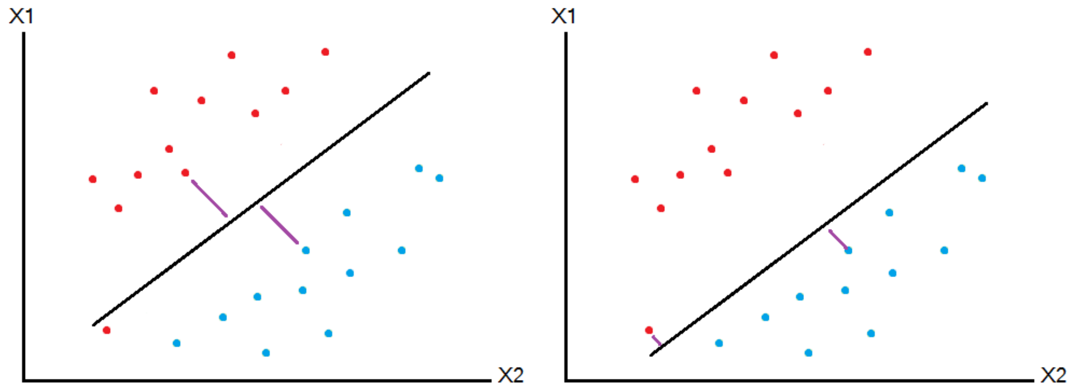
Support Vector Machines are based upon the concept of decision planes that define decision boundaries. A decision plane is a plane that separates objects that contain different class memberships. In the illustration below, each point is an object (or sample) belonging to either the RED or BLUE class. The separating line is known as a *hyperplane* and defines the boundary where any object to the right of the hyperplane is classified as BLUE, and any object to the left is classified as RED. SVM calculates a score for each sample, and sets a threshold value for the hyperplane. Any object with a score below or above that threshold is classified accordingly.



One parameter of SVM that needed to be tuned was the **C** parameter, which determines the distance between the hyperplane and each sample. This distance is known as the lowest minimum margin. Large values of **C** will choose a smaller-margin hyperplane, if that hyperplane achieves the best performance with correctly classifying the training samples. Conversely, smaller values of **C** will choose a larger-margin hyperplane, even if that hyperplane misclassifies more training samples. In other words, smaller values of **C**, can result in more misclassified samples even if the training samples are perfectly linearly separable.

In the ideal scenario, one wants a hyperplane with the largest minimum margin, and a hyperplane that correctly classifies as many samples as possible. In reality, both of these conditions are not always possible, and so choosing the optimal **C** value greatly depends on the distribution of the data set.

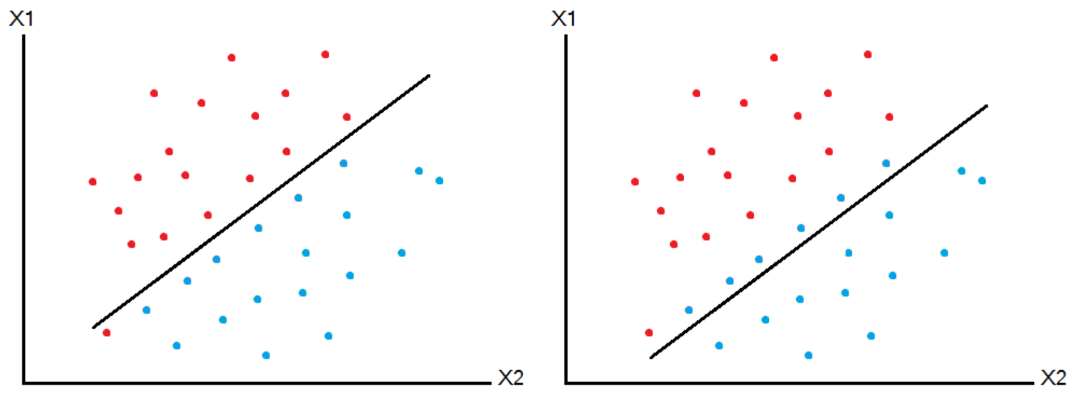
The illustrations below further explain the influence of **C**:



low C large C

In the figures above, each point is one of 24 samples plotted based on its features *X1* and *X2*. The red (*M=12*) and blue dots (*N=12*) represent the type of classification (RED or BLUE). In the figure on the left, a low **C** is chosen, resulting in a large minimum margin (purple lines) but with one outlier (red sample) misclassified. In the figure on the right, a large **C** is chosen. The outlier is not neglected, but results in a smaller minimum margin.

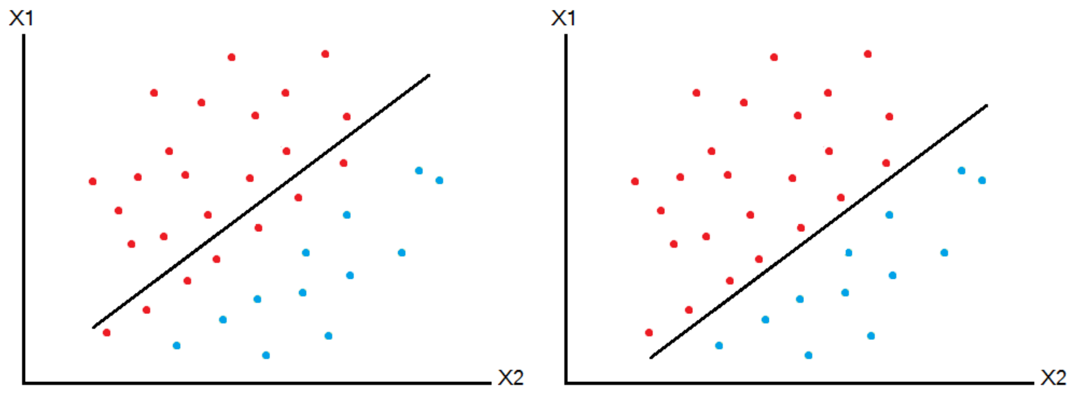
Choosing the optimal **C** depends on the distribution of future data:



Scenario A. low C large C

In scenario A, a low C achieves the best performance, even though it initially resulted in one misclassified sample.

On the other hand, the distribution of future data can also be:



Scenario B. low C large C

In this scenario, the larger C achieves the best performance, despite initially producing a smaller minimum margin.

In the context of this project, this parameter was tuned by looping over a large range of C values, and selecting the value that produced the best results.

The 'standard' experiment consisted of using the default metrics generated automatically by FreeSurfer (note: the definitions of aparc and aseg are given in section 9.3. A sample input feature table was provided in Table X):

a) aseg volume
b) aparc volume

c) aparc thickness
d) aseg mean of white matter pixel intensity
e) aseg standard deviation of white matter pixel intensity

In this vanilla experiment, the target labels were binarized from a post-edit QA data table, and assigned a value based on whether the segmentation volume received any edits at all. The values provided in the QA data table include, for every subject:

a) count - number of control points
b) nWMErase - number of WM voxels erased
c) nWMFill - number of WM voxels filled
d) nBMErase - number of brain mask voxels erased
e) nBMClone - number of brain mask voxels filled
f) nASegChanges - number of ASeg voxels changed
g) lhholes - number of defects in the left hemisphere
h) rhholes - number of defects in the right hemisphere
i) totholes - total number of defects
j) MaskVolToETIV - ratio of mask volume to estimated intracranial volume
k) WMmean - mean intensity in the WM
l) WMstd- standard deviation of intensity in the WM
m) WMmin - min intensity in WM
n) WMmax- max intensity in the WM
o) WMrange- range of intensity in the WM (max-min)
p) WMsnr - mean intensity in the WM/ WMstd

The vanilla experiment summed the values of a-e; any subject with a numeric total greater than 0 received a '1' label. An example of the binarized data table was provided in Table Y.

Many subsequent experiments were performed, and involve experiments varying the input features as well as experiments varying the rules that define a '0' or '1' target label (relabeling). A number of different input features were created (second layer of the project as described) and tested, for example, the gray/white image contrast underlying each segmented structure. By selecting specific values in the QA data table to include or exclude (a-p), a number of different rules defining a '0' or '1' class were applied, and constituted the 'relabeling' experiments.

9.2 Testing, Verification and Validation Implementation

Verification and validation of the machine learning model and results were performed in lock-step with algorithm training and testing. Verifying the reliability of predictions is most concerned with capturing the heterogeneity and variability of real-world unseen data, as well as capturing the variation in the algorithm's predictions. In many scenarios, it is not feasible to collect and re-collect new data to capture this variability. Therefore, two key techniques were applied in order to fully leverage the dataset available: a) jacknifing and b) cross-validation.

Jackknifing was a method applied during the selection of training data to every experimental run[10]. Given a dataset of *N* samples, jackknifing selects an observation at random and adds it to 'jackknife' training set. This process is repeated without replacement, i.e. selected samples are not returned to the original dataset during each sampling cycle, and so the jackknifed dataset cannot contain the same samples multiple times. All remaining data becomes test data applied during the test phase. For this project, each experiment was repeated across 70 permutations of jackknifed data.

Once a jackknifed dataset was created, the dataset was then passed through a cross-validation algorithm to optimize **C** parameter. The algorithm was trained on a range of C values (0 through 5 in increments of 1), with the highest performing C value selected. To validate the selection of the C value, cross validation was applied to each experimental run using a different C value.

During cross validation, the entire training set was divided into *k* subsets, with each *k* subset constituting a 'fold'. *K-1* subsets are used as training data during each fold. Figure 10.2 below illustrates this process for a 5-fold cross validation step, which was used for this project.

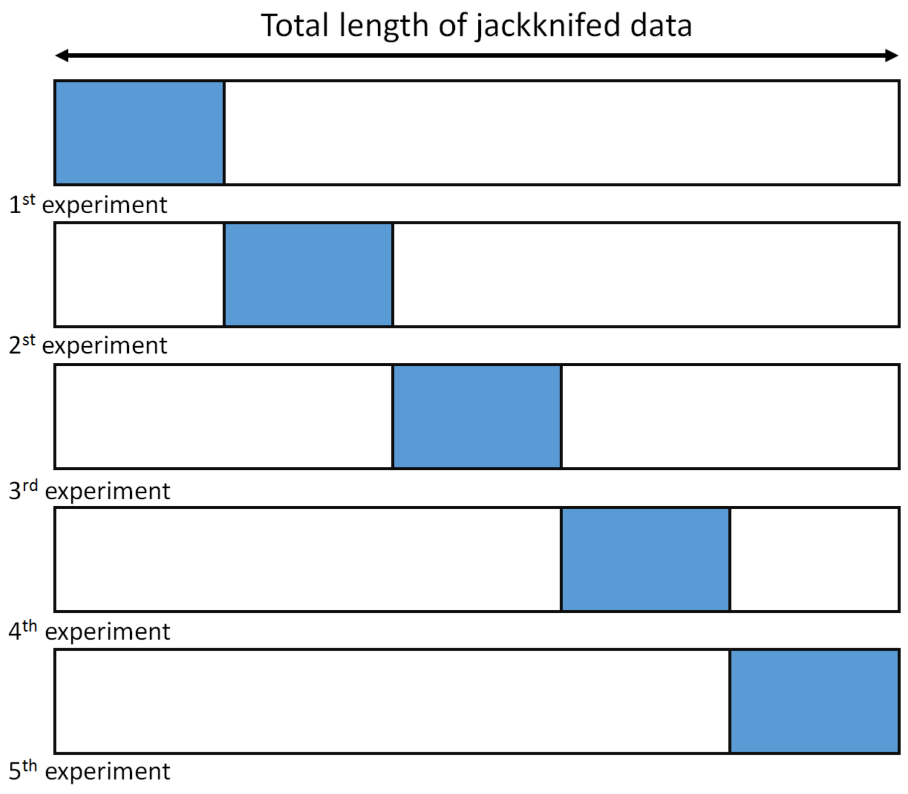


Figure 10.2.1 5-fold cross validation. In 5-fold cross validation, the entire data set is subdivided into 5 sets, with one set held as the test data (blue rectangles). The remaining data is applied as training data. This experiment is repeated *k* times, and the average of the results are used.

This project also drew from concepts in signal detection theory, namely receiver operating characteristics (ROC) graphs. ROC graphs are commonly used in medical decision-making and have been increasingly used in machine learning and data mining research.

Results to the algorithm's predictions were evaluated through accuracy rates, false positive rates (FPR), false negative rates (FNR), false positive discovery rates (FPDR), and false negative discovery rates (FNDR). The goal was to maximize accuracy while minimizing all other rates. FPR and FNR are defined in the truth table below (Figure 10.2.2). The rates are defined by the following formulas:

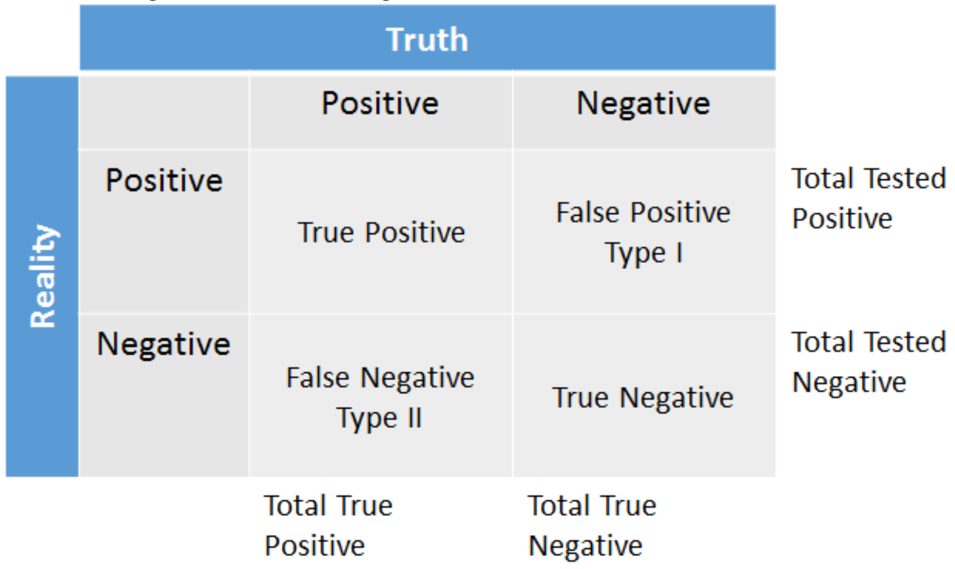
Accuracy rate: # (True Positive + True Negative) / # Positive + Negative

FPR: # False Positive / (# False Positive + # True Negative)

FNR: # False Negative / (# True Positive + # False Negative)

FPDR: # False Positive / # True Positive

FNDR: # False Negative / #True Negative



		Truth		
		Positive	Negative	
Reality	Positive	True Positive	False Positive Type I	Total Tested Positive
	Negative	False Negative Type II	True Negative	Total Tested Negative
		Total True Positive	Total True Negative	

Figure 10.2.2 Truth table

In binary classification problems, FPR and FNR are dependent upon the discrimination threshold imposed by the algorithm to categorize its predictions. More specifically, **scoring** classifiers do not return prediction values (e.g. 1s or 0s), but rather the **probabilities** that a sample belongs to a class. These probabilities fall between 0 and 1, and the discrimination threshold is the cutoff imposed on the predicted probabilities for assigning samples to each class. Consequently, FPR and FNR are affected by this discrimination threshold. However, false positives and false negatives do not have equal consequences. A false positive results in an acceptable segmentation misclassified as “needs edit”. This is less consequential than a false negative, which is an unacceptable segmentation result misclassified as “does not need edit”. Therefore, the effects of different discrimination thresholds were also examined using a signal analysis technique called receiver-operating characteristics (ROC). ROC graphs illustrate the performance of a binary classifier as the discrimination threshold is varied.

ROC graphs are two-dimensional graphs where TPR is plotted on the Y-axis and FPR is plotted on the X-axis[11]. Figure 10.2.2 below describes the properties of ROC space. Each point on the graph represents a classifier, labeled A through D, with an additional classifier labeled as ‘Perfect Classification’.

The lower left point (0,0) represents a threshold in which positive classifications are never issued. Thus, no false positives are incurred, all possible true negatives are detected, but no true positives are gained. The upper right point (1,1) represents the opposite strategy.

The upper left point (0,1) represents a perfect classifier – all possible positives are detected without incurring any false positives. An FPR of 0 also means that all possible true negatives are detected.

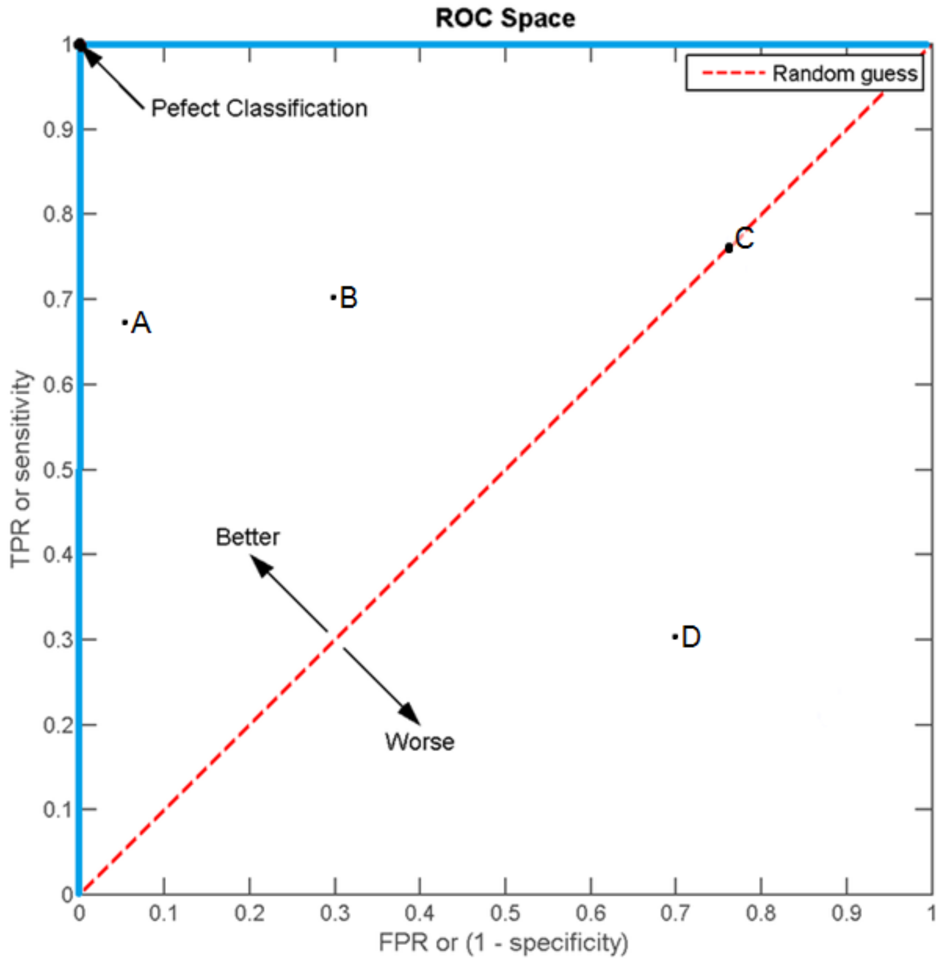


Figure 10.2.3. ROC graph showing five binary classifiers

In the above graph, the dotted red line *y* = *x* represents classifiers that are randomly guessing a class. A classifier randomly guessing the positive class half the time can be expected to yield half the positives and half the negatives correct. This is represented at the point (0.5, 05). As the frequency of positive predictions increases, that classifiers likelihood of incurring true positives and false positives increases proportionally. Point C represents a classifier that guesses the positive class about 78% of the time, thus also incurring false positives about 78% of the time (0.78, 0.78). In order to deviate away from the diagonal line, classifiers must be able to exploit information in the training data. Suppose that Point B is located at (0.3, 0.7) and Point D is located at (0.7, 0.3). represents a classifier that performs far below

For any classifier, an ROC curve can be traced by varying the discrimination threshold from -∞ to ∞. If the classifier output for a given sample is above the threshold, the classifier outputs a *Y*, else an *N*. An example of this curve is shown below in Figure 10.2.3.

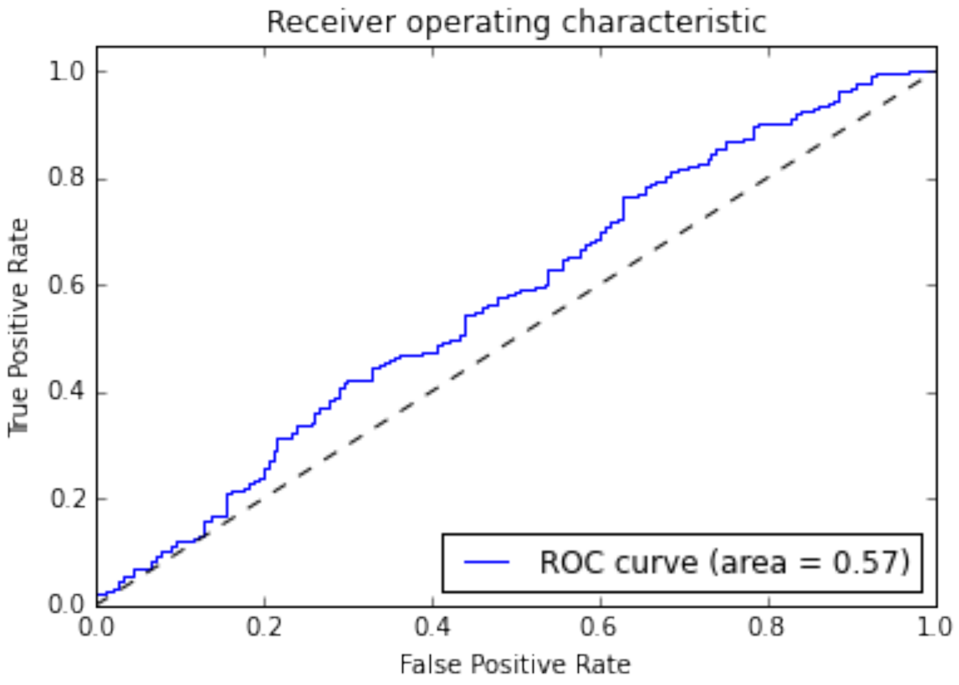


Figure 10.2.4. ROC curve from an experimental run using subcortical volumes as input features.

Note that ROC curves generated from a finite set of samples are step functions, but approach true curves as the number of samples approaches infinity.

To quantify performance, the area under the curve (AUC) is computed and applied as a scalar value to represent the classifier's performance. In a unit square, the maximum area is 1. Randomly guessing classifiers produce the diagonal line $y = x$, and so have areas of 0.5. Thus, a useful classifier should have an AUC as close to 1 but no less than 0.5. In the above figure, an AUC of 0.57 indicates mediocre performance.

To summarize jackknifing, cross validation, and ROC curves were techniques applied at every experimental run to validate results, account for statistical variance, and to quantify classifier performance.

Steps were also taken to thoroughly understand classifier behavior. Synthetic data generators were constructed to validate code, as well as to validate expectations of how the classifier should behave. For instance, classifiers should be able to predict 'perfect' data sets with perfect accuracy. Predictions on ‘chance’ data sets should not result in better or worse than 50% accuracy. Any deviations from these expectations would have indicated that the algorithms were behaving differently than expected, or that there were bugs in the code or errors in the way the algorithms were implemented.

The code snippet below is an example of how ‘perfect’ data sets were simulated in Python.

```python
import numpy as np

nsubjects = 1038;
nfeatures = 55;

features = np.random.rand(nsubjects,nfeatures);
labels = (np.mean(features,1) > .48).astype(int);

indPos = np.where(labels > 0)[0]
indNeg = np.where(labels > 0)[0]

offset = 0.2
features[labels > 0,:] = features[labels > 0,:] + offset
```

Synthetic data sets were also used to validate feature selection implementation by appending ‘irrelevant’ features to perfect data sets. In this scenario, the irrelevant features were expected to be eliminated as not useful.

9.3 Results and Technical Conclusion

In total, 50 experimental runs comprising various combinations of input features and labeling methodologies were performed (see Table 10.3.1). Each experiment was repeated 70 times using jackknifed data samples.

The ‘standard’ experiments comprised of using the input features listed in Section 10.1, i.e. the default metrics automatically generated by FreeSurfer. The labels were created by summing numeric values in the QA data table (items a-e also listed in Section 10.1). This experiment was performed 70 times for both the FHS and MIT data sets. Results for the FHS data set are shown in following figures.

	Category	Set	Type	Experiment
1	Default	FHS	Standard	Standard
2	Default	MIT	Standard	Standard
4	Relabeling	FHS	edit vol	# edits > 25 vox == 1
5	Relabeling	MIT	edit vol	# edits > 25 vox == 1
6	Relabeling	FHS	edit vol	# edits > 50 vox == 1
7	Relabeling	MIT	edit vol	# edits > 50 vox == 1
8	Relabeling	FHS	edit vol	# edits > 75 vox == 1
9	Relabeling	MIT	edit vol	# edits > 75 vox == 1
10	Relabeling	FHS	edit vol	# edits > 100 vox == 1
11	Relabeling	MIT	edit vol	# edits > 100 vox == 1
12	Relabeling	FHS	edit type	GM e > 0 == 1
13	Relabeling	MIT	edit type	GM e > 0 == 1
14	Relabeling	FHS	edit type	WM e > 0 == 1
15	Relabeling	MIT	edit type	WM e > 0 == 1
16	Relabeling	FHS	edit type	control points > 0
17	Relabeling	MIT	edit type	control points > 0
18	Relabeling	FHS	post-pre	Δ L Hippo vol > 10 vox
19	Relabeling	MIT	post-pre	Δ L Hippo vol > 10 vox
20	Relabeling	FHS	post-pre	Δ R Hippo vol > 10 vox
21	Relabeling	MIT	post-pre	Δ R Hippo vol > 10 vox
22	Relabeling	FHS	post-pre	Δ L CerebellumWM vol > .05(orig)
23	Relabeling	MIT	post-pre	Δ L CerebellumWM vol > .05(orig)
24	Relabeling	FHS	post-pre	Δ R CerebellumWM vol > .05(orig)
25	Relabeling	MIT	post-pre	Δ R CerebellumWM vol > .05(orig)
26	Relabeling	FHS	post-pre	Δ L amygdala > .05(orig)
27	Relabeling	MIT	post-pre	Δ L amygdala > .05(orig)
28	Relabeling	FHS	post-pre	Δ R amygdala > .05(orig)
29	Relabeling	MIT	post-pre	Δ R amygdala > .05(orig)
30	newFeats	FHS		Δ Laterality
31	newFeats	MIT		Δ Laterality
32	newFeats	FHS	contrast	G/W subcortical contrast
33	newFeats	MIT	contrast	G/W subcortical contrast
34	newFeats	FHS	contrast	G/W cortical contrast
35	newFeats	MIT	contrast	G/W cortical contrast
36	newFeats	FHS	contrast	G/W cortical contrast histogram
37	newFeats	MIT	contrast	G/W cortical contrast histogram
38	newFeats	FHS	contrast	cortical + subcortical histogram
39	newFeats	MIT	contrast	cortical + subcortical histogram
40	newFeats	FHS		WM parcellation
41	newFeats	MIT		WM parcellation
42	newFeats	FHS		normalized to ICV
43	newFeats	MIT		normalized to ICV
44	newFeats	FHS		preedit QA
45	newFeats	MIT		preedit QA
46	feat/label	FHS		WM feats/WM e > 0 ==1
47	feat/label	MIT		WM feats/WM e > 0 ==1
48	feat/label	FHS		GM feats/GM e > 0 == 1
49	feat/label	MIT		GM feats/GM e > 0 == 1
50	feat/label	FHS		WM feats/WM e > 0 ==1 seed 10
51	feat/label	MIT		WM feats/WM e > 0 ==1 seed 10

Table 10.3.1. List of types of experiments performed. Each experiment was repeated 70 times using different jackknifed datasets.

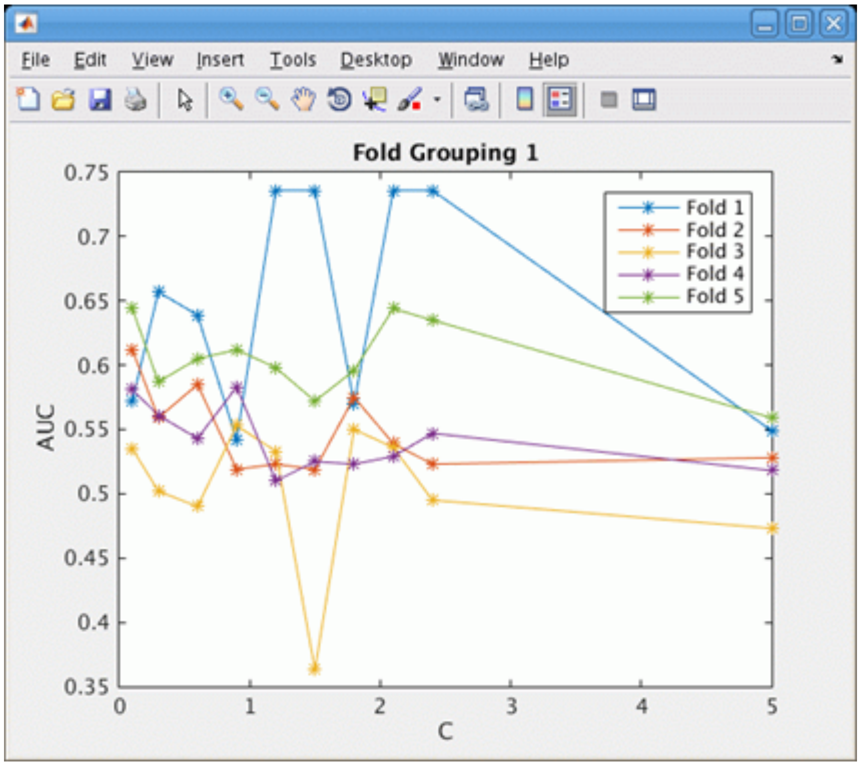


Figure 10.3.1

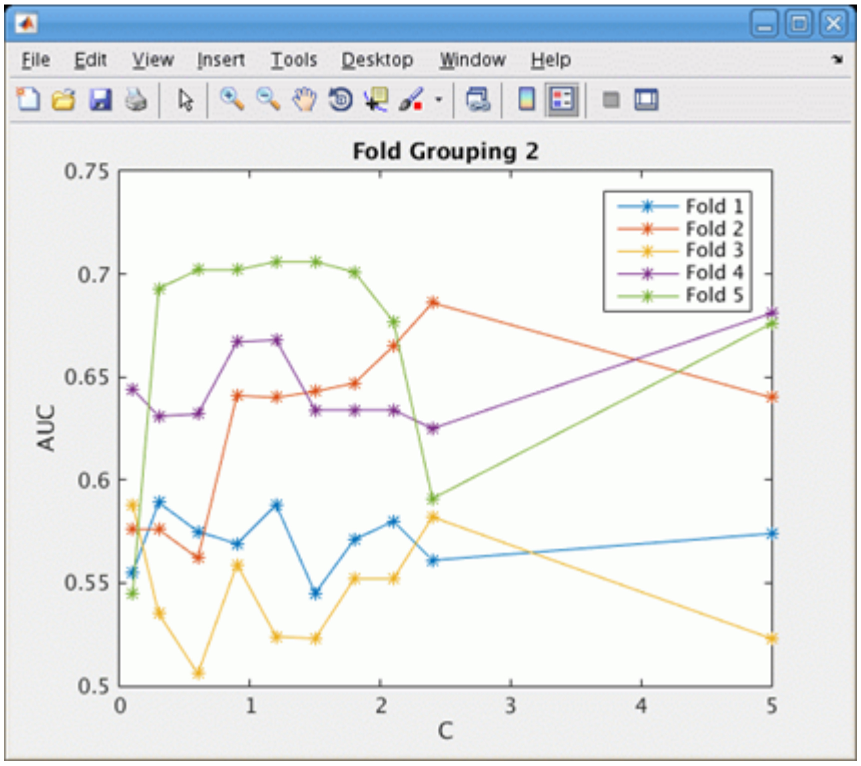


Figure 10.3.2

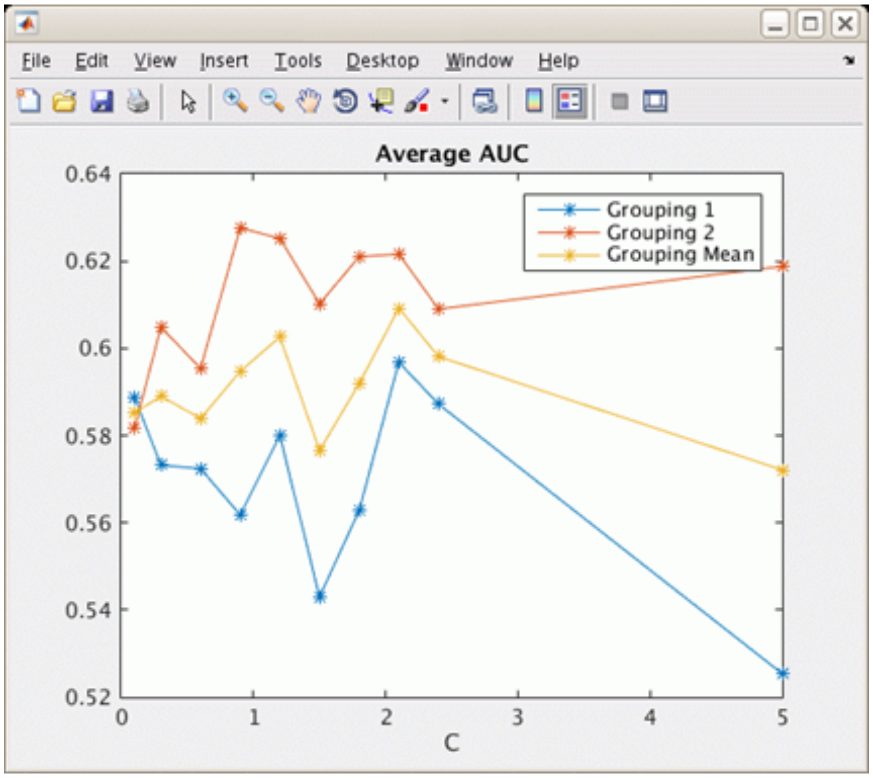


Figure 10.3.3

Figures 10.3.1 and 10.3.2 graphed the variation of AUC as the SVM C parameter was varied from 0 to 5. Figure 10.3.3 graphed the averages of these AUCs. Each line represented a fold within the 5-fold cross validation methodology described earlier. Ideally, AUC would have increased towards some maximum value as it approached an optimal C value, then decrease as it moved away from that C. Because this pattern was not observed, it was concluded that the C parameter had extremely little influence on prediction accuracy. All subsequent experiments were defaulted to C = 1.

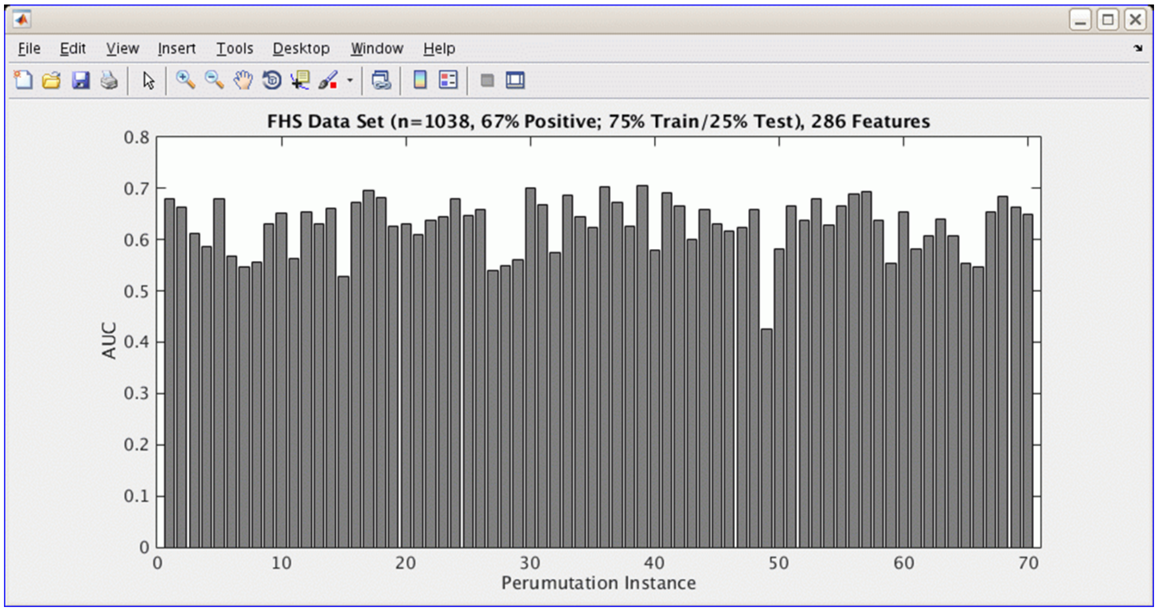


Figure 10.3.4

Figure 10.3.4 graphed the variation of AUC over 70 permutations. A unimodal distribution was expected around some value; in this case, it was roughly around 0.55, with one outlier observed (around Permutation 50).

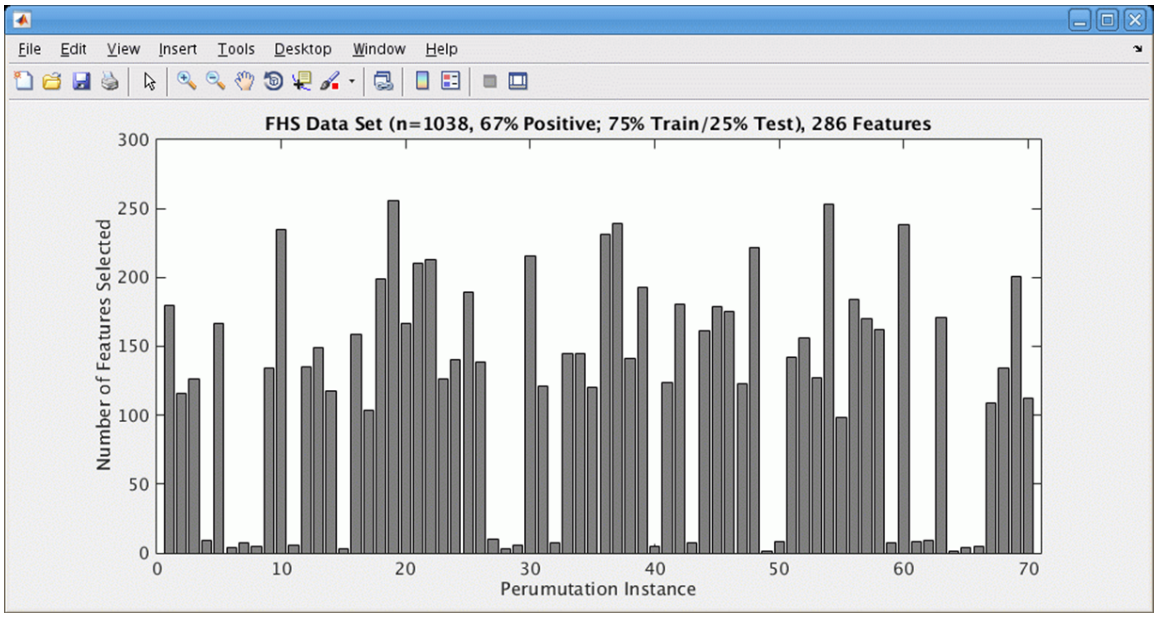


Figure 10.3.5

Figure 10.3.5 graphed the variation in the number of features selected across permutations. As evidenced here, no consistent pattern was observed, with significant variation across permutations. One hypothesis was that the data was too heterogeneous for any set of features to be consistently chosen. The selected feature sets were investigated further for all experiments. At the time of this writing, the analyses were inconclusive.

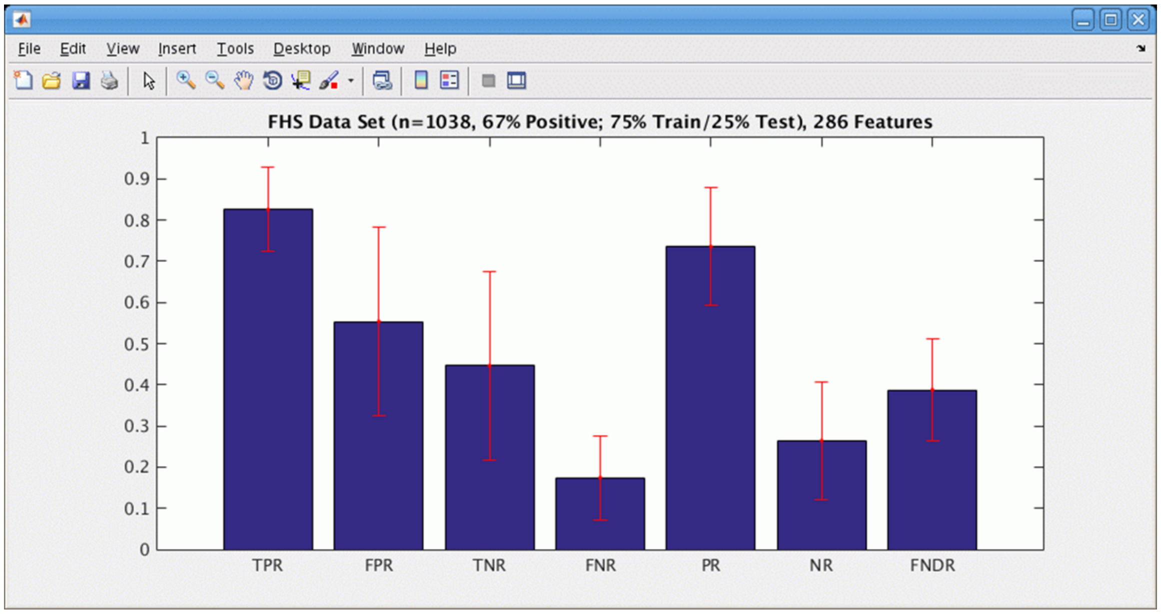


Figure 10.3.6

Rates for different types of errors were also examined (Figure 10.3.6) in every experiment. Generally, FNDR was desired to be as low as possible because FNDR indicates the percentage of predicted negatives that were incorrect predictions (i.e should have been labeled positive). In this graph, the FNDR was 0.4. What this means for the user is that approximately 40% of the declared negatives were in fact positives. As explained in an earlier section, false negative predictions are more detrimental than false positive predictions.

Of the 50 types of experiments performed, the most promising results came from a) generating labels based only on WM edits, and b) using contrast values as input features. The results are shown in following figures.

	Category	**Set**	**Type**	**Experiment**
1	Default	FHS	Standard	Standard
2	Default	MIT	Standard	Standard
3	Relabeling	FHS	edit vol	# edits > 25 vox == 1
4	Relabeling	MIT	edit vol	# edits > 25 vox == 1
5	Relabeling	FHS	edit vol	# edits > 50 vox == 1
6	Relabeling	MIT	edit vol	# edits > 50 vox == 1
7	Relabeling	FHS	edit vol	# edits > 75 vox == 1
8	Relabeling	MIT	edit vol	# edits > 75 vox == 1
9	Relabeling	FHS	edit vol	# edits > 100 vox == 1
10	Relabeling	MIT	edit vol	# edits > 100 vox == 1
11	Relabeling	FHS	edit type	GM e > 0 == 1

12	Relabeling	MIT	edit type	GM e > 0 == 1
13	Relabeling	FHS	edit type	WM e > 0 == 1
14	Relabeling	MIT	edit type	WM e > 0 == 1
15	Relabeling	FHS	edit type	control points > 0
16	Relabeling	MIT	edit type	control points > 0
17	Relabeling	FHS	post-pre	Δ L Hippo vol > 10 vox
18	Relabeling	MIT	post-pre	Δ L Hippo vol > 10 vox
19	Relabeling	FHS	post-pre	Δ R Hippo vol > 10 vox
20	Relabeling	MIT	post-pre	Δ R Hippo vol > 10 vox
21	Relabeling	FHS	post-pre	Δ L CerebellumWM vol > .05(orig)
22	Relabeling	MIT	post-pre	Δ L CerebellumWM vol > .05(orig)
23	Relabeling	FHS	post-pre	Δ R CerebellumWM vol > .05(orig)
24	Relabeling	MIT	post-pre	Δ R CerebellumWM vol > .05(orig)
25	Relabeling	FHS	post-pre	Δ L amygdala > .05(orig)
26	Relabeling	MIT	post-pre	Δ L amygdala > .05(orig)
27	Relabeling	FHS	post-pre	Δ R amygdala > .05(orig)
28	Relabeling	MIT	post-pre	Δ R amygdala > .05(orig)
29	newFeats	FHS		Δ Laterality
30	newFeats	MIT		Δ Laterality
31	newFeats	FHS	contrast	G/W subcortical contrast
32	newFeats	MIT	contrast	G/W subcortical contrast
33	newFeats	FHS	contrast	G/W cortical contrast
34	newFeats	MIT	contrast	G/W cortical contrast
35	newFeats	FHS	contrast	G/W cortical contrast histogram
36	newFeats	MIT	contrast	G/W cortical contrast histogram
37	newFeats	FHS	contrast	cortical + subcortical histogram
38	newFeats	MIT	contrast	cortical + subcortical histogram
39	newFeats	FHS		WM parcellation
40	newFeats	MIT		WM parcellation
41	newFeats	FHS		normalized to ICV
42	newFeats	MIT		normalized to ICV
43	newFeats	FHS		preedit QA
44	newFeats	MIT		preedit QA
45	feat/label	FHS		WM feats/WM e > 0 ==1
46	feat/label	MIT		WM feats/WM e > 0 ==1
47	feat/label	FHS		GM feats/GM e > 0 == 1
48	feat/label	MIT		GM feats/GM e > 0 == 1
49	feat/label	FHS		WM feats/WM e > 0 ==1 seed 10
50	feat/label	MIT		WM feats/WM e > 0 ==1 seed 10

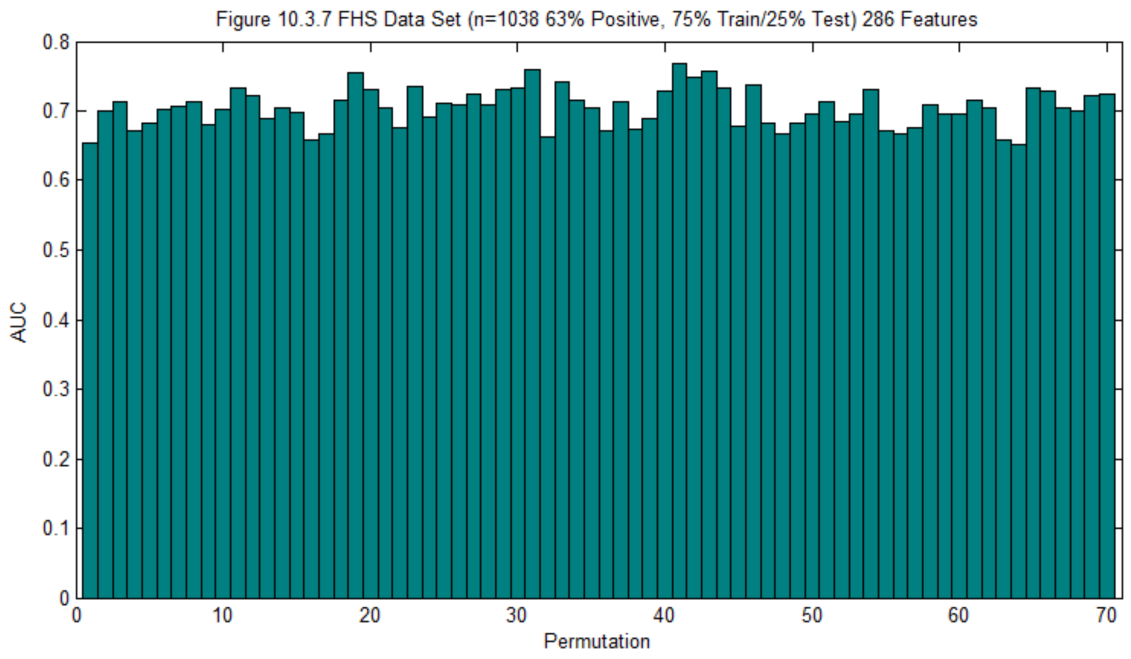


Figure 10.3.7

In the above figure, the labels were created using only WM QA edit information. The AUCs over 70 permutations is quite respectablely distributed between 0.6 and 0.7.

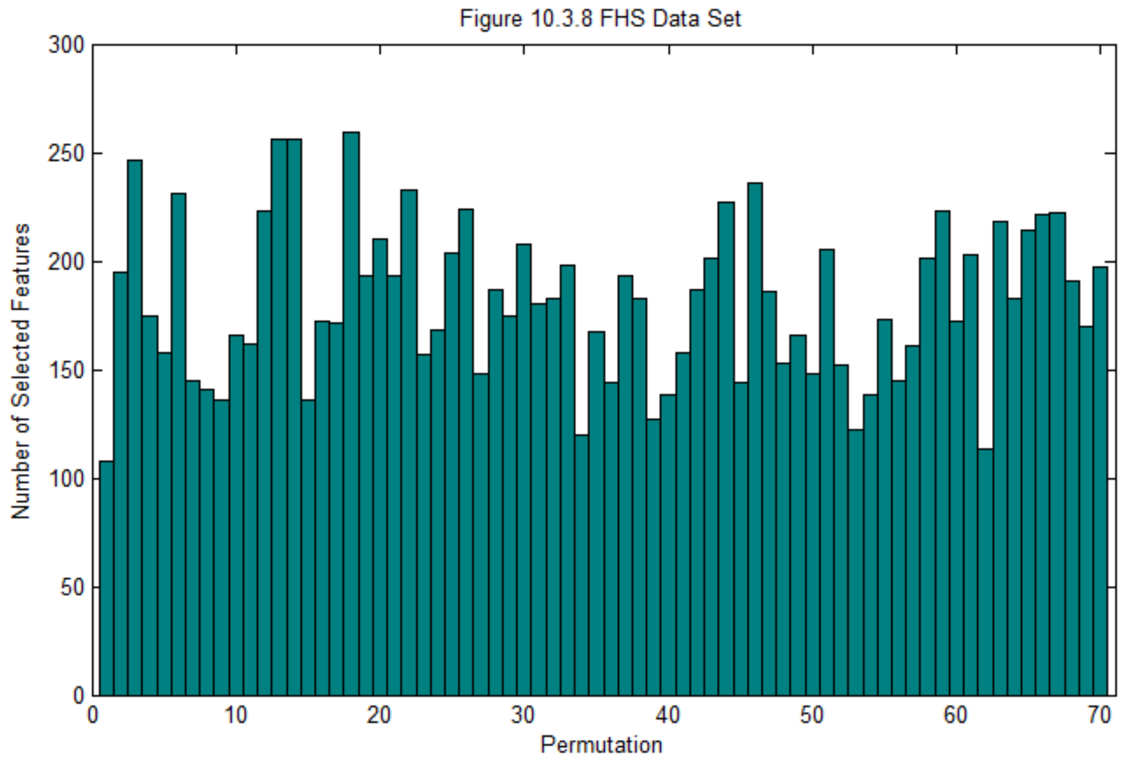


Figure 10.3.8

However, as seen in Figure 10.3.8 above, there remains considerable variation in the number of features selected.

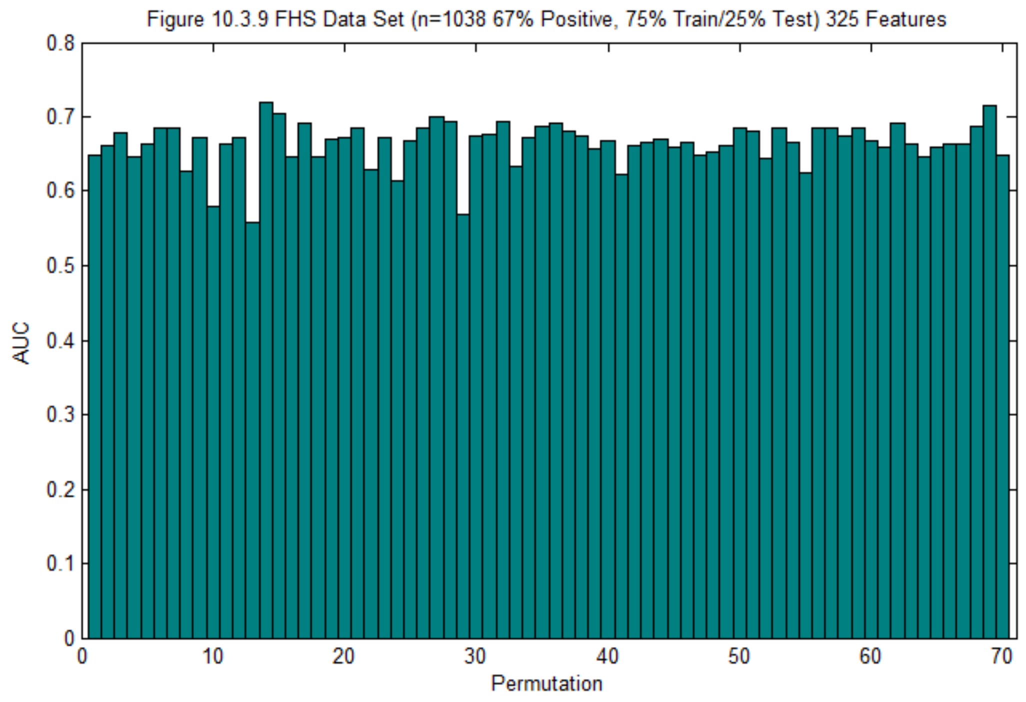


Figure 10.3.9. Cortical and subcortical contrast values as features

In Figure 10.3.9 above, the input features were cortical and subcortical contrast values. Again, the AUCs are respectably distributed around 0.65.

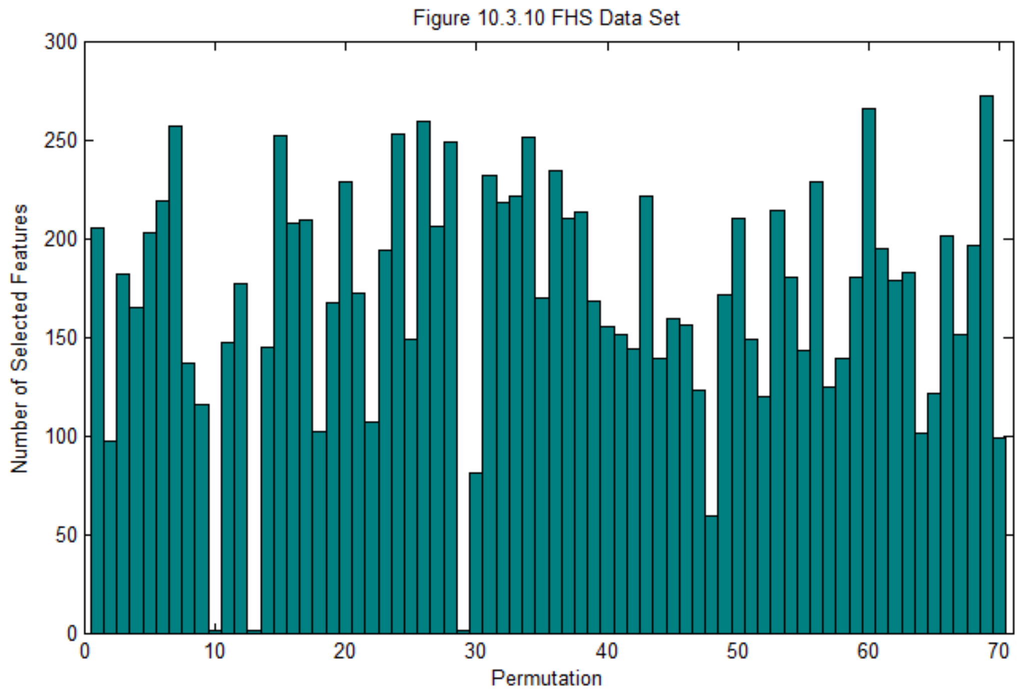


Figure 10.3.10

As in the previous case, the variation in number of selected features is significant.

In summary, there was significant progress made in elucidating what information best provide machine-learning algorithms with the data to make the desired predictions. From a proof of concept standpoint, this project successfully demonstrated that machine learning is a viable approach to performing segmentation QA. Nevertheless, understanding the variation in the feature selection process is the next critical step before the tool can be confidently deployed to users.

9.4 Scientific and Technical Challenges

The most significant challenges were in learning and thoroughly understanding the various machine learning concepts, and in developing the programming skillset in Python to reliably apply algorithms to data. The extent of programming was not limited to writing the QA tool – extensive programming was needed to write supplementary tools to perform meta-analysis; this portion was perhaps the least anticipated.

Some of the major machine learning concepts that needed to be understood were:

a) Supervised vs unsupervised machine learning
    a. Algorithms – support vector machines, random forests, Naïve Bayes
b) Feature selection algorithms (e.g. recursive feature selection, Lasso)
c) Regression vs classification
d) Validation methodologies – bootstrapping, jackknifing, cross validation

Additional critical components to the project were in evaluating results, making inferences, and adjusting experiments and/or devising new experiments based on drawn conclusions. These processes involved high levels of critical thinking, which added to the scientific and technical challenges. For instance, considerable time was spent understanding how *scikit* tools performed feature selection, computed scores, and assigned classes 'under the hood'. Discussion articles, forums, and documentations were researched to gain better understanding, but in many cases, there were no clear and concise conclusions on how some of the tools functioned. Therefore, making the right assumptions and decisions based on what was understood was also a technical challenge.

10 PROJECT PLAN

10.1 Statement of work

The purpose of the project was to develop a quantitative method to FreeSurfer segmentation quality assessment. The approach was to implement existing machine learning algorithms and refine the features used to make the classification. The goal was a QA tool that indicated to users whether manual editing of a FreeSurfer segmentation was needed. The algorithms and software tools available at http://www.scikit-learn.org/stable were tested and built upon to QA segmented volumes. The algorithm in particular that was used was support vector machines.

Training data comprised approximately 3700 pre-inspected segmented volumes containing editing information provided by manual user QA.

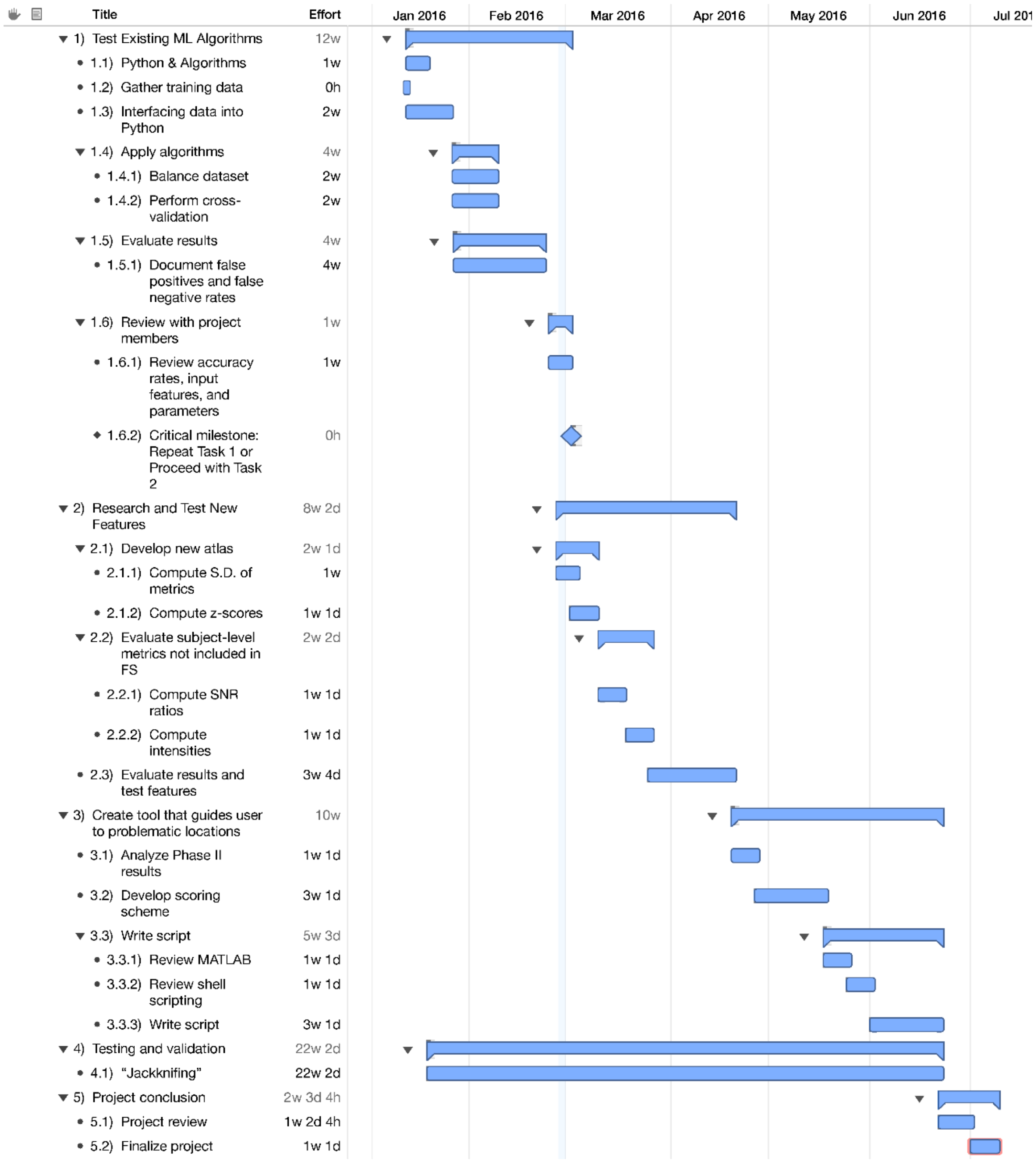
10.2 Schedule

10.2.1 Work Breakdown Structure

WBS	Task Name	Outcome	WBS Predecessors
1	**Phase I: Test existing ML algorithms**	Select and test algorithm(s) best suited for making classification	
1.1	Python & Algorithms	Familiarize self with Python language, machine learning concepts	
1.2	Gather training data	Set of features that will be used to train ML algorithm	1.1
1.3	Interfacing data into Python	Modify algorithm so that data can be used as input	1.2
1.4	Apply algorithms	Test algorithms	1.1, 1.2, 1.3
1.4.1	Balance dataset	Removes weighting of classes	
1.4.2	Implement jackknifing	Accounts for variance in prediction accuracy	
1.4.3	Implement cross-validation	Model validation	
1.4.4	Implement feature selection	Removes irrelevant features	
1.4.5	Implement C optimization	Optimize SVM parameters	
1.5	Evaluate results	Assess results of tested algorithm	1.1, 1.2, 1.3, 1.4
1.5.1	Implement ROC analysis	Measure classifier performance	
1.5.2	FPR, FNR, FPDR, FNDR, TPR, TNR	Measure error rates	1.5.1
1.5.3	Write script to evaluate selected features	Validate feature selection	
1.5.4	Write synthetic data generator	Validate algorithm behavior expectations, bugs in code	
1.6	Conclusions	Discuss results and plan next course of action	
1.6.1	Critical Milestone: Repeat Task 1 or Proceed to Task 2		
2	**Phase II: Develop New Features, Test relabeling**	Develop features that better predict output class; relabel to analyze which types of predictions are best	1
2.1	Obtain post edit data	Test post-pre differences as features	
2.2	Obtain contrast data	Test contrast information as features	
2.3	Obtain WM parcellation data	Test parcellation metrics as features	
2.4	Laterality differences	Test laterality differences as features	
2.5	Normalized to ICV	Test normalized values to ICV as features	
2.6	Relabeling tests	Test better ways to form labels	
2.7	SNR	Test SNR as features	
2.8	Mask/ICV ratios	Test mask/ICV ratio as features	
2.9	Conclusions		2.1, 2.2
2.10	Critical milestone: repeat Task 2 or proceed to Task 3		
3	**Phase III: Create tool that guides user to problematic locations**	Develop tool that tells user where to edit in segmentation volume	
3.1	Analyze Phase II results	Select metrics most useful to achieve goal	2
3.2	Develop “scoring” scheme	Express metric at each vertex	2

3.3	Write script		2, 3.1, 3.2
3.3.1	Review MATLAB		
3.3.2	Review shell scripting		
3.3.3	Write script		
4	**Testing and Validation**	Refer to 1.4.2	
	Jackknifing		
	Cross validation		
	Feature selection		
5	**Project Conclusion**		
5.1	Review work		
5.2	Finalize project	Prepare for presentation	

10.2.2 Gantt chart

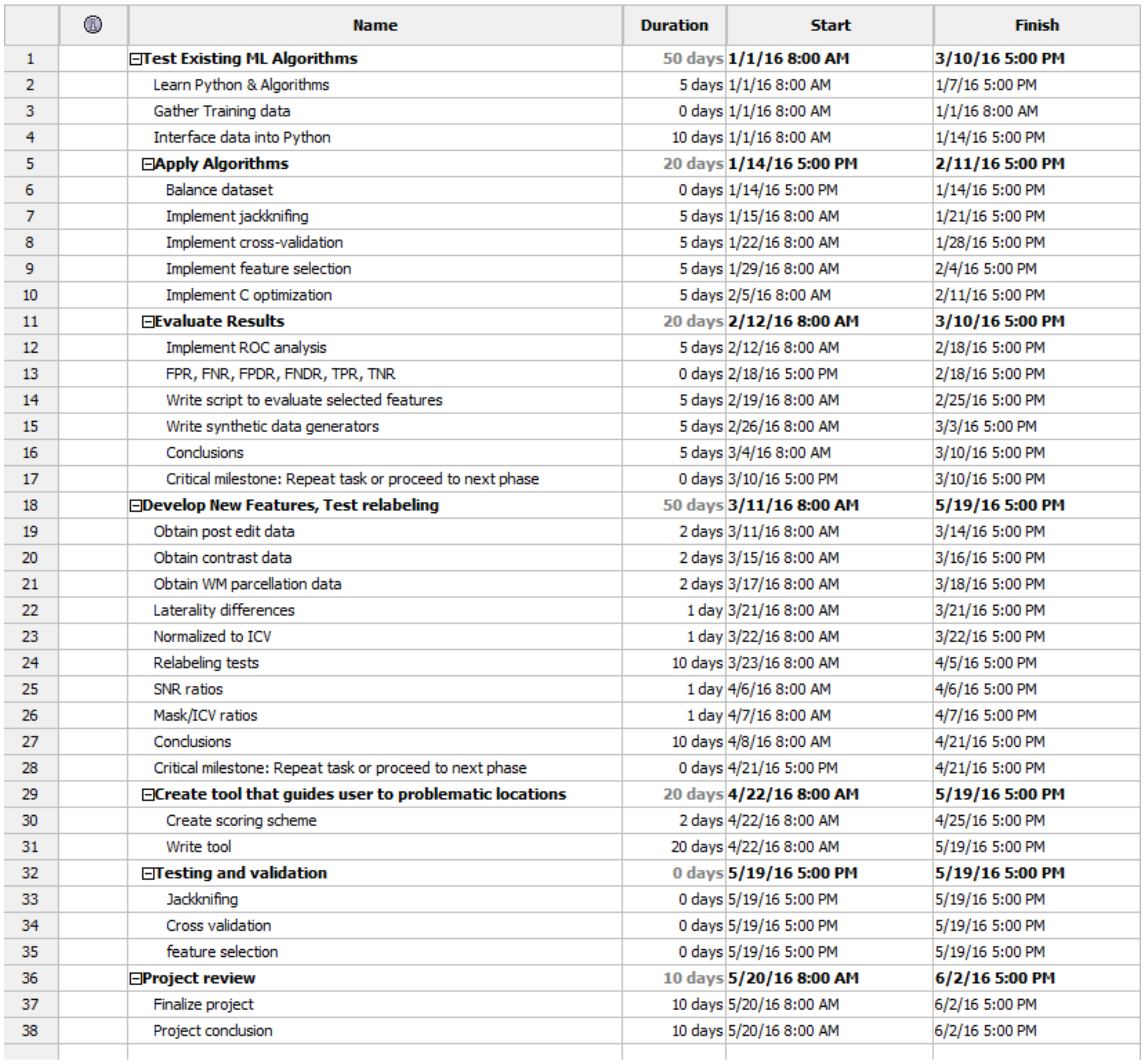


Title	Effort
1) Test Existing ML Algorithms	12w
1.1) Python & Algorithms	1w
1.2) Gather training data	0h
1.3) Interfacing data into Python	2w
1.4) Apply algorithms	4w
1.4.1) Balance dataset	2w
1.4.2) Perform cross-validation	2w
1.5) Evaluate results	4w
1.5.1) Document false positives and false negative rates	4w
1.6) Review with project members	1w
1.6.1) Review accuracy rates, input features, and parameters	1w
1.6.2) Critical milestone: Repeat Task 1 or Proceed with Task 2	0h
2) Research and Test New Features	8w 2d
2.1) Develop new atlas	2w 1d
2.1.1) Compute S.D. of metrics	1w
2.1.2) Compute z-scores	1w 1d
2.2) Evaluate subject-level metrics not included in FS	2w 2d
2.2.1) Compute SNR ratios	1w 1d
2.2.2) Compute intensities	1w 1d
2.3) Evaluate results and test features	3w 4d
3) Create tool that guides user to problematic locations	10w
3.1) Analyze Phase II results	1w 1d
3.2) Develop scoring scheme	3w 1d
3.3) Write script	5w 3d
3.3.1) Review MATLAB	1w 1d
3.3.2) Review shell scripting	1w 1d
3.3.3) Write script	3w 1d
4) Testing and validation	22w 2d
4.1) "Jackknifing"	22w 2d
5) Project conclusion	2w 3d 4h
5.1) Project review	1w 2d 4h
5.2) Finalize project	1w 1d

10.2.3 Project Planning Assessment

The project moved slower than scheduled, partly due to unanticipated additional tasks and partly due to technical challenges that delayed scheduled start and end times. In addition, the candidate did not most effectively influence and advocate for needs concerning technical guidance and overall project vision. Towards the second half of the project, a better rhythm was established that pushed progress at a faster rate, but nevertheless the previous delays as well as lingering technical complexities remained troublesome. Delays were dealt with by expediting some tasks, but in most cases, deadlines were merely pushed back. Looking back, this was not the most efficient approach.

A revised and more accurate WBS is shown below



		Name	Duration	Start	Finish
1		**⊟Test Existing ML Algorithms**	50 days	**1/1/16 8:00 AM**	**3/10/16 5:00 PM**
2		Learn Python & Algorithms	5 days	1/1/16 8:00 AM	1/7/16 5:00 PM
3		Gather Training data	0 days	1/1/16 8:00 AM	1/1/16 8:00 AM
4		Interface data into Python	10 days	1/1/16 8:00 AM	1/14/16 5:00 PM
5		**⊟Apply Algorithms**	20 days	**1/14/16 5:00 PM**	**2/11/16 5:00 PM**
6		Balance dataset	0 days	1/14/16 5:00 PM	1/14/16 5:00 PM
7		Implement jackknifing	5 days	1/15/16 8:00 AM	1/21/16 5:00 PM
8		Implement cross-validation	5 days	1/22/16 8:00 AM	1/28/16 5:00 PM
9		Implement feature selection	5 days	1/29/16 8:00 AM	2/4/16 5:00 PM
10		Implement C optimization	5 days	2/5/16 8:00 AM	2/11/16 5:00 PM
11		**⊟Evaluate Results**	20 days	**2/12/16 8:00 AM**	**3/10/16 5:00 PM**
12		Implement ROC analysis	5 days	2/12/16 8:00 AM	2/18/16 5:00 PM
13		FPR, FNR, FPDR, FNDR, TPR, TNR	0 days	2/18/16 5:00 PM	2/18/16 5:00 PM
14		Write script to evaluate selected features	5 days	2/19/16 8:00 AM	2/25/16 5:00 PM
15		Write synthetic data generators	5 days	2/26/16 8:00 AM	3/3/16 5:00 PM
16		Conclusions	5 days	3/4/16 8:00 AM	3/10/16 5:00 PM
17		Critical milestone: Repeat task or proceed to next phase	0 days	3/10/16 5:00 PM	3/10/16 5:00 PM
18		**⊟Develop New Features, Test relabeling**	50 days	**3/11/16 8:00 AM**	**5/19/16 5:00 PM**
19		Obtain post edit data	2 days	3/11/16 8:00 AM	3/14/16 5:00 PM
20		Obtain contrast data	2 days	3/15/16 8:00 AM	3/16/16 5:00 PM
21		Obtain WM parcellation data	2 days	3/17/16 8:00 AM	3/18/16 5:00 PM
22		Laterality differences	1 day	3/21/16 8:00 AM	3/21/16 5:00 PM
23		Normalized to ICV	1 day	3/22/16 8:00 AM	3/22/16 5:00 PM
24		Relabeling tests	10 days	3/23/16 8:00 AM	4/5/16 5:00 PM
25		SNR ratios	1 day	4/6/16 8:00 AM	4/6/16 5:00 PM
26		Mask/ICV ratios	1 day	4/7/16 8:00 AM	4/7/16 5:00 PM
27		Conclusions	10 days	4/8/16 8:00 AM	4/21/16 5:00 PM
28		Critical milestone: Repeat task or proceed to next phase	0 days	4/21/16 5:00 PM	4/21/16 5:00 PM
29		**⊟Create tool that guides user to problematic locations**	20 days	**4/22/16 8:00 AM**	**5/19/16 5:00 PM**
30		Create scoring scheme	2 days	4/22/16 8:00 AM	4/25/16 5:00 PM
31		Write tool	20 days	4/22/16 8:00 AM	5/19/16 5:00 PM
32		**⊟Testing and validation**	0 days	**5/19/16 5:00 PM**	**5/19/16 5:00 PM**
33		Jackknifing	0 days	5/19/16 5:00 PM	5/19/16 5:00 PM
34		Cross validation	0 days	5/19/16 5:00 PM	5/19/16 5:00 PM
35		feature selection	0 days	5/19/16 5:00 PM	5/19/16 5:00 PM
36		**⊟Project review**	10 days	**5/20/16 8:00 AM**	**6/2/16 5:00 PM**
37		Finalize project	10 days	5/20/16 8:00 AM	6/2/16 5:00 PM
38		Project conclusion	10 days	5/20/16 8:00 AM	6/2/16 5:00 PM

Some tasks that were not anticipated were implementing ROC analysis, implementing cross-validation, feature selection, as well as writing additional scripts to analyze the feature selection process.

Bugs in the code were occasionally problematic, sometimes delaying the completion of tasks by days or weeks.

10.3 Budget and Costs Assessment

Material costs – N/A – all data, software, and tools were freely available and/or already acquired

Labor – N/A – this is taken on as an independent unpaid project

10.4 Risk Plan and Mitigation Assessment

The biggest risk was in failing to identify a machine-learning algorithm that could sufficiently make the desired classification. Reaching this conclusion depended heavily on whether the team had confidently determined that all reasonable features and methods of relabeling were tested and assessed.

Weekly meetings were scheduled to evaluate progress and results. In these meetings, the results from the validation steps as well as prediction results were evaluated to determine the best course of action.

The most important milestone was establishing whether new features needed to be created, and if so, what types. A second critical milestone was establishing whether algorithms should be explored, and if so, which ones. These milestones were necessary for the project team to conclude whether machine learning was a viable approach to performing segmentation QA. Deadlines for these milestones were scheduled in the WBS.

As the project progressed, the focus shifted towards the first milestone, and so other algorithms were not explored.

Task 3 in the WBS carried the least importance, so whenever additional time was needed, it was taken from Task 3.

11 LEADERSHIP

11.1 Leadership Capabilities Assessment

Because this project was sought out and requested independently, it was my sole responsibility to drive the project. I needed to effectively coordinate with the project team in order to leverage the support I needed to successfully complete the project.

The specific capabilities and skills that were identified as development opportunities were the following:

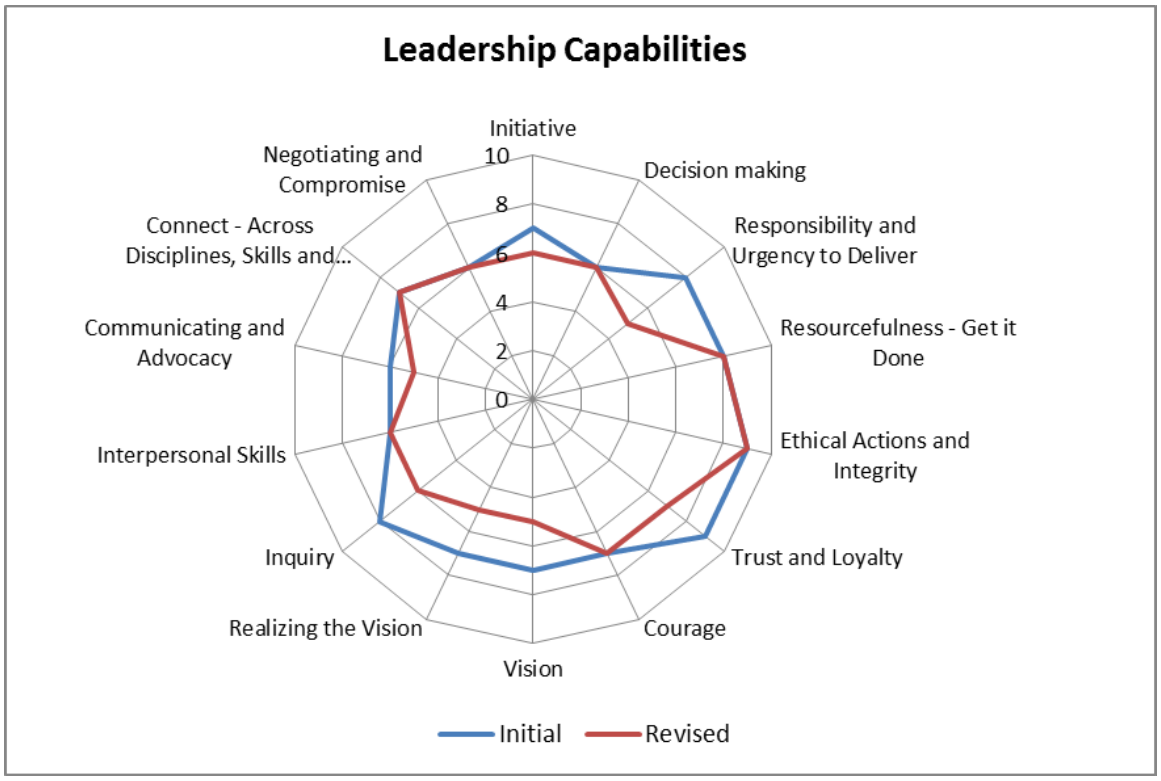
Realizing the Vision
In the past, I had been more comfortable with relying on others to provide the driving force behind a project. This was in part a reflection of weakness in translating abstract ideas into implementations. One major source of delay in completing my work assignments was in not knowing how or where to obtain the help I needed, whether technical or instructional. For team projects, I thus often depended on others to formulate solutions. Since I was the lead on this project, an excellent opportunity was presented to develop skills in this area. Furthermore, because substantial new technical knowledge is involved, it was critical that I took an active approach.

Decision Making
I was typically more comfortable with supplying facts and relying on others to make decisions. Thus, a leadership challenge was to improve decision-making capabilities. A major component to this project was gauging success or failure, making technical decisions, and decisions on when and whether alternative action is needed.

Interpersonal skills/Communicating and Advocacy
Effectively communicating opinions, confusion, and needs was often a weakness. I would occasionally express understanding of a topic when in actuality, additional clarification was needed. In this context, the leadership skill that needed improvement was in improving inquisitiveness and interpersonal skills so that others could understand when more information was needed.

Each member of the project team held expertise in different areas, so there was an excellent opportunity to practice effective communication. One point to mention was the theme of “influence without authority” - many of the individuals involved shared the same ‘rank’ within the organization, so an important skill was in being able to leverage members to take action.

My initial and revised assessments of leadership capabilities are shown in the spider chart below.



The lower revised scores should not be interpreted as regression, but as improved awareness of which skills needed attention, for example vision, advocacy, and courage.

For the majority of the project, I worked remotely which was not a preferred working style. E-mail was the primary mode of communication, which became challenging in terms of effectively communicating thoughts that were better expressed verbally, and from a logistics perspective, e.g., group members have to sacrifice time in order to read and respond to emails. This challenge became closely tied with advocacy – if I was not receiving the information I needed, it was my duty to take proactive steps to remedy the situation. For example, I adapted my communication style to be more assertive by following up regularly through email.

Courage was a characteristic that became a central area of improvement. More specifically, ‘try’ and ‘tell’ courage. ‘Try’ courage is the courage of initiative and action, making first attempts in spite of any doubts. For a large portion of the project, I was hesitant to implement ideas on my own because I assumed my lack of technical knowledge would be too major of a barrier. I preferred instead to defer to team members. This was highly ineffective due to competing priorities and time constraints. Moreover, as the lead on the project, it was ultimately my responsibility to own the task. Once this realization was made, I attempted more and more assignments on my own, for example, a) rewriting code to implement other algorithms, and b) writing code to analyze the feature selection process. This proved to be an invaluable leadership characteristic because it allowed me to drive the project at a faster rate. Nevertheless, this characteristic remains an active area of improvement.

‘Tell’ courage is the courage of voice, expressing doubts, differing opinions, and concerns. My personal vision, ideas, and priorities occasionally deviated from my team’s towards the end of the project. I learned to be more assertive in putting forth these ideas for discussion, and not always defaulting to the original plan. By having these discussions, I was able to better understand the overall vision and technical approach.

At one point in time, the project was in a critical state where corrective action was needed to avoid project failure. Thus, I strove to improve communication style and to effect ‘try’ and ‘tell’ courage. One way communication was improved was through writing up meeting minutes to ensure that I understood key takeaways after each meeting. I also implemented a time tracker to gauge which areas of the project I needed to spend additional time. From the standpoint of ‘tell’ courage, I became more assertive in expressing thoughts or needs. These actions enabled me to put the project on a better path to success.

To summarize, several key lessons were learned:

1) Influential leaders need to be able to make important decisions even without the necessary technical knowledge. In these situations, it is important that they leverage those with the knowledge to provide the answers they need
2) Proactivity vs reactivity (making things happen vs letting things happen) is a function of effective communication, assertiveness, and ‘try’ and ‘tell’ courage
3) Realizing the vision is not solely dependent on technical knowledge. Rather, it is equally important that the leader is capable of assessing outcomes and implications of tasks at a macroscale level. One mistake I made during this project was focusing too much on individual assignments and not stepping back to evaluate how smaller pieces fit together in the big picture. Much of the ability to do so requires strong inquiry – asking the right questions to stimulate forward thinking in terms of next steps, and to align goals and priorities.

11.2 Team staffing and organization

The involved Martinos Center members are Douglas Greve, Ender Konukoglu, and Satrajit Ghosh.

Douglas Greve was my organization sponser. He is a Martinos Center faculty member and an Assistant Professor in Radiology at Harvard Medical School. He was the primary point person for the project. The idea of developing a FreeSurfer QA tool had been on the organization’s “wish list” for a long time, but none had been able to fully commit to its development. Doug’s role was to provide expertise on FreeSurfer, machine learning, and overall project guidance.

Ender Konukoglu is an Instructor in Radiology at Harvard Medical School, and was regularly involved with the project. He lent his expertise on machine learning and was an incredible resource in driving the project’s vision.

Satrajit Ghosh is a Principal Research Scientist at MIT and Assistant Professor in Otology and Laryngology at Harvard Medical School. He supplied us with additional datasets.

The final members of the project team are Elise Sargent, Gordon Mentor, and Deniz Erdogmus, Faculty Advisor. Elise’s role was to provide leadership guidance and mentorship; she was incredible in both regards.

Deniz Erdogmus is an Assistant Professor in Electrical and Computer Engineering at Northeastern University. His added expertise on machine learning was invaluable to the project.

11.2.1 Project Team

Gordon Candidate: Matthew Goh
ISA: Doug Greve
Gordon Mentor: Elise Sargent
Faculty Advisor: Deniz Erdogmus
Others: Satrajit Gosh, Ender Konukoglu

12 SUMMARY

Completing this Challenge Project was an incredible experience. Though challenging from both technical and leadership standpoints, its value to my career cannot be overstated. I now have skills in machine learning and Python programming that I otherwise would not have developed. I also have a newfound interest in machine learning—prior to this project, it was not an area I had considered as a career path, but is now something I find extremely fascinating. Through this project, I also furthered my knowledge of bioimage signal processing, which is the very essence of my major and concentration.

This Challenge Project was also invaluable to becoming a better leader, worker, follower, and student. Working with differing work styles, personalities, and expectations pushed me to not only face my flaws, but also to be held accountable for them. Moving forward, I believe I am in a much stronger position to engage as an effective employee, leader, and follower.

14 REFERENCE

[1] Withy, D.J., Koles, Z.J. (2008). *A Review of Medical Image Segmentation: Methods and Available Software*. International Journal of Bioelectromagnetism.
[2] Vuoksimaa, E., Panizzon, M.S., et al. (2014). *The Genetic Association Between Neocortical Volume and General Cognitive Ability Is Driven by Global Surface Area Rather Than Thickness.* Cerebral Cortex.
[3] Desikan, R.S., Cabral, H.J., et al. (2008). *Temporoparietal MR Imaging Measures of Atrophy in Subjects with Mild Cognitive Impariment That Predict Subsequent Diagnosis of Alzheimer Disease.* American Journal of Neuroradiology. pp 1-7.
[4] Salat, D.H., Greve, D.N., et al. (2009). *Regional white matter volume differences in nondemented aging and Alzheimer's disease.* NeuroImage. pp 1247-1258.
[5] Van Horn, J.D. and Toga, A. (2014). *Human Neuroimaging as a "Big Data" Science*. Brain Imaging and Behavior. pp 323-331.
[6] https://www.glassdoor.com/Salaries/staff-research-associate-salary-SRCH_KO0,24.htm
[7] http://www.payscale.com/research/US/Job=Research_Associate_(Unspecified_Type)/Salary
[8] http://mrrc.yale.edu/users/charges.aspx
[9] https://www.framinghamheartstudy.org/participants/original.php
[10] C61. Abdi, H. & Williams, L.J. (2010). Jackknife. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): Encyclopedia of Research Design. Thousand Oaks (CA): Sage. pp. 655-660.
[11] Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*. pp. 861-875.