# OUTLIER DETECTION IN BRAIN MRI DIFFUSION DATASETS

# USE OF MACHINE LEARNING FOR OUTLIER DETECTION IN HEALTHY HUMAN BRAIN MAGNETIC RESONANCE IMAGING (MRI) DIFFUSION TENSOR (DT) DATASETS

BY

NEIL A. MACPHEE, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF SCHOOL OF BIOMEDICAL ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2022) McMaster University

(School of Biomedical Engineering) Hamilton, Ontario, Canada

| | |
|---|---|
| TITLE: | Use of Machine Learning for Outlier Detection in Healthy Human Brain Magnetic Resonance Imaging (MRI) Diffusion Tensor (DT) Datasets |
| AUTHOR: | Neil A. MacPhee<br>B.Eng. (Electrical & Biomedical Engineering),<br>McMaster University, Hamilton, Canada |
| SUPERVISOR: | Michael D. Noseworthy, Ph.D. |
| NUMBER OF PAGES: | xiv, 147 |

# Lay Abstract

Artificial intelligence (AI) refers to the ability of a computer or robot to mimic human traits such as problem solving or learning. Recently there has been an explosive interest in its uses for assisting in clinical analysis. However, successful use of these methods require a significantly large training set which can often contain outliers or incorrectly labeled data. Due to the sensitivity of these techniques to outliers, this often leads to poor classification rates as well as low specificity and sensitivity. The focus of this work was to evaluate different methods of outlier detection and investigate the presence of anomalies in large brain MRI datasets. The results of this study show that these large brain MRI datasets contain anomalies and provide a method best fit for identifying them.

# Abstract

Machine learning (ML) and deep learning (DL) are powerful techniques that allow for analysis and classification of large MRI datasets. With the growing accessibility of high-powered computing and large data storage, there has been an explosive interest in their uses for assisting clinical analysis and interpretation. Though these methods can provide insights into the data which are not possible through human analysis alone, they require significantly large datasets for training which can difficult for anyone (researcher and clinician) to obtain on their own. The growing use of publicly available, multi-site databases helps solve this problem. Inadvertently, however, these databases can sometimes contain outliers or incorrectly labeled data as the subjects may or may not have subclinical or underlying pathology unbeknownst to them or to those who did the data collection. Due to the outlier sensitivity of ML and DL techniques, inclusion of such data can lead to poor classification rates and subsequent low specificity and sensitivity. Thus, the focus of this work was to evaluate large brain MRI datasets, specifically diffusion tensor imaging (DTI), for the presence of anomalies and to validate and compare different methods of anomaly detection.

A total of 1029 male and female subjects ages 22 to 35 were downloaded from a global imaging repository and divided into 6 cohorts depending on their age and sex. Care was made to minimize variance due to hardware and hence only data from

a specific vendor (General Electric Healthcare) and MRI B0 field strength (i.e. 3 Tesla) were obtained. The raw DTI data (i.e. in this case DICOM images) was first preprocessed into scalar metrics (i.e. FA, RD, AD, MD) and warped to MNI152 T1 1mm standardized space using the FMRIB software library (FSL). Subsequently data was segmented into regions of interest (ROI) using the JHU DTI-based white-matter atlas and a mean was calculated for each ROI defined by that atlas. The ROI data was standardized and a Z-score, for each ROI over all subjects, was calculated. Four different algorithms were used for anomaly detection, including Z-score outlier detection, maximum likelihood estimator (MLE) and minimum covariance determinant (MCD) based Mahalanobis distance outlier detection, one-class support vector machine (OCSVM) outlier detection, and OCSVM novelty detection trained on MCD based Mahalanobis distance data.

The best outlier detector was found to be MCD based Mahalanobis distance, with the OCSVM novelty detector performing exceptionally well on the MCD based Mahalanobis distance data. From the results of this study, it is clear that these global databases contain outliers within their healthy control datasets, further reinforcing the need for the inclusion of outlier or novelty detection as part of the preprocessing pipeline for ML and DL related studies.

*In loving memory of my hairline*

*Gone but not forgotten.*

# Acknowledgements

I would first like to thank Dr. Noseworthy for all of his guidance and support over the years. It has been an amazing experience being a part of this lab and I'm very thankful for all of the insights and expertise I've gained during my time here.

I would also like to thank my colleagues for all of their assistance throughout this entire process, from helping me learn new scripting languages, to listening to me complain about my code, and even letting me sublet a room in their apartment (Thanks again, Cam).

In addition, I would also like to thank my parents, John and Marie MacPhee, for supporting me through out the entire process. I truly could not have done this without them.

# Contents

# List of Figures

# List of Tables

# Abbreviations

**AD** Axial Diffusivity

**ADC** Apparent diffusion coefficient

**ADNI** Alzheimer's Disease Neuroimaging Initiative

**BBB** Blood-brain barrier

**BOLD** Blood oxygen level dependent

**CNS** Central nervous system

**CSF** Cerebral Spinal Fluid

**CT** Computed tomography

**DL** Deep Learning

**DTI** Diffusion tensor imaging

**DWI** Diffusion-weighted image

**FA** Fractional Anisotropy

**fMRI** Functional magnetic resonance imaging

**FSL** FMRIB Software Library

**GMM** Gaussian mixture model

**IDE** Integrated Development Environment

**MAD** Median Absolute Deviation

**MCD** Minimum Covariance Determinant

**MD** Mean Diffusivity

**ML** Machine Learning

**MLE** Maximum Likelihood Estimator

**MRI** Magnetic resonance imaging

**mTBI** Mild traumatic brain injury

**PNS** Peripheral nervous system

**NMR** Nuclear magnetic resonance

**OCSVM** One-Class Support Vector Machine

**PCA** Principal Component Analysis

**PD** Proton density

**PSD** Phase sensitive detection

**RBF** Radial Base Function

**RD** Radial Diffusivity

**RF** Radio frequency

**ROI** Region Of Interest

**SVM** Support Vector Machine

**TE** Echo time

**TI** Inversion time

**TR** Repeat time

# Chapter 1

# Introduction

## 1.1 The Brain

### 1.1.1 Brain Anatomy

The human brain accounts for only approximately 2% of a person's body weight, yet it is the main consumer of glucose derived energy in the body (Mergenthaler *et al.*, 2013) and has been estimated to contain over 80 billion neurons (Azevedo *et al.*, 2009). In an adult, the average human brain weighs between 1300-1400 grams (Holloway, 1968). It is situated within the skull and is protected from physical and chemical injuries by cerebral spinal fluid (Tortora and Derrickson, 2014). At the ultrastructural level the brain is protected from microorganisms and toxins found in the blood by the blood-brain barrier (BBB), which is made up of tight junction sealed endothelial cells, basement membrane and surrounding astrocytic foot processes.

Macroscopically, there are 3 main sections of the adult brain: the cerebrum, the cerebellum and the brain stem. These sections are covered by protective layers,

called the cranial meninges, made up by the dura mater, the arachnoid mater, and the pia matter. These 3 sections, along with the spinal chord, make up the majority of the central nervous system (CNS). The cerebral cortex, or the outer layer of the cerebrum, contains grooves and ridges in order to increase the surface area of the brain. The ridges are referred to as gyri and the folds between gyri are referred to as sulci. Certain areas of the brain which have deep sulci are referred to as fissures.

Microscopically, the tissues of the brain can be divided up into two main tissue types: white matter and grey matter. Both the cerebral and cerebellar cortex are made up of grey matter, while the subcortical areas are mostly white matter. The majority of white matter is comprised of myelinated axons which gives it it's whiteish colour. Myelin allows for the faster transmission of nerve impulses through the axons. Grey matter is comprised of the unmyelinated axons, neuronal cell bodies, dendrites and more. The lack of myelination and the presence of Nissl bodies gives it it's greyish colour. Compared to grey matter, white matter has far fewer cell bodies and consumes only a fraction as much oxygen.

The brain receives it's blood supply from two sources, the internal carotid arteries and the vertebral arteries. The internal carotid arteries split to form the middle and anterior cerebral arteries. The right and left vertebral arteries combine to form the basilar artery, which connects with the blood supply from the internal carotid arteries via the posterior and anterior communicating arteries to form the circle of Willis. The circle of Willis is an arterial ring which improves the chances of blood flow to the regions of the brain in the case of an occlusion in the arterial vascular supply. Due to the physiological nature of neurons, they are more sensitive to oxygen deprivation than most cells in the body due to their high rate of metabolism requiring

a constant vascular supply.

### 1.1.2 Physiology

Each part of the brain specializes in its own specific roles which combine to allow for the overall function of the brain. The cerebrum controls the sensory and motor information of the body along with memory, emotions and behaviours. The cerebellum coordinates voluntary movement as well as fine tuning the precision of motor movements from the cerebrum, with the brain stem bridging the gap from the cerebrum and cerebellum to the spinal cord. The brain stem also performs autonomic processes such as breathing, heart rate and temperature regulation, as well as sleep cycles. Though the CNS is comprised of both white matter and grey matter, the majority of normal, everyday human function relies on the high number of neurons in the grey matter (Dolz *et al.*, 2020). These neurons allow for the processing of external information from the peripheral nervous system (PNS) as well as the transmission of new information through axons in the white matter (Chiao *et al.*, 2019). White matter also connects different regions of the brain to form functional circuits. The myelin acts as electrical insulator that allows high-speed transmission of impulses through a process call saltatory conduction. Damaged myelin can effect the conduction of these impulses resulting in cognitive, sensory and motor deficiencies.

### 1.1.3 Diseases

Though the brain has physical, chemical and biological mechanisms of defense, it can still become vulnerable to many diseases. These may arise from physical injury,

bacterial or viral infections, degenerative disorders, genetic disorders (e.g. inborn errors in metabolism and cancers) and through processes that have yet to be fully understood. The symptoms and their severity vary depending on the areas and brain matter that are effected.

One important degenerative disorder is Alzheimer's disease, being the cause of an estimated 60% to 80% of all dementia cases (alz, 2020). Alzheimer's disease has been shown to cause atrophy and inflammation in the brain, and though the entire progression is still unclear, accumulation of beta-amyloid and an abnormal version of the tau protein are associated with cognitive changes (Hanseeuw *et al.*, 2019). Alzheimer's disease is believed to begin decades before the onset of clinical symptoms (Jack *et al.*, 2009) with the progression of the disease being mostly unnoticeable until significant brain changes have occurred. The symptoms often experienced from the progression of Alzheimer's disease include language impairment and memory loss as the neurons responsible for these cognitive functions are injured or lost (alz, 2020). The progression of Alzheimer's disease from unnoticeable changes to the presence of symptoms causing disability is referred to as the Alzheimer's disease continuum and consists of three broad phases: preclinical Alzheimer's disease, mild cognitive impairment due to Alzheimer's disease, and dementia due to Alzheimer's disease. The preclinical Alzheimer's disease phase is defined as measurable brain changes without the presence of cognitive impairment, however, the detection of these changes becomes difficult in a clinical setting where the tools and expertise are not as widely available as in a research setting, leading to cases often going undiagnosed. In order to gauge the ability of neuroimaging and chemical biomarkers as assessment tools, the Alzheimer's Disease Neuroimaging Initiative (ADNI) has collected data on a large,

diverse cohort of subjects ranging from healthy controls to those on the Alzheimer's disease continuum (Petersen *et al.*, 2009).

Another more common disorder causing cognitive impairment is mild traumatic brain injury (mTBI), which encompasses the majority of traumatic brain injuries (Mcdonald *et al.*, 2012). It is estimated that at least 75% of all traumatic brain injuries are considered mild by the commonly used clinical criteria (Thornhill, 2000), with the majority of those traumatic brain injuries going unevaluated or unreported by a medical professional (Kay *et al.*, 1993). Mild TBIs are the result of physical impact or drastic changes in acceleration which is thought to cause the brain to collide with the inside of the skull and are characterized by their lack of a prolonged period of unconsciousness following the incident (Vos *et al.*, 2002). The majority of people who suffer an mTBI face minor symptoms and recover within a few months (Mccrea *et al.*, 2009). However, about 10% to 15% of people report persistent symptoms or dysfunctions long after the incident (Chapman *et al.*, 1999). Common symptoms include headaches, fatigue, anxiety, depression, increased irritability, as well as visual and vestibular disturbances (Comper *et al.*, 2005). Although there have been advances in the use of functional neuroimaging for the detection of physiological changes, structural magnetic resonance imaging (MRI) usually fails to show the presence of brain damage, making diagnosis of mTBI difficult.

### 1.1.4 Evaluation of White Matter Integrity

In order to ensure normal, everyday performance, any diseases or disorders which might affect the brain and its functions must be diagnosed and assessed as early as possible. While the majority of imaging studies surrounding the brain focuses on

the cortical gray matter, due to its importance to cognition, focus must also be put on the remaining 40% to 50% of the brain which lies underneath the cortex in order to ensure overall brain health. The majority of this region is comprised of white matter that facilitates the high-speed transmission of signals throughout the brain that are necessary for distributed neural systems such as memory and attention (Madden *et al.*, 2009). Reduced white matter integrity may result in disconnections between these distributed neural systems, as well as instability of signal transmission and neural noise.

The most common forms of brain imaging used are computed tomography (CT) and magnetic resonance imaging (MRI). For cognitive and behavioral studies functional MRI (fMRI; using blood oxygen level dependent (BOLD) effect) to assess cortical function, and structural MRI (assesses brain volumetric changes) are used. For examining white matter integrity neither approaches are used as they provide little information on the microscopic structures of the myelinated axons. However, another form of MRI called Diffusion Tensor Imaging (DTI) allows for the in vivo measurement of the microstructures of white matter by quantifying the direction and diffusion of water within a tissue. Since myelin restricts the diffusion of water perpendicular to the axon, leaving diffusion along the axon relatively free, these measurements can be used to gauge the integrity of white-matter structures. The diffusion is classified as either isotropic, having the same or similar value of diffusion in all directions, or anisotropic, having different rates of diffusion in different directions. Isotropic diffusion occurs in areas where water is unimpeded and allowed to diffuse in all directions, such as the ventricles, with the random movement of these molecules driven via Brownian motion (Kingsley, 2006). Anisotropic diffusion occurs in organized tissues, like

the axons of white matter, where the diffusion of water molecules is restricted in certain directions. The diffusion measurements in these structures can then be represented by different metrics to better describe the characteristics of the tissue. Using these metrics, and an atlas of brain white-matter structures, regional white-matter integrity can be compared to that of healthy subjects to assist in indicating relative health in that region. Research with healthy adults has expanded the size of these healthy brain data sets, allowing increased statistical power and estimations of the correlation between white-matter integrity and brain function (e.g. cognition).

## 1.2 Diffusion Tensor Imaging

Diffusion Tensor Imaging (DTI) is a MRI pulse sequence that has grown in popularity for its ability to provide information on the microscopic structures of white matter, not possible with any other imaging modality.

### 1.2.1 Magnetic Resonance Imaging

In order to fully understand DTI and its use, one must first understand the basics of MR imaging. The nuclear magnetic resonance (NMR) phenomenon was first observed by Bloch and Purcell in 1946, who went on to win the Nobel prize for physics in 1952. This was the beginning of the history which led to the development of MRI. The core components of an MR system are an radio frequency (RF) subsystem (RF amplifier, receiver and tuned RF coil), 3 orthogonal gradient coils (x,y, and z), a superconducting magnet, a controller/acquisition computer system, shielding against RF interference (i.e. Faraday cage) and active shielding (or bucking field)

to minimize the magnet fringe field. The magnets used in clinical systems usually range between 0.2 T and 3 T. Although recently there have been advances in head only and whole body 7T, 9T and 11T systems. There are considerable numbers of smaller bore animal research MRIs going up to 900MHz (i.e. 20T). The magnet is the main component of the MR system, employing superconductivity in order to produce the static magnetic field (B0). Superconductive magnets utilize the properties of certain materials that if cooled to almost absolute zero (-273.16 C, 0 K) have an electrical resistance of zero. This allows for a current to be run endlessly through a superconductive loop, maintaining the magnetic field indefinitely. These magnets, when compared to previous options such as iron-cored and permanent magnets, produce a much higher and more stable/homogenous magnetic field within the bore of the magnet. However, due to their need to be cooled to such low temperature, they require a helium cooling system in order to keep them functioning.

As mentioned, the RF subsystem is comprised of an RF amplifier, a tuned RF transmit/receive RF coil (or two separate coils, one for transmit and one for receive) and components for demodulation and signal storage. The transmitter generates RF pulses. If RF is transmitted at the Larmor frequency (Equation 1.2.1., proton spins with absorb energy leading to an excited state.

$$\omega_0 = \gamma B_0 \tag{1.2.1}$$

where $\omega_0$ is the frequency, $\gamma$ is the gyromagnetic ratio of the atom (a measured constant from nuclei having the quantum mechanical property of spin), and $B_0$ is the strength of the applied magnetic field.

After RF transmission and absorption of energy by the object, energy is released

and detected with a phase sensitive detection (PSD) RF receiver. After a number of steps including preamps, analog to digital conversion and demodulation of the carrier wave the data is stored in a 2D (or 3D) array of values called k-space. The k-space has real and imaginary channels and are used to reconstruct the final images. The primary transmitting coil used is the body coil, which is built in to the MRI and located inside the bore of the magnet. With this setup arrays of receive coils are then placed around the patient anatomy of interest to detec the signals. Ideal receiver coils maximize the MR signal while minimizing scanning noise. Since the presence of a patient or phantom inside the MRI will affect the field homogeneity, a "pre-scan" is used in order to optimize the transmitter and receiver and determine the exact Larmor frequency needed to properly achieve the correct excitation frequency.

MR systems employ the use of three orthogonal linear magnetic field gradients; Gx, Gy, and Gz. Scanners have these 3 components all collectively together in a gradient tube. These lie within the bore of the magnet and can be electromagnetically combined in order to produce a gradient in any direction and magnitude (within defined limits for safety). The gradients are used to spatially localize an MR signal to an area of interest within the body and are applied to spatially encode an image. There are a number of additional gradients that can also be employed, in addition to imaging gradients, that can be used for example in the encoding of diffusion. The axis along the bore is referred to as "z", the axis to the left and right is referred to as "x", and the axis going up and down is referred to as "y". The magnetic field gradients are applied through the use of three gradient amplifiers which apply voltage and current to each gradient coil in order to generate the linearly varying magnetic field. Linearity within this magnetic field is very important for gradients, as it is used

to spatialize the MR signal to different points within the scanner. Nonlinearity of the field can lead to errors such as geometric deformities as well as misplaced signal which result in outliers and artefacts in the data.

MR images are generated through a combination of all subsystems; the RF and gradient fields excite and give spatial information, respectively, and the measured signal of the spins returning to thermal equilibrium is recorded and transformed into an MR image. Based on the parameters set on the system, such as the echo time (TE), repeat time (TR), flip angle and, if used, the inversion time (TI) different types of image contrast may be selected. Contrast, or weighting, is governed by these settings working in conjunction with inherent tissue properties that drive the system back to thermal equilibrium. These tissue properties are the spin-spin relaxation (T2 and T2*), spin-lattice relaxation (T1) and proton density (PD). Subsequently images may be T1, T2, T2* or PD weighted.

### 1.2.2 Diffusion Weighted Imaging

Routine MR images will show the anatomy of a tissue of interest with different contrasts. However, MRI contrast may also be weighted towards diffusivity of water. The so called diffusion-weighted image (DWI) can be used to quantify the movement of water molecules within the tissue. The movement of water molecules follows Brownian motion. Classically diffusion weighting is done in 3 orthogonal planes producing diffusion weighted images along X, Y and Z. The degree of diffusion weighting is governed by the b-value:

$$b = \gamma^2 G^2 \delta^2 \cdot \left( \Delta - \frac{\delta}{3} \right) \tag{1.2.2}$$

where $\gamma$ is the gyromagnetic ratio, G is the amplitude of the diffusion encoding gradient (more details below), $\delta$ is the length of time the gradient is on and $\Delta$ is the diffusion encoding time, or time between diffusion encoding gradient lobes. The units of b-value are seconds per $mm^2$. Typical clinical scanning involves acquiring 3 DWI images (i.e. along X, Y and Z), each using the same b-value, and one image with the same timing but without diffusion encoding gradients (i.e. $b = 0s/mm^2$).

After diffusion weighted imaging is performed a parametric image, called an apparent diffusion weighted image is calculated according to Equation 1.2.3:

$$S(b) = S_{b=0} \cdot e^{-b \cdot ADC} \tag{1.2.3}$$

Which uses both the diffusion weighted image and the image with no diffusion weighting. It should be noted that the units of the parametric image, apparent diffusion coefficient (ADC), are $mm^2$ per second. The ADC is calculated from DWI data and measures the magnitude of diffusion within a voxel, shown in Equation 1.2.4.

$$ADC_k(\widehat{g_k}, b_k) = -\frac{1}{b_k} \log(\frac{S_k}{S_0}) \tag{1.2.4}$$

Where $S_0$ is the signal intensity with no diffusion gradients, $S_k$ is the signal intensity from the k-th signal from applying the b-value $b_k$. Note that the ADC does not contain any sort of measure of the direction, which is fine for use in areas such as grey matter where the orientation of the tissue does not affect the apparent diffusion. However, in more organized tissues with restricted diffusion, such as white matter, the apparent diffusion changes depending on the orientation of the tissue.

Such a DWI approach is not quantifiable. To quantify water diffusion a DWI

scheme is performed to acquire images encoding diffusion in multiple non-collinear directions (minimum number of 6). This approach, called diffusion tensor imaging (DTI) allows diffusivity within a voxel to be represented by a mathematical model called a "tensor". A tensor describes the shape, magnitude, and direction of water movement within every tissue voxel. Without restrictions or barriers, water is able to diffuse freely which is referred to as "isotropic" diffusion. Represented by a tensor, isotropic diffusion looks like a sphere as water molecules can thermally move in any direction. In the case of restrictions and barriers, such as in white matter or other directionally organized tissues, water is predominantly restricted to diffusion in a direction along the dominant axis of the fibre and is referred to as "anisotropic". Represented by a tensor, anisotropic diffusion can look like a cigar shape as the magnitude and direction of diffusion is restricted primarily to a certain axis. The information collected from DWI and the tensors calculated from them can be used to determine the underlying microscopic architecture of tissues.

DTI data is acquired by repeating DWI scans from a tissue of interest while varying the magnitude and orientation of the diffusion-sensitizing gradients. In order to calculate a tensor for a given voxel, at least 6 non-collinear directions must be acquired, in addition to a baseline image (no diffusion weighting). Routine clinical DTI acquisition is done typically acquiring only the minimum recommended number of 15-25 directions with one baseline image. But, for research-based acquisitions diffusion tensor data is often great than 60 directions with multiple baseline images. The model based diffusion tensor approach uses the Gaussian mixture model (GMM)

to describe the movement of water molecules, shown in Equation 1.2.5.

$$S_k = S_0 \sum_{i=1}^{N} f_n \exp(-b_k \widehat{g_k} D_n \widehat{g_k}) + \eta_k \tag{1.2.5}$$

The diffusion tensor, D, is a 3x3 symmetrical, positive-definite matrix, which means it has 3 positive eigenvalues representing the magnitude of diffusion, and 3 orthogonal eigenvectors representing each eigenvalue direction, as shown in Equation 1.2.6. The eigenvectors can be used as coordinates for calculating fibre tractography, with the major eigenvector defining the direction of greatest diffusion. The combination of these eigenvalues and eigenvectors gives the ellipsoid which represents the diffusion in that voxel.

$$D = \begin{bmatrix} D_x x & D_x y & D_x z \\ D_x y & D_y y & D_y z \\ D_x z & D_y z & D_z z \end{bmatrix} \tag{1.2.6}$$

Magnetic field gradients are notably used for spatial encoding, however they are crucial to diffusion weighted imaging for sensitizing MRI signals to the diffusion of water molecules within tissues. The gradient pulses used for diffusion weighted imaging most commonly have two gradient lobes of equal area which lie on either side of the 180° RF pulse. These gradients are usually referred to as Stejskal-Tanner gradients or bipolar gradients, even though they both go in the same direction for spin echo scans. As for the shape of the lobes, a variety of different pulses can be used, however, the trapezoidal shape is most popular due to their time efficiency, as seen in Figure 1.1. Since magnetic field gradients are still required for slice selection during the scan, the gradient pulses for spatial encoding usually occur right before diffusion

sensitizing gradient field pulses.

### 1.2.3 DTI Theory

The first person to incorporate diffusion terms into the signal equation that governs NMR, called the Bloch equation, was Torrey (Torrey, 1956). This helped describe the diffusion of free water molecules by incorporating the attenuation of the MR signal due to this physical property. Around a decade later, these Bloch-Torrey equations were solved by Stejskal and Tanner to describe the anisotropic diffusion of water molecules within a NMR volume (Stejskal and Tanner, 1965). It wasn't until much later that a model for estimating the diffusion tensor, D, was developed using multivariate linear regression (Basser *et al.*, 1994). Before this, diffusion related imaging could only be preformed in tissues where the direction of the fibres was known, such as in giant squid axons.

Diffusion tensor imaging has since been used for a wide variety of pathological research, such as in autism (Travers *et al.*, 2012), aging (Westlye *et al.*, 2009), Multiple Sclerosis (MS) (Filippi *et al.*, 2001), and schizophrenia (Kubicki *et al.*, 2007), to name but a few.

The diffusion tensor is measured by encoding water diffusivity along multiple non-collinear directions (at least 6 directions). The diffusion encoding direction is done through the application of combinations of amplitudes and timings of 3 orthogonal magnetic field gradients. Once water diffusion is measured, there are several ways one can quantify the shape of the tensors within each voxel. The simplest approach to DTI analysis is through a rank-2 tensor model leading to calculation of scalar metrics such as Fractional Anisotropy (FA), Mean Diffusivity (MD), Axial Diffusivity (AD), and

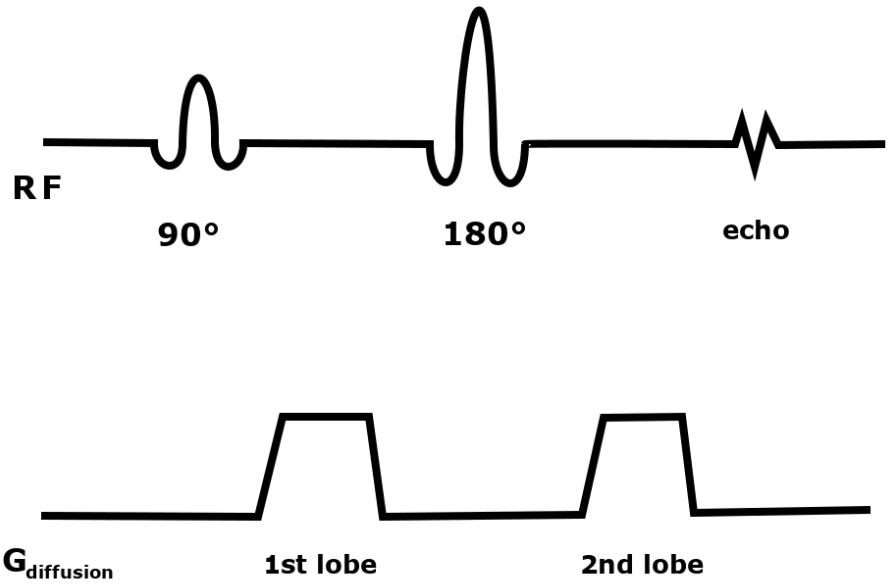



Figure 1.1: An example of a Stejskal-Tanner gradient with a trapezoidal shape.

Radial Diffusivity (RD). These measures directly relate to the three main Eigenvalues of the tensor, $\lambda_1+\lambda_2+\lambda_3$. Specifically, in AD only the value of $\lambda_1$ is quantified. While in RD the average of $\lambda_2$ and $\lambda_3$ is taken. MD provides an average of all three, $\lambda_1$, $\lambda_2$, and $\lambda_3$, which is also equal to the trace (TR) of the tensor. And finally, FA provides the relative difference between the largest eigenvalue compared to the others and represents the fraction of diffusion that is anisotropic, shown in Equation 1.2.7. The eigenvalue for each tensor is the value of the displacement/diffusion for each specific vector. In essence FA is a scalar metric providing the relative shape of the tensor. It scales between 0 and 1 where 0 is spherical (anisotropic) and 1 is a theoretically long and thin cylinder (anisotropic).

$$FA = \frac{1}{\sqrt{2}} \frac{\sqrt{(\lambda_1 - \widehat{\lambda})^2 + (\lambda_2 - \widehat{\lambda})^2 + (\lambda_3 - \widehat{\lambda})^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{1.2.7}$$

Any of these scalar diffusivity metrics can be shown as a parametric scalar images for quantification of the local tensors. MD is specifically sensitive to cerebral spinal fluid (CSF), which has high values of diffusivity. AD, however, is only sensitive to diffusion in the longest eigenvalue. For a parametric image of AD, highly organized structures like white matter pathways are bright. Large open cavities, like ventricles, have general high levels of diffusion which translates to high $\lambda_1$ values. RD represents the two shortest eigenvalues and shows dark values in highly organized and dense structures like white matter, intermediate values in grey matter, and high values in regions with CSF. FA plots the relative length of $\lambda_1$, compared to $\lambda_2$ and $\lambda_3$.

### 1.2.4 DTI Databases

With the growing use of machine learning for classification and exploratory data analysis, there is a need for large, publicly accessible databases containing a diverse population of medical imaging data. Currently, these databases are usually focused on a specific disorder (e.g. austism, Alzheimer's disease) and contain multi-site data acquired from numerous imaging centres and MRI systems across the world. These databases allow researchers to access large datasets containing healthy and diseased subjects which is crucial for medical imaging based machine learning and deep learning.

In this study, the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu) was used, which is comprised of healthy controls as well as subjects within the three phases of Alzheimer's disease. ADNI is a mutlisite study containing the data from 63 facilities across Canada and the US with the goal of tracking the progression of Alzheimer's disease and aging through imaging, clinical assessment, genetic and biospecimen biomarkers. Care has been taken in the compiling of these databases by researchers to ensure all scanning is done on a similar scanner (3T), and scan quality of each image was visually graded by an analyst. The scan quality is graded from 1 to 4, with grades 1, 2, and 3 being accepted and a grade of 4 being unusable. For this study, only the healthy controls were used from the database.

## 1.3 Machine Learning

### 1.3.1 What is Machine Learning?

There has been an immense rise in the interest of using machine learning within the field of MRI. From decoding brain states to finding early biomarkers for disease, machine learning and pattern recognition have shown themselves to be important tools in analyzing large, high-dimensional data. The goal of machine learning is to be able to make predictions based on prior observations and provide insights into the underlying structure of the data.

Machine learning can be broken down into two main categories, supervised learning and unsupervised learning. Supervised machine learning requires prior knowledge on the expected outcome of the data, such as a class label or a continuous output value. The goal of a supervised machine learning model is to learn the relationships between input variables, or features, and the output value based on a sample of the data. Supervised machine learning requires the partitioning of the sample data in to a "training set" and a "test set". The machine learning model is fed the training set in order to learn the patterns associated with the input and output variables to make predictions based on the data. Once trained, the model is tested on the input variables of the test set with the outputs removed and predictions based on the relationships it learned. The predictions are then compared with the outputs of the test set in order to gauge the accuracy and precision of the model. Supervised machine learning can be further broken down into classification or regression based on whether the interest is in predicting a discrete value, like a class label, or a continuous value.

Unsupervised machine learning does not require explicitly labeled output data

and is used in analyzing the underlying structure of the data. It is often used for exploratory analysis of data in order to provide insights on the features and their relationships with one another, as well as dimensionality reduction. Since there are no labels provided with the data, the models classify or group the data based solely on the characteristics and relationships of the input features. Unsupervised machine learning can be further broken down into clustering and association based problems. Clustering is where the model tries to group similar data points together to discover the possible presence of classes/groupings within the data, where as association tries to discover trends in the data, for example, if someone buys product X from a store they are likely to also buy product Y. A common use of unsupervised machine learning for high-dimensional data is dimensionality reduction, which is a method used to represent the input data with less features by transforming it into a lower dimensional space while maintaining the important information. This often leads to the removal of redundant features, usually containing noise, while preserving the majority of the variance.

### 1.3.2 Outlier Detection

Both supervised and unsupervised machine learning methods can be used to detect erroneous data within a dataset, depending on the availability of labeled data for training. The unsupervised approach is referred to as "outlier detection" and often utilizes clustering, one-class support vector machines (OCSVM), and/or a variety of distance metrics to identify anomalies. Outlier detectors are best fit for datasets where it is suspected that there may be incorrectly included data within the dataset. Since there are no class labels, the outlier detectors require manual tuning of parameters

in order to achieve maximum performance. In this study, OCSVM and Mahalanobis distance were chosen as outlier detector. Support vector machines are the most widely used machine learning model in health care (Jiang *et al.*, 2017) and minimum covariance determinant (MCD) based Mahalanobis distance has been shown to be an efficient tool for outlier detection (Hubert *et al.*, 2017).

The supervised approach to anomaly detection is referred to as "novelty detection", and requires a clean dataset on which the model can be trained before it can be used to detect anomalies. Novelty detectors are a form of "one-class classification" where the model learns the characteristics of labeled normal data. This form of anomaly detection is best suited for situations where there is an abundance of labeled "normal" data, but there lacks enough anomaly data to construct non-normal classes. In this study, a "clean" subset of data derived from the MCD-based Mahalanobis distance was used to train a OCSVM novelty detector to investigate the possibility of training a novelty detector from the output of a outlier detector. One of the main reasons to investigate this is to allow for automated hyperparameter tuning, which is not possible with outlier detectors.

### 1.3.3 Principal Component Analysis

Principal component analysis (PCA) is an unsupervised machine learning algorithm for reducing the dimensionality of a dataset first presented in 1901 by Karl Pearson (Pearson, 1901) but was later developed independently in the 1930s by Harold Hotelling (Hotelling, 1933) where it received it's current name. Most dimensionality reduction models fall into two classes: feature elimination, where features are removed

to reduce the number of variables considered, and feature extraction, where new, independent variables based on the prior ones are created and ordered on how well they explain the variance of the data. PCA is a form of feature extraction where features are combined and projected onto a new, lower dimensional feature space that best explains the variance of the data. Once the data is in the new feature space, feature elimination can be used to remove principal components which do not explain a significant portion of the variance, reducing the number of features while maintaining the majority of the original information.

Before feature extraction can be applied, PCA requires either normalization or standardization of the independent variables $X$. In this study, standardization was chosen as the z-scores of the data were of interest. Once the data has been standardized, the standardized matrix $Z$ was then transposed and multiplied by $Z$ to compute the covariance matrix, $Z^T Z$. From the covariance matrix the eigenvectors and eigenvalues are calculated. The covariance matrix $Z^T Z$ is decomposed into $PDP^{-1}$ through eigendecomposition, where $P$ is a matrix containing the eigenvectors and $D$ is a diagonal matrix with the eigenvalues located on the diagonal. The eigenvalues of $D$ are associated to the eigenvectors $P$ in the corresponding columns. The eigenvalues are then sorted from largest to smallest as well as their corresponding eigenvectors. This new matrix of sorted eigenvectors is denoted as $P^*$ and is ordered by the importance of the eigenvectors, as determined by their eigenvalue. The matrix $Z^*$ is then calculated using Equation 1.3.1.

$$Z^* = ZP^* \tag{1.3.1}$$

Where $Z^*$ is a standardized version of $X$ with the new features being a combination of the original features, in order of their relative importance.

At this point, feature elimination is usually employed to remove principal components which are redundant or only explain a negligible proportion of the explained variance. This can be done by constructing a scree plot to observe the proportion of the variability explained by each principal component and choosing a cut off point based on the "elbow" of the plot or a pre-determined percentage of the total variance.

### 1.3.4 Support Vector Machines

SVMs are a machine learning algorithm used for both classification and regression which work by finding a line or hyperplane that best separates classes within the data. Though most SVMs are used in supervised learning, OCSVMs can be used in unsupervised learning for the detection of anomalies within the data (Schölkopf *et al.*, 1999). One of the most useful properties of SVMs is their ability to form non-linear decision boundaries by projecting the data onto a higher dimensional space using a non-linear function. Non-linear decision boundaries are often useful for OCSVMs in determining whether a data point is within the normal distribution as anomalies can be located anywhere outside the one class, often making linear separation impossible.

For non-linear decision boundaries, data points are separated by projecting data points from the original space, $X$, to a higher dimensional space, $F$, where a hyperplane can be used to separate the data. When projected back down onto the original space, a non-linear decision boundary is formed. The hyperplane determines the margin between the classes, and with the closest points from both classes being equidistant

from the decision boundary, the hyperplane searches for the "maximal margin separating them. For OCSVMs according to Schölkopf (Schölkopf *et al.*, 1999), the data is separated from the origin in the feature space $F$ and attempts to maximize the distance from the hyperplane to the origin. Since the data has been standardized, the origin lies at the center of the data distribution. This captures where the probability density of the data is in the input space, $X$. The hyperplane used to form the decision boundary for a dataset $X = x_1, x_2, ..., x_i$, where $i \in N$ and $N$ is the number of observations, is shown in Equation 1.3.2.

$$w^T x + b = 0 \tag{1.3.2}$$

Where $w$ is a vector normal to the hyperplane and $b$ is a variable representing the offset. This results in a small region of inliers given the class label +1, with everything outside the decision function given a class label of -1. The aim of the OCSVM is to solve the minimization function, shown in Equation 1.3.3 (Schölkopf *et al.*, 1999).

$$\min_{w,\xi,\rho} \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \tag{1.3.3}$$

subject to:

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \qquad \text{for all i = 1,...,n}$$

$$\xi_i \geq 0 \qquad \text{for all i = 1,...,n}$$

Where w is a vector normal to the hyperplane, $\phi$ is a feature map that maps the input data into the feature space, $\xi$ is a slack variable that allows some points to lie within the margins, and $\nu$ is a unique variable used to tune the OCSVM. The $\nu$

variable acts as both the upper bound of the fraction of data points excluded by the decision function as outliers, as well as the lower bound of the fraction of data points used as support vectors. The range of values that can be used for $\nu$ must be between 0 and 1.

The "kernel trick" can be used to reduce the complexity of the calculations necessary for non-linear decision boundaries. The decision function only relies on the dot product of vectors in the feature space, $F$, so it is not necessary to project the data onto the higher dimensional space. Using the kernel function, $K$, shown in Equation 1.3.4, the same results can be achieved without the need for explicit projection.

$$K(x, x_i) = \rho(x)^T \rho(x) \tag{1.3.4}$$

Since the function $K$ returns the same results as projecting the data to $F$, it can be used as a replacement. This gives SVMs a lot of power with non-linearly separable points as the feature space can theoretically be an unlimited number of dimensions, allowing the hyperplane to form highly complex decision functions. There are several kernel functions that can be used with SVMs, including linear, sigmoidal, and polynomial kernels. However, the kernel used in this study is the Gaussian Radial Base Function (RBF), shown in Equation 1.3.5

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \tag{1.3.5}$$

Where $\sigma$ is the variance and hyperparameter of the kernel and $||x - x'||$ is the Euclidean distance used as a dissimilarity measure. When the minimization function is solved using Lagrange multipliers and the "kernel trick", the decision function

becomes Equation 1.3.6.

$$f(x) = sgn((w \cdot \phi(x_i)) - \rho) = sgn(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho) \tag{1.3.6}$$

Where $\alpha_i$ are the Lagrange multipliers. Only $\alpha_i > 0$ are weighted when forming the decision function, meaning only the support vectors, or the points closest to the margin, are used when creating the decision boundary.

### 1.3.5 Mahalanobis Distance

Mahalanobis distance is a distance metric that measures the distance from a point to the center of a distribution. It is very useful when working with multivariate data as well as for one-class classification, multivariate anomaly detection, and semi-supervised problems with heavily unbalanced datasets. First introduced by P.C. Mahalanobis in 1939, the Mahalanobis distance is effectively a multivariate form of Euclidean distance. The formula for calculating the Mahalanobis distance can be seen in Equation 1.3.7.

$$d_{(\mu,\Sigma)}(x_i)^2 = (x - \mu)^T \Sigma^{-1} (x - \mu) \tag{1.3.7}$$

Where $d_{(\mu,\Sigma)}(x_i)^2$ is the squared Mahalanobis distance, $x$ is a vector of the observation, $\mu$ is a vector of the means of the features, and $\Sigma^{-1}$ is the inverse covariance matrix. With (x-$\mu$) being the distance from a point to the distribution divided by the covariance matrix, it is essentially the multivariate equivalent to standardization. If the covariance is high due to the features being correlated, the distance will be

reduced. The Mahalanobis distance works by first transforming the features into uncorrelated variables, scaling the variance of those features to 1, and then calculating the Euclidean distance. This addresses the problem of correlation as well as scaling.

In a practical setting, $\mu$ and $\Sigma$ are replaced by estimations of the mean and covariance. Maximum likelihood estimator (MLE) and minimum covariance determinant (MCD) estimator are often used in the estimation of these parameters. MLE is more sensitive to outliers within the data which can lead to issues with the calculated Mahalanobis distance whereas MCD is a far more robust estimator which forms a "clean" subset of the data from which it estimates the parameters. MCD was first introduced by P.J. Rousseuw in 1984 (Rousseeuw, 1984), and works by finding a subset of $\frac{n_{samples}+n_{features}+1}{2}$ samples that have the smallest determinant for their empirical covariance.

### 1.3.6 Outlier Detection in the Brain

Several groups, using a variety of measures, have investigated the problem of outliers within quantitative brain MR images. Mejia et al. (Mejia *et al.*, 2017) investigated a form of outlier detection for use in high-dimensional functional magnetic resonance (fMRI) data that provided a single measure of "outlyingness". After preprocessing and scaling each volume to its median and median absolute deviation (MAD), the PCA leverage was calculated and used as a measure of outlyingness due to its relation to the Mahalanobis distance. Using the ABIDE database containing multiple subjects from 16 different sites ( Di Martino et al, 2014), Mejia et al. were able to improve the reliability of resting state networks and functional connectivity. The rs-fMRI results also concluded that while a threshold of 4 to 5 times the median leverage worked best

for detecting artifacts, different MRI approaches and differing imaging modalities, and types of outliers, may require different thresholds.

Looking at structural brain MR images, Moldovanu et al. (Moldovanu *et al.*, 2015) proposed the inclusion of a "detection of outliers" stage between acquisition and pre-processing using elimination criteria and distance functions. The elimination criteria included Chauvenet's, Pierce's and Grubb's criteria and were applied to the histogram image distributions to remove outlier values. Once removed, the similarity between images with and without outliers were measured using three distance functions: Euclidean, Minkowsky, and cosine distance. The results indicated that differences in contrast can affect the detection of outliers, and the inefficiency of cosine distance as a method of outlier detection.

Mourão-Miranda et al. (Mourão-Miranda *et al.*, 2011) investigated the use of a one-class SVM (OC-SVM) outlier detector in order to differentiate depressed patients from healthy controls based on their fMRI data. Rather than approaching the problem as a differentiation between two or more classes, they utilized the concept of a range of "normality" in which a subject has a measurable distance from, and utilized the discriminatory nature of OC-SVMs to detect signs of depression in response to sad facial expressions during functional brain imaging. Having formed a distribution of "normal" patterns by healthy controls in response to specific stimuli, they were able to form an objective measure of the degree in which a psychological disorder caused deviations from the normal. They found a significant correlation between patients' Hamilton Rating Scale for depression scores and their distance from the normal decision boundary, and they were able to identify two subgroups within the cohort which differed in their response to treatment for depression.

Looking at diffusion magnetic resonance imaging (dMRI), Andersson et al. (Andersson *et al.*, 2016) developed a method to detect and replace outlier data caused by patient movement. Due to motion during the diffusion encoding part of the imaging, signal loss can occur causing parts of the scan to become unusable and can biased estimations due to the presence of outlier values. Where as in the past one might exclude voxels, slices, or volumes which contain outlier values, Andersson et al. proposed a method using a Gaussian process in order to predict what a certain region should look like based on an assumption about neighbouring points in Q-space, an advanced DTI technique using multiple b-values. They were able to replace outliers using non-outlier data from surrounding image areas, therefore minimizing the impact of the outlier on any subsequent diffusion analysis.

Also looking at single patient outlier detection and replacement in dMRI, Niethammer et al. (Niethammer *et al.*, 2007) proposed a detection method based on thresholding the Laplacian of the apparent diffusion coefficient (ADC) and replacing the outlier values with sphere-neighborhood-interpolated values. To find severe outlier values, the Laplacian of the ADC values over the sphere were calculated and a threshold was established from reference DWI images. Maximum and minimum "normal" Laplacian values for an ADC value were determined and used as a very simple outlier rejection method. Outlier voxels were then replaced with smoothly interpolated values based on the neighbouring voxels. Niethammer et al. were able to improve the directional information from linear least squares tensor estimation using their outlier detection and replacement method, reducing the effects of outlier values.

Most recently, Sairanen et al. (Sairanen *et al.*, 2018) developed a slice-wise outlier detection (SOLID) method for diffusion MRI data that uses a degree of "uncertainty"

to adjust outlier values rather removing them. In order to avoid distributing slice wise outliers over multiple volumes, SOLID was applied before image registration and transformation. The Z-score was calculated for each slice and outlier detection was done by comparing each slice observation to the mean for the whole brain. The modified Z-score was also used to detect suspicious slices, allowing for their identification and visual inspection. Uncertainty weightings, called voxel-wise SOLID weights, were derived from the modified Z-score and scaled linearly from 1 (normal) to 0 (outlier) and used to inform a model estimator, based on which model the user chooses. SOLID was able to identify all outliers when tested against other slice-wise outlier detection methods and produced fewer false positives than the others.

# Chapter 2

# Problem Definition and Hypothesis

## 2.1 Problem Definition

With the growing use of machine learning and deep learning as complementary tools in medical diagnostics, medical datasets need to be preprocessed and analyzed prior to the training process in order to catch outliers so as to ensure the best possible learning. Depending on the occurrence rate, outliers can drastically effect the ability of a model to correctly classify or predict data. DTI data has a high dimensional data format which makes it impossible to analyze with the naked eye. Methods, such as z-score, Mahalanobis distance, isolation forests, and density based spatial clustering of applications with noise (DBSCAN) are currently used to identify outliers. Although these are reliable methods for detecting outliers, there remains room for improvement and better specificity and sensitivity.

## 2.2 Hypothesis Statement and Proposed Solution

There are increasing numbers of large repositories of imaging data. This includes not only MRI, but also PET/CT, ultrasound, mammograms, etc. Although there are supposedly strict measures for inclusion of normal data to these repositories, it is not only possible, but highly likely, that non-normal data is inadvertently incorporated into the data. Non-normal data, for this case, means data (images) that have some region/voxel/pixel that is not within what would be considered a normal range, depending on the metric being assessed. The image type can be highly varied. In this work, diffusion tensor imaging (DTI) data was chosen, which, in its raw form, constitutes thousands of images. These images are then 'distilled' through some form of tensor decomposition into DTI scalar metrics (described previously). DTI metrics are used to describe the health of white-matter myelin and as such are affected by many brain diseases. For example, multiple sclerosis (MS) causes inflammation, edema and breakdown of myelin while concussion causes transient (hopefully) loss in myelin integrity, both of which can be assessed with DTI. Numerous groups have shown non-normative values of DTI metrics for most psychological disorders (schizophrenia, depression, ADHD, bipolar, etc.). Non-normative data could enter into these normative datasets through anything from sub-clinical or undiagnosed disease to previous head injury that was not recalled or reported. Unfortunately, these inclusions in the normative dataset repositories could lead to problems in training for machine learning algorithms leading to elevated risk of higher false positive and false negative rates. Thus, the objective of this project was to test and compare a combination of machine learning methods to identify outliers in normative brain image datasets. Specifically, this work focused on DTI due to it's widespread application in

understanding brain white-matter damage.

# Chapter 3

# Materials and Methods

## 3.1 Data

The data used in this study was obtained through the public access database ADNI (Petersen *et al.*, 2009). Care was made to minimize variance due to hardware and hence only data from a specific vendor (General Electric Healthcare) and MRI $B_0$ field strength (i.e. 3 Tesla) were obtained. The raw DTI data (i.e. in this case DICOM images) chosen from this database were initially 1036 healthy controls, 473 male and 563 female, between the ages of 22 and 40. Due to the much smaller dataset size of the male and female subjects age 36 to 40 (5 subjects or less per cohort) they were excluded from the present study as they did not contain enough data to represent their cohort. The remaining 1029 subjects were grouped by age and sex into 6 different datasets shown below in Table 3.1.

| Datasets | Number of Subjects |
|---|---|
| Male 22 to 25 | 140 |
| Male 26 to 30 | 206 |
| Male 31 to 35 | 125 |
| Female 22 to 25 | 80 |
| Female 26 to 30 | 246 |
| Female 31 to 35 | 232 |

Table 3.1: Age and sex grouped datasets.

## 3.2 Hardware and Software

This study was done using a server located at the Imaging Research Center at St. Joseph's Healthcare in Hamilton, ON. The server had an AMD Ryzen Threadripper 3960X 24-core CPU, an nVidia RTX 2080 GPU and 64GB of RAM and was running Ubuntu 20.04.4 LTS as its operating system.

The preprocessing was done using FMRIB Software Library v5.0 (FSL) (Jenkinson *et al.*, 2012), a free library of analysis tools designed for brain imaging data. Several of the toolboxes from the library were used in the preprocessing phase and are mentioned in section 3.3.1. The programming language used for modeling was Python version 3.8 (Van Rossum and Drake, 2009), using the open-source integrated development environment (IDE) Spyder (Raybaut, 2009). The Python libraries used during research are listed below in Table 3.2.

| Python Library | Description |
|---|---|
| matplotlib | A library used for plotting and visualizing data (Hunter, 2007) |
| numpy | A library adding multiple mathematical functions useful for arrays and matrices (Harris *et al.*, 2020) |
| pandas | A library used for data manipulation and analysis, also adding useful data structures for multivariate data (McKinney *et al.*, 2010) |
| os | A built in module for directory fetching, creation, and navigation as well as other operating system dependent functionality (Van Rossum and Drake, 2009) |
| sklearn | A machine learning library designed for Python allowing for predictive and exploratory data analysis (Pedregosa *et al.*, 2011) |
| scipy | A library for scientific computing offering functions for a wide variety of scientific and engineering tasks (Virtanen *et al.*, 2020) |

Table 3.2: List of Python libraries used during research

## 3.3 Data Processing

### 3.3.1 Preprocessing

The raw DTI data was converted into NIfTI (Neuroimaging Informatics Technology Initiative) files using dcm2niix (Li *et al.*, 2016) for future processing. FMRIB Software Library v5.0 (FSL) (Jenkinson *et al.*, 2012) and it's toolboxes were used to prepare the NIfTI files for analysis. Brain extraction was performed using FSL's Brain Extraction Tool (Smith, 2002) to isolate the brains from the surrounding anatomy.

With the brains extracted, eddy current and motion correction was performed on the data using FSL's FDT Diffusion tool (Behrens *et al.*, 2003, 2007). The diffusion tensor was then reconstructed from the processed data using FSL's dtifit tool (Behrens *et al.*, 2003, 2007) and output as diffusion scalar metrics (i.e FA, RD, AD, MD).

The reconstructed data was then registered to the MNI152 T1 1mm standardized space (Mazziotta *et al.*, 1995) in order to allow for comparison between brains. The brains were then segmented into 18 regions of interest (ROI) using binary masks according to the JHU DTI-based white-matter tractography atlas (Mori *et al.*, 2005) and a mean was calculated for each ROI defined by that atlas. The JHU white-matter tractography atlas was chosen as it identifies the probability of certain voxels pertaining to specific white matter structures of interest within the brain. In order to ensure each subject had the same number of voxels per ROI, the atlas was applied to one of the brains from the dataset at random and the probabilistic ROIs were saved as binary masks. These binary masks were used to extract identically shaped ROIs from each brain. The ROIs from the atlas used are listed below in Table 3.3.

The binary masks were applied to each subject using fslmaths, part of FSL's 'FSLUTILS' toolbox which allows for mathematical manipulation of images through the command-line, in order to create nifty files of the isolated ROIs. The pipeline was automated using Bash scripts which were executed through a command-line, and a flow chart showing the entire process is shown in Figure 3.1. A Python script was used to iterate through the segmented brains of a cohort to ensure each ROI had the same number of non-zero voxels. The means of the 18 ROIs for each subject were then added to a CSV file corresponding to each of the 4 diffusion metrics for their sex and age grouped datasets. This was performed in Python version 3.8 (Van Rossum

| **JHU White-matter Tractography Atlas ROIs** |
|---|
| 'Acoustic radiation Left' |
| 'Acoustic radiation Right' |
| 'Callosal body' |
| 'Cingulum Left' |
| 'Cingulum Right' |
| 'Corticospinal tract Left' |
| 'Corticospinal tract Right' |
| 'Fornix' |
| 'Inferior occipito-frontal fascicle Left' |
| 'Inferior occipito-frontal fascicle Right' |
| 'Optic radiation Left' |
| 'Optic radiation Right' |
| 'Superior longitudinal fascicle Left' |
| 'Superior longitudinal fascicle Right' |
| 'Superior occipito-frontal fascicle Left' |
| 'Superior occipito-frontal fascicle Right' |
| 'Uncinate fascicle Left' |
| 'Uncinate fascicle Right' |

Table 3.3: Names of white-matter structures chosen as ROIs.

and Drake, 2009) using the function "pandas.DataFrame.to_csv" from the library "Pandas".

### 3.3.2 Outlier Creation

In order to assure that the outlier detection techniques being investigated were able to identify outliers within the datasets, abnormal data was created from existing data with a range of "outlyingness" to test each method. The synthesized outliers were created from 5 subjects in the excluded cohorts so as to not reuse subjects from the dataset being tested. The data was loaded into Python and the outliers were

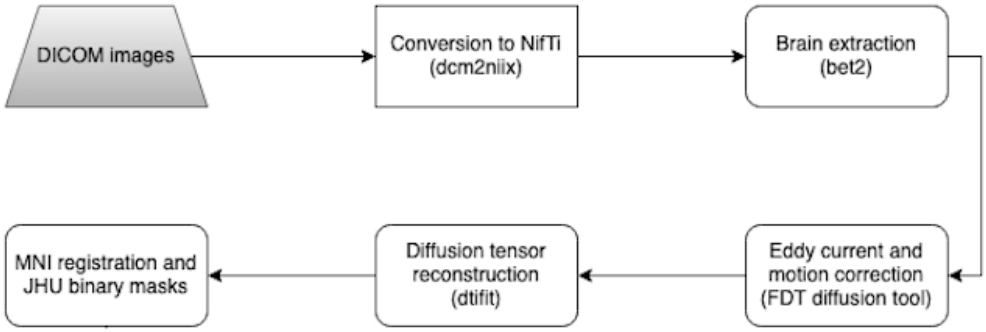



Figure 3.1: Flow chart displaying the preprocessing pipeline.

given a degree of "outlyingness" which was determined by multiplying their subject number (a range from 1 to 5) by 0.35, adding 1, and then multiplying the means of their ROIs by that degree. The synthesized outliers had their ROI means increased by %135, %170, %205, %240, and %275 respectfully. Using a range of different degrees of "outlyingness" was chosen to observe how the different outlier detection techniques interact with mild to extreme outliers. The outliers were then added to the Female 26 to 30 cohort for testing. An example showing the Z-scores of the effected dataset can be seen in Figure 4.3.

Initially, the outliers were created by adding random standard Gaussian noise or uniform noise to the raw nifti file after being converted from DICOM, prior to any preprocessing. This was done by utilizing the "randn" function, which adds Gaussian noise, as well as the "rand" function, which adds uniform noise, from the "fslmaths" toolbox. For the uniform noise, 0.1 unit increments were used to adjust the amount of noise added to the image file as the "rand" function takes a range from 0 to 1. However, it became apparent that the amount of noise added to the image did not clearly dictate the degree of "outlyingness" after it had been preprocessed.

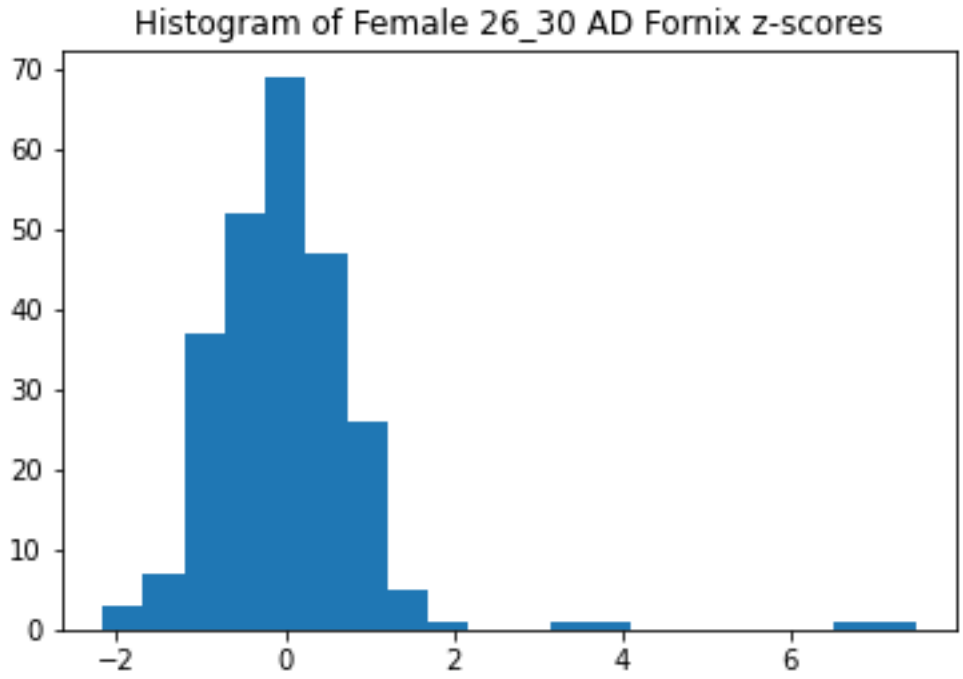



Figure 3.2: Histogram plot showing the Z-scores of the Fornix from the AD Female 26 to 30 cohort with synthesized outliers

A more quantitative approach post-processing was chosen to create the outliers, as described above.

### 3.3.3 Feature Extraction

Feature extraction is a fundamental step in machine learning in order to ensure your model is performing at it's best. First, the diffusion metric data containing the means of the ROIs was loading into Python using the function 'pandas.read_csv', each sex and age cohort being analyzed separately. The means were then standardized using 'sklearn.StandardScaler" which sets the mean of the feature to zero and the standard deviation to one. The equation used to standardize the data is shown in

Equation 3.3.1.

$$z = \frac{(x - \mu)}{\sigma} \tag{3.3.1}$$

Where $x$ is a sample from the dataset, $\mu$ is the mean of the dataset, $\sigma$ is the standard deviation the sample and $z$ is the standard score, or Z-score. Once the data had been standardized, histograms were produced for each ROI to ensure that the distributions were normal. An example of one of these histograms is shown in Figure 3.3.

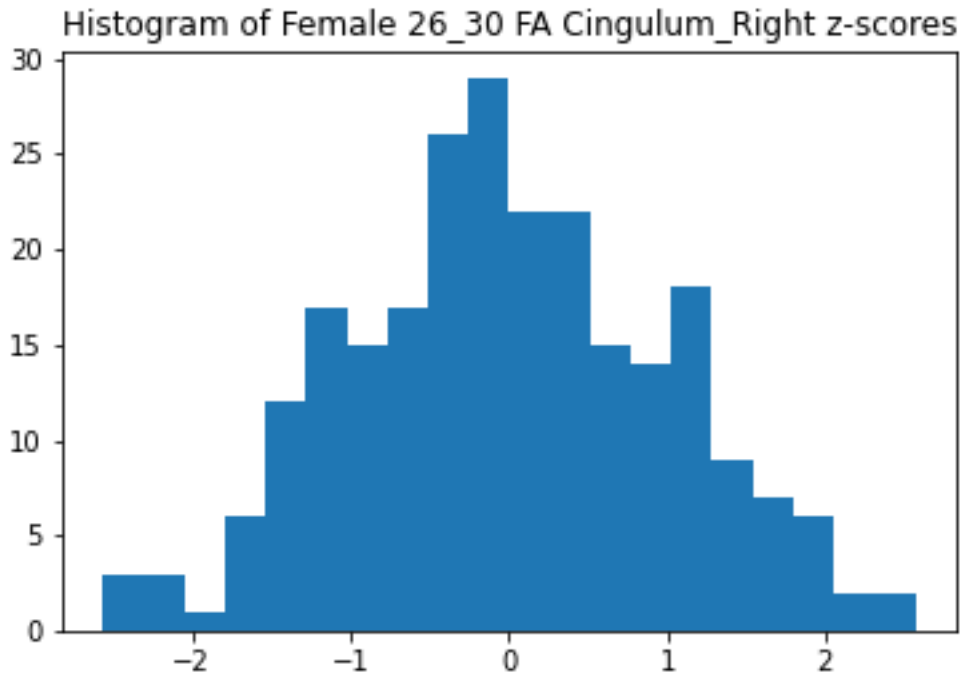



Figure 3.3: Histogram plot showing the Z-scores of the right Cingulum from the FA Female 26 to 30 cohort

In order to observe how many ROIs from each subject had Z-scores which

may be classified as outliers, a cut off of 99.5% (or a Z-score of >2.807/<-2.807) was chosen. Each subject with an ROI Z-score beyond the cutoff was recorded along with which ROI and diffusion metric it was from. A plot was produced showing the number of Z-scores beyond the cutoff for each subject, with each diffusion metric shown in a different colour.

The diffusion metric data was combined for each ROI to create CSV files containing the standardized means FA, MD, RD, and AD of each subject. A CSV file was created for each ROI so that they could be analyzed separately from one another. Principal component analysis (PCA) was performed on the ROI data using the function "PCA" from the library "sklearn.decomposition" to reduce the dimensionality of the datasets. A scree plot was made to determine the number of principal components needed to encapsulate at least 95% of the explained variance, and can be seen in Figure 3.4 and a table containing the explained variance values for each principal component found in Table 3.4. A cut-off of 2 principal components was chosen as >95% of the explained variance was encapsulated in the first two principal components.

## 3.4 Modeling

### 3.4.1 Mahalanobis Distance

Once the principal components of the data had been computed and the first two principal components were selected, minimum covariance determinate (MCD) estimator and maximum likelihood estimator (MLE) were used to estimate the covariance

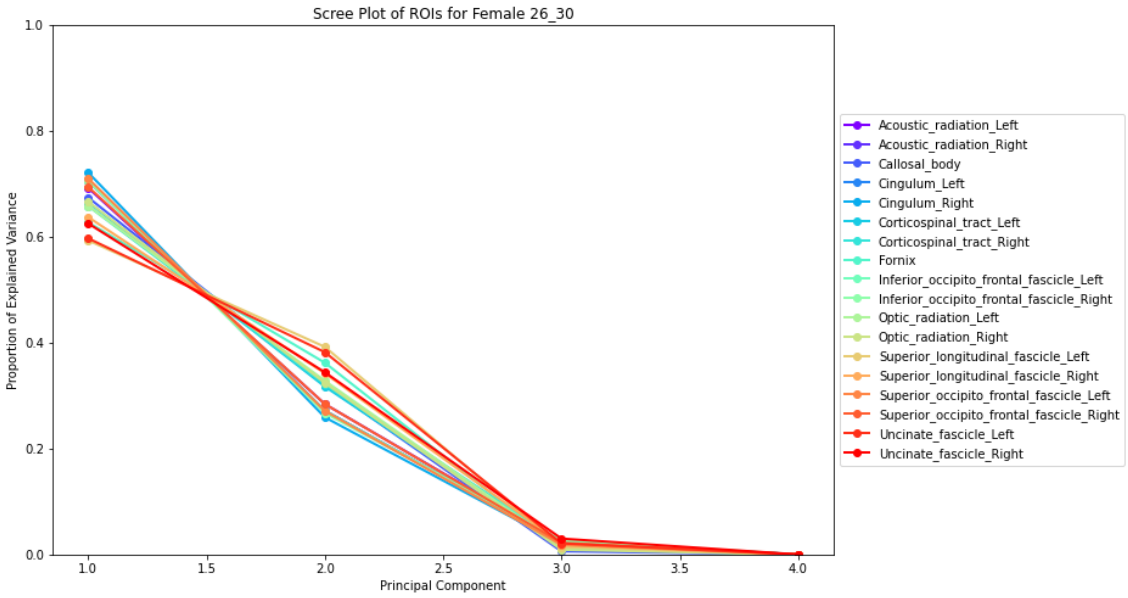



Figure 3.4: Scree plot of the ROIs for the Female 26 to 30 cohort

matrix and mean of the data. The subset of the data with the smallest determinant is desired as the determinant of the covariance matrix represents how broad of a distribution it has. The broader the distribution is, the more sparse the data is. The idea behind MCD is to find a "pure" subset of the data while minimizing the effects of any outliers in order to have the best estimate of the covariance matrix and the mean. Both MCD and MLE estimate the parameters of a distribution. However, MLE tends to be more sensitive to outliers within the data (Hardin and Rocke, 2005). The size of the subset created by MCD is determined by equation 3.4.1

$$\frac{n_{samples} + n_{features} + 1}{2} \tag{3.4.1}$$

The MCD was estimated using the function "MinCovDet" from the Python library "sklearn.covariance". The MLE was estimated using the function "EmpiricalCovariance" from the library "sklearn.covariance". The estimated covariance matrix and

| ROI | PC1 | PC2 | PC3 | PC4 |
|---|---|---|---|---|
| Acoustic_radiation_Left | 0.692 | 0.284 | 0.023 | 0.0 |
| Acoustic_radiation_Right | 0.705 | 0.271 | 0.024 | 0.0 |
| Callosal_body | 0.674 | 0.32 | 0.006 | 0.0 |
| Cingulum_Left | 0.707 | 0.272 | 0.021 | 0.0 |
| Cingulum_Right | 0.721 | 0.259 | 0.02 | 0.0 |
| Corticospinal_tract_Left | 0.665 | 0.317 | 0.018 | 0.0 |
| Corticospinal_tract_Right | 0.658 | 0.321 | 0.021 | 0.0 |
| Fornix | 0.627 | 0.362 | 0.011 | 0.0 |
| Inferior_occipito_frontal_fascicle_Left | 0.705 | 0.271 | 0.024 | 0.0 |
| Inferior_occipito_frontal_fascicle_Right | 0.706 | 0.267 | 0.027 | 0.0 |
| Optic_radiation_Left | 0.658 | 0.328 | 0.014 | 0.0 |
| Optic_radiation_Right | 0.666 | 0.324 | 0.009 | 0.0 |
| Superior_longitudinal_fascicle_Left | 0.593 | 0.392 | 0.014 | 0.0 |
| Superior_longitudinal_fascicle_Right | 0.637 | 0.342 | 0.021 | 0.0 |
| Superior_occipito_frontal_fascicle_Left | 0.71 | 0.271 | 0.018 | 0.0 |
| Superior_occipito_frontal_fascicle_Right | 0.694 | 0.284 | 0.022 | 0.0 |
| Uncinate_fascicle_Left | 0.597 | 0.382 | 0.021 | 0.0 |
| Uncinate_fascicle_Right | 0.625 | 0.344 | 0.03 | 0.0 |

Table 3.4: Table showing the explained variance ratios of the four principal components

mean from the MCD and MLE were then used to compute the Mahalanobis distance for every point by fitting them to the data using the function "fit" from the library "sklearn". The equation for Mahalanobis distance is given in Equation 3.4.2.

$$d_{(\mu,\Sigma)}(x_i)^2 = (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \tag{3.4.2}$$

Where $x_i$ is a sample from the dataset and $\mu$ is the mean, with $(x_i - \mu)$ representing the distance of an observation to the mean of the distribution. This is then multiplied by the inverse covariance matrix (or divided by the covariance matrix) $\Sigma$. In this case, both $\mu$ and $\Sigma$ are replaced by the estimations from either MCD or MLE. With the

Mahalanobis distances calculated, a cut off was set at an F1-score of 99.5%, classifying all outside data points as outliers. Contour plots were then made using the "contour" function from "matplotlib" showing the MCD and MLE based contours for the same ROI datasets, with the outliers classified by the MCD based Mahalanobis distance shown in Figure 3.5 in red. Plots showing the MCD and MLE based Mahalanobis distance for each subject were also plotted with the cutoff line shown in red. A comparison plot was also made to show the difference between the two estimators by showing which values were above and below the cutoff lines and the differences in classified outliers.

### 3.4.2 One Class Support Vector Machine Outlier Detector

The Mahalanobis distance decision functions were then used to compare to a One Class Support Vector Machine (OCSVM) outlier detector which were trained on each ROI dataset independently. Since OCSVM outlier detectors are an unsupervised machine learning technique, there is no ground truth in which to train and test your results with, making it impossible to automate tuning of the hyperparameters. Several OCSVMs were trained with a range of $\nu$ and $\gamma$ values and their decision functions were plotted in order to choose the best model. An increase in $\gamma$ increases the complexity of the decision function which can lead to an issue with over fitting, as shown in Figure 3.5

The parameter $\nu$ represents a bounded parameter that fine tunes the trade off between generalization and overfitting the data. $\nu$ is the upper bound for the number of samples to be excluded by the decision function, and the lower bound for the number of points that are support vectors. For example, a $\nu$ of 0.10 means that not

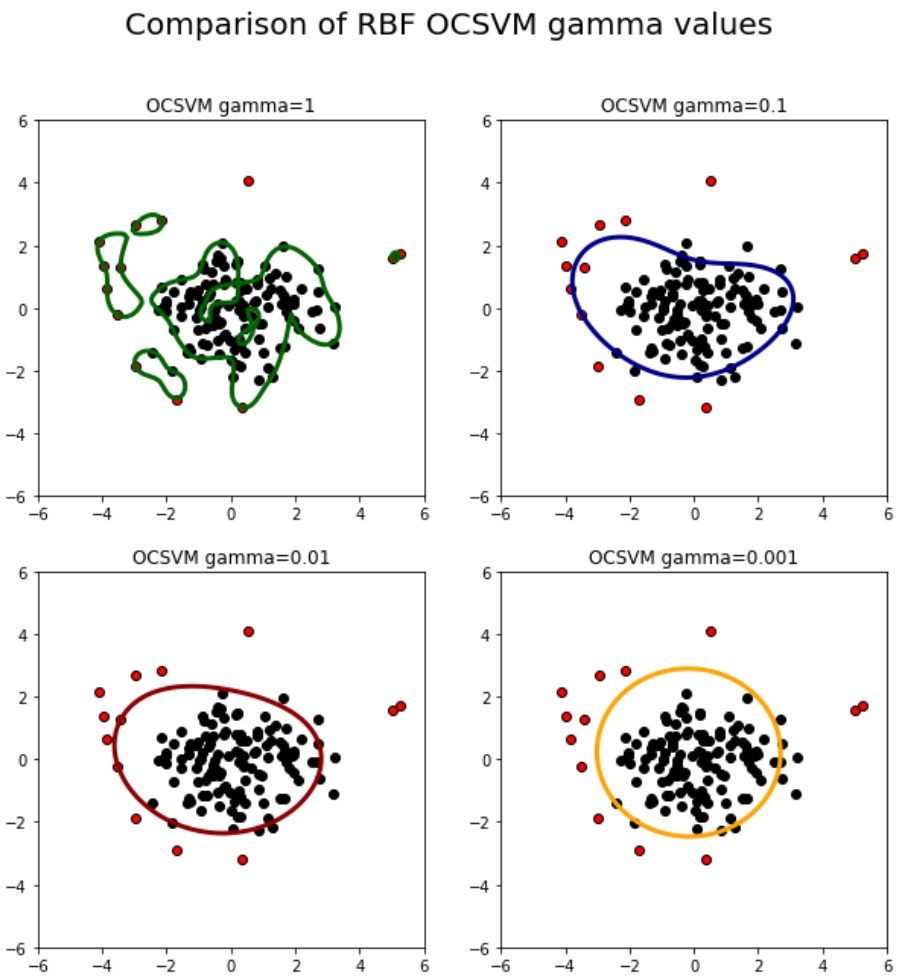



Figure 3.5: Comparison of 4 different $\gamma$ values used when training an RBF OCSVM with the same $\nu$ value, with the MCD based outliers shown in red

more than 10% of the samples can be excluded as outliers. The kernel chosen for the OCSVM was a radial basis function (RBF) due to the elliptical nature of the data. Linear, Polynomial and sigmoid kernels were also tested, however, their shapes did not fit the data as well as RBF, as shown in Figure 3.6. 3.6.

The OCSVMs were fitted to the PCA data using the function "OneClassSVM" from the library "sklearn.svm". After a series of tests to observe how the different $\gamma$ and $\nu$ values affected the decision function it was decided to make two separate

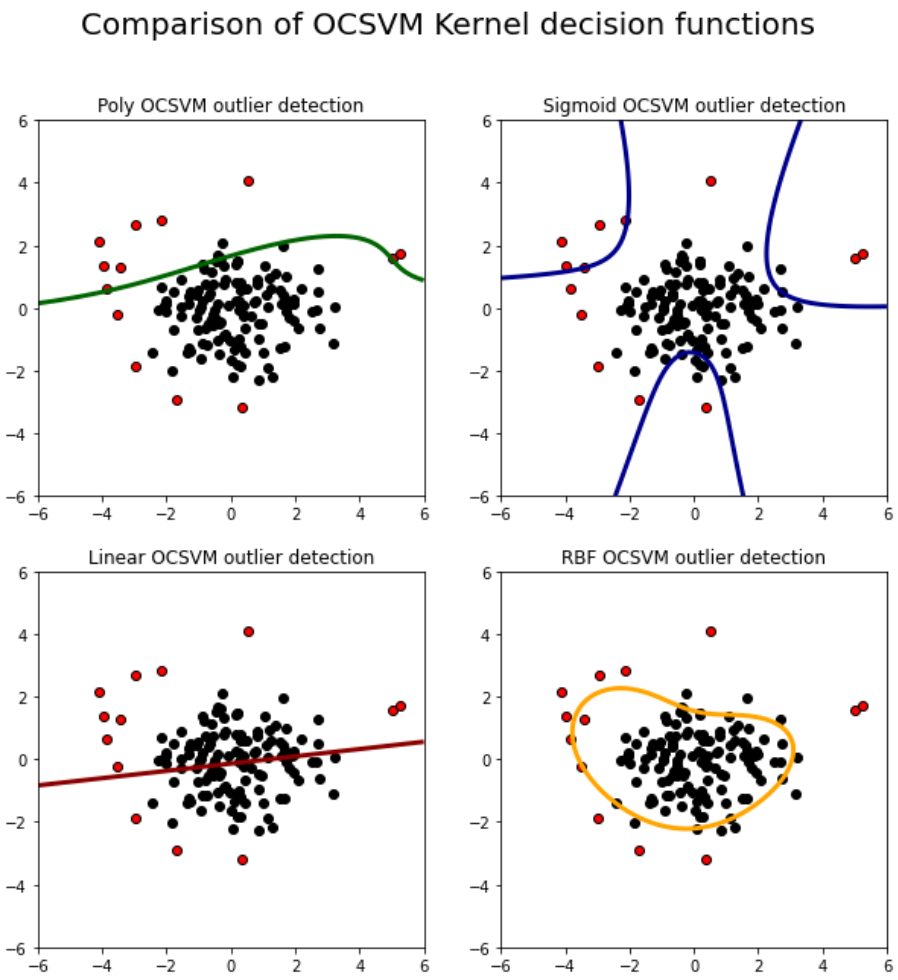



Figure 3.6: Comparison of 4 different OCSVM kernels, with the MCD based outliers shown in red

plots where one of the values was held constant while the other varied. Since each ROI dataset for each cohort contained different amounts of out-of-distribution data, there was no one set of parameter values which excluded outliers consistently. For the first plot, a $\nu$ value of 0.15 was selected while $\gamma$ values of 0.01, 0.05, and 0.1 were tested. All three OCSVM decision functions were plotted on a single figure with the MCD based outliers shown in red for reference, though these outlier labels were not used to train the models. For the second plot, a $\gamma$ value of 0.05 was selected while

$\nu$ values of 0.1 and 0.2 were tested along with a dynamic $\nu$ value calculated for each ROI dataset. The dynamic $\nu$ was calculated from the number of MCD based outliers, creating a pseudo novelty detector while still training the models in an unsupervised way. Since the $\nu$ value is used at the upper bound of how many samples are excluded by the decision function, a dynamic $\nu$ value was calculated using Equation 3.4.3

$$\nu_{dynamic} = \frac{n_{MCD-outliers}}{n_{subjects}} * 1.5 \tag{3.4.3}$$

Where $\nu_{dynamic}$ is the ratio of MCD based outliers to the number of subjects multiplied by a compensation constant. A compensation constant was added to allow the upper bound to go above the estimated ratio of outliers to allow for a wider range of excluded outliers. If in the case that no outliers were detected by the MCD based Mahalanobis distance, a $\nu$ value of 0.05 was chosen for the $\nu_{dynamic}$.

### 3.4.3 One Class Support Vector Machine Novelty Detector

A OCSVM novelty detector was trained using the "pure" subset of data determined by the MCD based Mahalanobis distance. Since a OCSVM novelty detector is only trained on "pure" data, the outliers determined by MCD were excluded from the training process. A 70/30 training/test split was chosen for the data, with the outliers being added to the testing split. The function "train_test_split" from the library "sklearn.model_selection" was used to split the inlier data. A OCSVM novelty detector decides whether something is in the class, giving it a label of +1, or outside of the class, giving it a label of -1. Therefore, all "pure" data from the MCD subset was given a label of +1 prior to training.

Hyperparameter tuning was done using the "GridSearchCV" function from the

library "sklearn.model_selection". The range of $\gamma$ values used during cross validation were 0.0005, 0.001, 0.01, 0.1 and 1. The range of $\nu$ values used were between 0.001 and 1, with a 0.001 step increment, resulting in 1000 $\nu$ values being tested. The scoring parameter chosen to evaluate the performance of the model was the "f1" score, which combines precision and recall. This scoring metric was chosen as it is a better measure of incorrectly classified data, and can be seen in Equation 3.4.4:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.4.4}$$

Where $TP$ is the number of true positives, $FP$ is the number of false positives, and $FN$ is the number of false negatives. A total of 5000 different parameters were tested and the best parameters from the cross validation stage were then saved and used to fit the OCSVM novelty detector to the training data using the output "best_params_" from the "GridSearchCV" function. The classification metrics were saved using the function "classification_report" from the library "sklearn.metrics". A countour plot was made showing the decision function using the "contour" function from the library "matplotlib".

# Chapter 4

# Results

## 4.1 Preprocessing

Image preprocessing was done according to the methodology described in Chapter 3. At several points throughout the preprocessing pipeline, visual inspection of the data was performed to ensure that any errors which might have occurred were not being propagated. After brains were extracted, eddy current and motion correction were performed, and the diffusion tensor was calculated, several subjects were chosen at random for inspection. The subject data was viewed using FSLeyes (McCarthy, 2021), an image viewer designed for visualizing neuroimaging data from the FSL toolbox, and inspected for any possible preprocessing errors. An example of one of these images can be seen in Figure 4.1.

After selected subjects were inspected and deemed clear of process errors, the brains were registered to the MNI152 standardized space and white-matter binary masks from the JHU white-matter tractography atlas were applied.

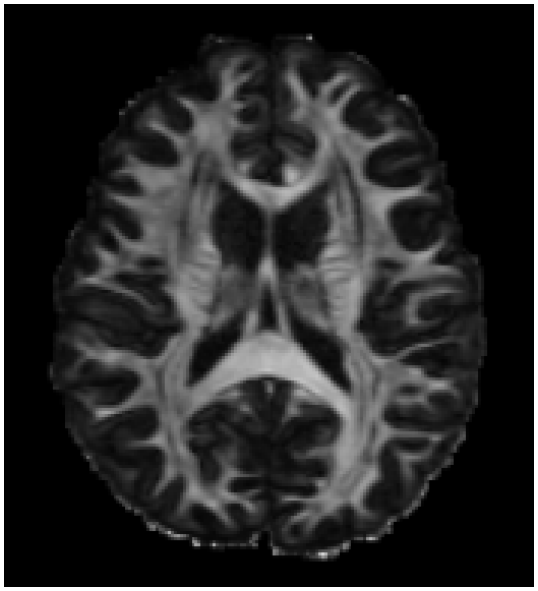


Figure 4.1: An axial slice fractional anisotropy (FA) parametric map of a subject in the Male 22 to 25 cohort. The FA is a scaler metric, ranging from 0 to 1, describing the shape of the tensor. A FA value of 0 indicates perfectly isotropic diffusion while a value of 1 represents an infinitely long cylinder with infinitely negligible cross section.

## 4.2 Feature Extraction

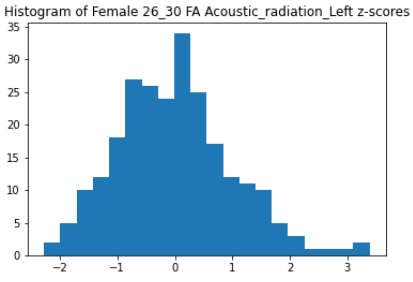
An example of histogram plots (standardized Female 26 to 30 cohort) produced using the methodology described in section 3.3.3 of Chapter 3 are seen in Figure 4.2. To reduce the number of figures in this section, the Female 31 to 35 cohort was chosen for the majority of examples. For each cohort a total of 72 histograms were produced, totalling 504 histograms for the 6 cohorts plus the synthesized outlier cohort. An example showing the histogram plots of the synthesized outlier cohort (Female 26 to 30 with added outliers) can be seen in Figure 4.3 with the synthesized outliers trailing off to the right of each histogram.

In the histogram plots shown in 3.3, subjects with a Z-score beyond the confidence interval cut off of 99.5% (Z-score of 2.807) can be seen in the majority of the histograms. This reinforces the belief that these healthy control datasets contain outliers.

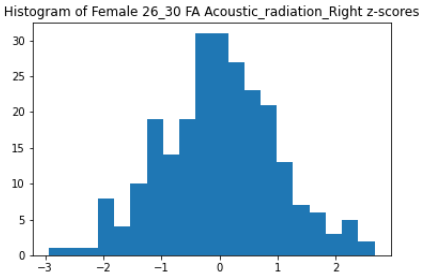
## 4.3 Z-score Outlier Detection

### 4.3.1 Model Validation With Synthesized Outliers

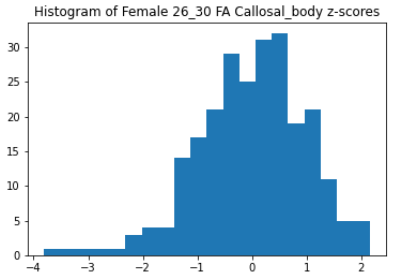
Z-score outlier detection was tested on its ability to identify synthesized outliers within a dataset. A total of 5 synthesized outliers were added to the data, however, as seen in Table 4.1, only some of the outliers were identified within each ROI. For the DTI metrics AD, RD and MD, the Z-score outlier detector was able to consistently identify 4 of the 5 outliers. Seeing as the synthesized outliers had a range of increasing outlyingness of %135, %170, %205, %240, and %275, it is understandable that the first outlier may not be detected with this method as the presence of extreme outliers



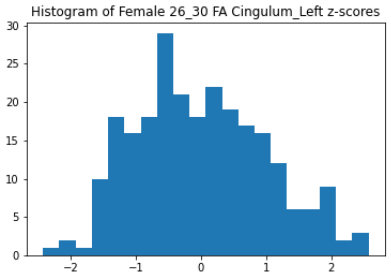

(a) Histogram of the Left Acoustic radiation ROI



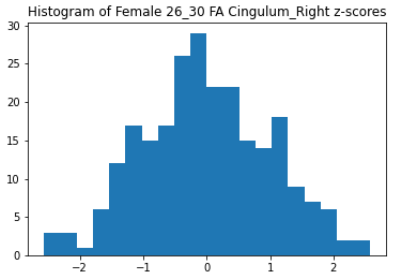

(b) Histogram of the Right Acoustic radiation ROI



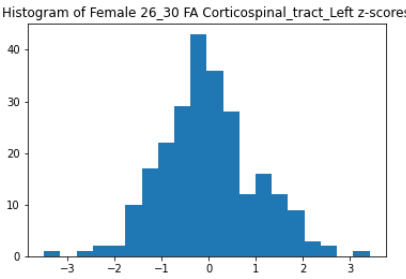

(c) Histogram of the Callosal body ROI




(d) Histogram of the Left Cingulum ROI

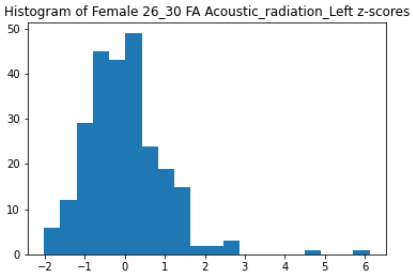



(e) Histogram of the Right Cingulum ROI

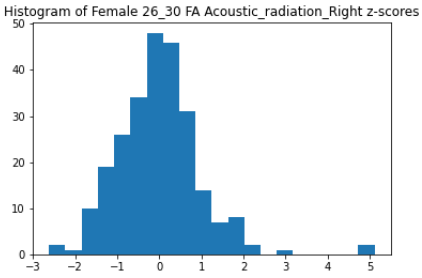



(f) Histogram of the Left Corticospinal tract ROI

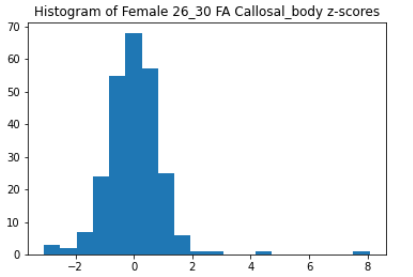
Figure 4.2: Histogram plots of FA ROI Z-scores from the Female 26 to 30 cohort



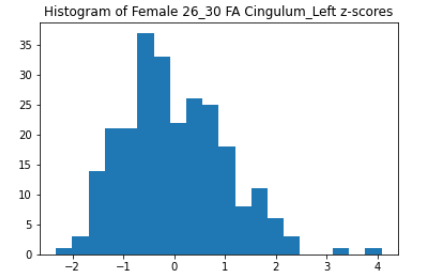

(a) Histogram of the Left Acoustic radiation ROI



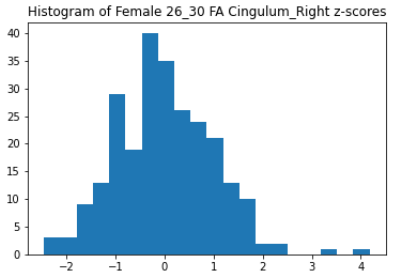

(b) Histogram of the Right Acoustic radiation ROI



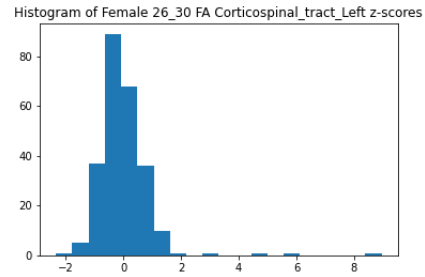

(c) Histogram of the Callosal body ROI




(d) Histogram of the Left Cingulum ROI




(e) Histogram of the Right Cingulum ROI




(f) Histogram of the Left Corticospinal tract ROI

Figure 4.3: Histogram plots of FA ROI Z-scores from the synthesized outlier containing Female 26 to 30 cohort

within the dataset can change the standard deviation, making less erroneous data harder to identify. However, for FA, the z-score outlier detector missed two or more of the outliers for the majority of the ROIs. This may be from the fact that the variance of the FA metric for the cohorts is orders of magnitude larger, $10^{-3}$, compared to the other DTI metrics, $10^{-7}$. The variance for each DTI metric was calculated using the function "var" from the library "pandas.DataFrame".

| **ROI** | **FA** | **AD** | **RD** | **MD** |
|---|---|---|---|---|
| Acoustic_radiation_Left | 3 | 4 | 4 | 4 |
| Acoustic_radiation_Right | 3 | 4 | 4 | 5 |
| Callosal_body | 3 | 4 | 4 | 4 |
| Cingulum_Left | 2 | 4 | 2 | 4 |
| Cingulum_Right | 2 | 4 | 3 | 4 |
| Corticospinal_tract_Left | 4 | 4 | 4 | 4 |
| Corticospinal_tract_Right | 4 | 4 | 4 | 4 |
| Fornix | 2 | 4 | 3 | 4 |
| Inferior_occipito_frontal_fascicle_Left | 2 | 4 | 4 | 4 |
| Inferior_occipito_frontal_fascicle_Right | 2 | 4 | 5 | 4 |
| Optic_radiation_Left | 3 | 4 | 4 | 4 |
| Optic_radiation_Right | 4 | 4 | 4 | 4 |
| Superior_longitudinal_fascicle_Left | 3 | 4 | 4 | 4 |
| Superior_longitudinal_fascicle_Right | 5 | 4 | 3 | 4 |
| Superior_occipito_frontal_fascicle_Left | 5 | 4 | 5 | 5 |
| Superior_occipito_frontal_fascicle_Right | 4 | 4 | 4 | 5 |
| Uncinate_fascicle_Left | 2 | 3 | 4 | 4 |
| Uncinate_fascicle_Right | 2 | 4 | 4 | 4 |

Table 4.1: Number of outliers per ROI identified by Z-score outlier detection in the Female 26 to 30 cohort with synthesized outliers out of 251 subjects.

### 4.3.2 Model Testing With Real Data

With Z-score outlier detection shown to be able to pick up the majority of synthesized outliers, the model was then tested on the 6 data cohorts. A table showing the number of outliers detected by Z-score thresholding in the Female 31 to 35 cohort can be seen in Table 4.2. Across the 4 DTI metrics for this cohort, at least one outlier was detected for each ROI, with some having up to 8. In other cohorts, such as the Male 26 to 30 cohort, up to 11 outliers were detected in a single ROI across the 4 DTI metrics. This further enforces the likelihood that there are anomalies within these multisite DTI databases. For the 6 cohorts, only 10 out of the 108 ROIs tested did not have an outlier detected, which is only about 9% of the total ROIs. A table showing the number of outliers detected for each ROI within each cohort can be seen in Table 4.3. Following the methodology outlined in Chapter 3, an example plot showing the number of ROI Z-scores beyond the cutoff for each subject is seen in Figure 4.4, with the plot identifying which DTI scalar metrics the Z-score is from. From the figure it is noted that several subjects within the dataset have 1 to 3 abnormal Z-scores, with a few subjects having multiple ROIs beyond the threshold.

| **ROI** | **FA** | **AD** | **RD** | **MD** |
|---|---|---|---|---|
| Acoustic_radiation_Left | 0 | 1 | 2 | 1 |
| Acoustic_radiation_Right | 1 | 2 | 1 | 1 |
| Callosal_body | 0 | 2 | 1 | 1 |
| Cingulum_Left | 1 | 1 | 2 | 0 |
| Cingulum_Right | 1 | 2 | 2 | 2 |
| Corticospinal_tract_Left | 0 | 1 | 0 | 1 |
| Corticospinal_tract_Right | 1 | 0 | 2 | 2 |
| Fornix | 0 | 2 | 0 | 0 |
| Inferior_occipito_frontal_fascicle_Left | 0 | 0 | 2 | 2 |
| Inferior_occipito_frontal_fascicle_Right | 0 | 1 | 0 | 0 |
| Optic_radiation_Left | 0 | 1 | 0 | 2 |
| Optic_radiation_Right | 1 | 3 | 0 | 0 |
| Superior_longitudinal_fascicle_Left | 2 | 1 | 2 | 0 |
| Superior_longitudinal_fascicle_Right | 1 | 0 | 1 | 1 |
| Superior_occipito_frontal_fascicle_Left | 2 | 2 | 2 | 2 |
| Superior_occipito_frontal_fascicle_Right | 2 | 1 | 2 | 2 |
| Uncinate_fascicle_Left | 0 | 3 | 1 | 0 |
| Uncinate_fascicle_Right | 2 | 2 | 2 | 2 |

Table 4.2: Number of outliers per ROI identified by Z-score outlier detection in the Female 31 to 35 cohort out of 232 subjects.

| **ROI** | M22_25 | M26_30 | M31_35 | F22_25 | F26_30 | F31_35 |
|---|---|---|---|---|---|---|
| Acoustic_radiation_Left | 5 | 6 | 0 | 0 | 7 | 4 |
| Acoustic_radiation_Right | 5 | 3 | 2 | 2 | 7 | 5 |
| Callosal_body | 5 | 3 | 2 | 1 | 9 | 4 |
| Cingulum_Left | 2 | 6 | 2 | 0 | 3 | 4 |
| Cingulum_Right | 2 | 7 | 0 | 1 | 4 | 7 |
| Corticospinal_tract_Left | 1 | 4 | 2 | 1 | 4 | 2 |
| Corticospinal_tract_Right | 0 | 5 | 2 | 2 | 6 | 5 |
| Fornix | 3 | 3 | 1 | 0 | 4 | 2 |
| Inferior_occipito_frontal_fascicle_Left | 6 | 1 | 1 | 3 | 2 | 4 |
| Inferior_occipito_frontal_fascicle_Right | 5 | 5 | 2 | 0 | 6 | 1 |
| Optic_radiation_Left | 1 | 4 | 7 | 0 | 7 | 3 |
| Optic_radiation_Right | 2 | 3 | 4 | 0 | 10 | 4 |
| Superior_longitudinal_fascicle_Left | 3 | 6 | 0 | 3 | 6 | 5 |
| Superior_longitudinal_fascicle_Right | 3 | 7 | 1 | 3 | 11 | 3 |
| Superior_occipito_frontal_fascicle_Left | 2 | 11 | 2 | 5 | 9 | 8 |
| Superior_occipito_frontal_fascicle_Right | 1 | 6 | 2 | 2 | 7 | 7 |
| Uncinate_fascicle_Left | 2 | 3 | 2 | 1 | 8 | 4 |
| Uncinate_fascicle_Right | 6 | 5 | 5 | 1 | 5 | 8 |
| **Cohort size** | 140 | 206 | 125 | 80 | 246 | 232 |

Table 4.3: Number of outliers detected for each ROI within each cohort of

## 4.4 Mahalanobis Outlier Detection

According to the methodology outlined in Chapter 3, PCA was performed and the eigenvalues of the principal components were plotted on a scree plot. For each of the 6 cohorts, over 95% of the explained variance was contained in the first 2 principal components. Feature extraction was used to remove the last two principal components. An example scree plot (Female 31 to 35 cohort) is seen in Figure 4.5 with the explained variance contained in each principal component shown in Table 4.4. With the features being reduced to two principal components, data visualization and decision functions become much simpler.

| **ROI** | **PC1** | **PC2** | **PC3** | **PC4** |
|---|---|---|---|---|
| Acoustic_radiation_Left | 0.701 | 0.271 | 0.028 | 0.0 |
| Acoustic_radiation_Right | 0.688 | 0.286 | 0.025 | 0.0 |
| Callosal_body | 0.647 | 0.348 | 0.005 | 0.0 |
| Cingulum_Left | 0.687 | 0.289 | 0.023 | 0.0 |
| Cingulum_Right | 0.711 | 0.266 | 0.023 | 0.0 |
| Corticospinal_tract_Left | 0.68 | 0.305 | 0.015 | 0.0 |
| Corticospinal_tract_Right | 0.683 | 0.298 | 0.019 | 0.0 |
| Fornix | 0.638 | 0.35 | 0.012 | 0.0 |
| Inferior_occipito_frontal_fascicle_Left | 0.709 | 0.264 | 0.026 | 0.0 |
| Inferior_occipito_frontal_fascicle_Right | 0.701 | 0.265 | 0.033 | 0.0 |
| Optic_radiation_Left | 0.629 | 0.359 | 0.012 | 0.0 |
| Optic_radiation_Right | 0.642 | 0.35 | 0.008 | 0.0 |
| Superior_longitudinal_fascicle_Left | 0.62 | 0.367 | 0.012 | 0.0 |
| Superior_longitudinal_fascicle_Right | 0.659 | 0.324 | 0.016 | 0.0 |
| Superior_occipito_frontal_fascicle_Left | 0.727 | 0.255 | 0.018 | 0.0 |
| Superior_occipito_frontal_fascicle_Right | 0.662 | 0.315 | 0.023 | 0.0 |
| Uncinate_fascicle_Left | 0.593 | 0.387 | 0.02 | 0.0 |
| Uncinate_fascicle_Right | 0.638 | 0.329 | 0.032 | 0.0 |

Table 4.4: Proportion of explained variance contained in each principal component for the Female 31 to 35 cohort

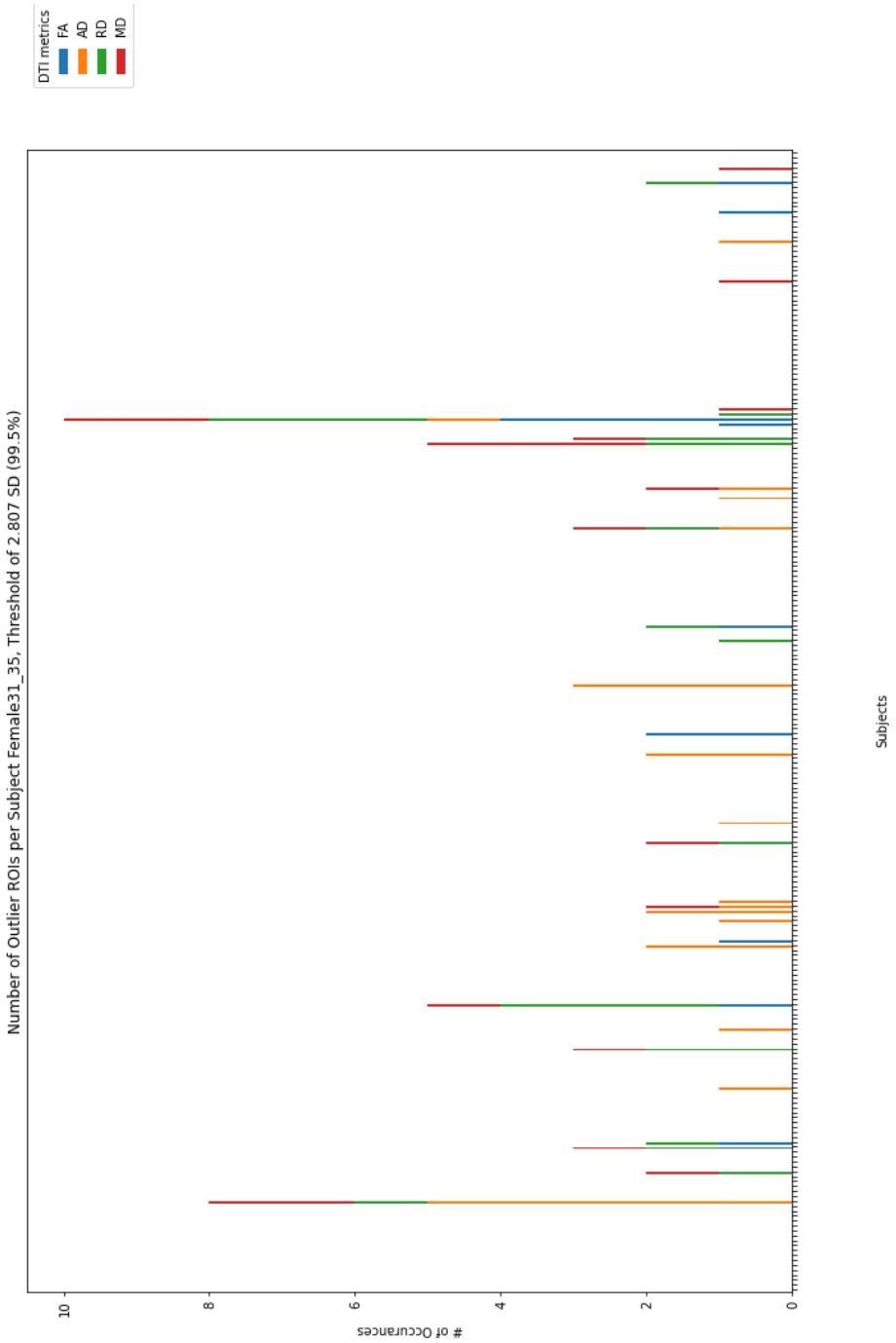



Figure 4.4: Plot showing the number of Z-scores per subject beyond the cutoff of 99.5% for the Female 31 to 35 cohort, out of 232 subjects.

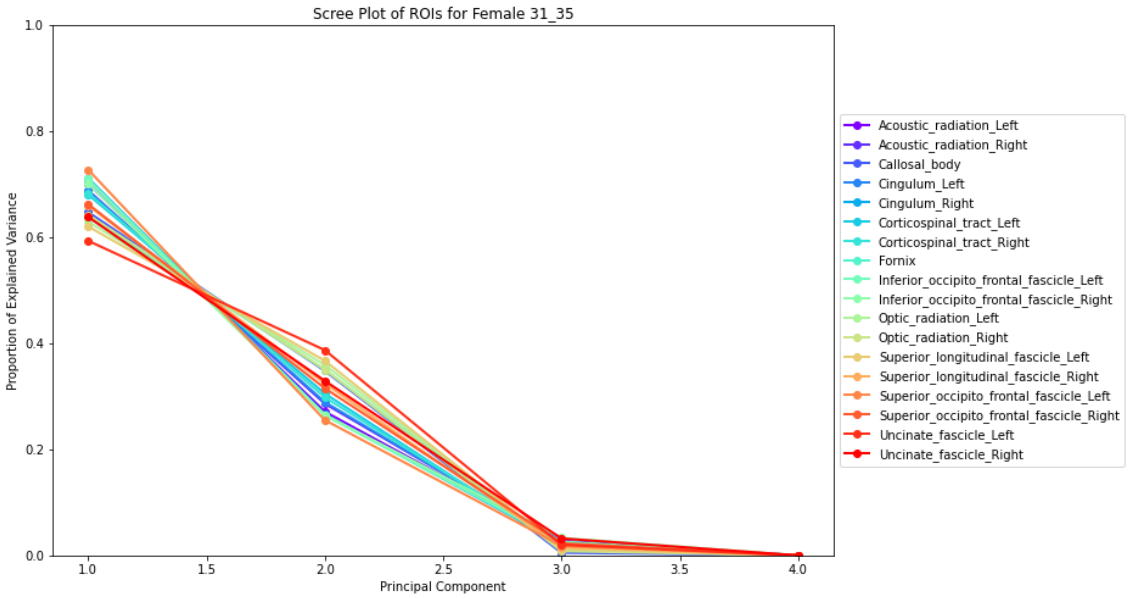



Figure 4.5: Scree plot of the eigenvalues of the Female 31 to 35 cohort, out of 232 subjects.

### 4.4.1 Model Validation With Synthesized Outliers

The Mahalanobis distances were calculated using the maximum likelihood estimator (MLE) and minimum covariance determinant (MCD) estimations, and contour plots were produced to examine how both estimators handled the synthesized mild to extreme outliers, as shown in Figure 4.6. The presence of extreme outliers heavily affected the covariance and mean estimation of MLE, while it had a noticeably smaller effect on the MCD. In Figure 4.6, the contours of the MLE based Mahalanobis distance, shown in blue, are unable to fit the distribution due to the outliers shown on the right. The MCD based Mahalanobis distance performs much better, ignoring the extreme outliers for the mean and covariance estimations.

The number of outliers detected by both estimator based Mahalanobis distance models is noted in Table 4.5. The MLE based detector was unable to accurately estimate the covariance and mean of the data, leading to a poor decision function for the 99.5% F-score threshold and lower detection rates as compared to MCD. Since the synthesized outliers were added to a dataset suspected to contain other anomalies, other erroneous datapoints were also identified by both models, especially for the MCD based Mahalanobis distance detector. The MLE was used to compare to the MCD for the rest of the cohorts, however, due to it's susceptibility to outliers, it was not considered for training a one-class support vector machine (OCSVM) for novelty detection later on.

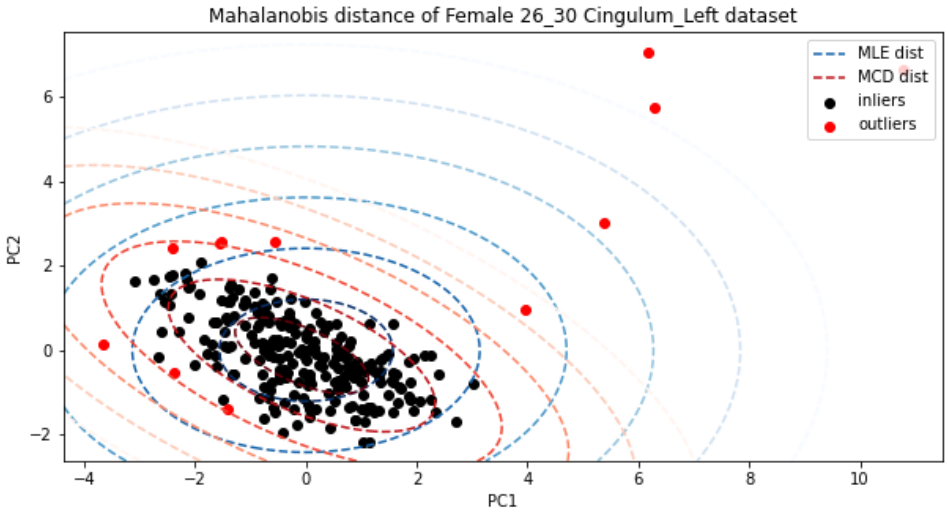



Figure 4.6: Contour plots of the MLE based and MCD based Mahalanobis distance decision functions. This example shows the Left Cingulum ROI in the synthesized outlier cohort, with the MCD based outliers shown in red.

| **ROI** | **MLE** | **MCD** |
|---|---|---|
| Acoustic_radiation_Left | 6 | 15 |
| Acoustic_radiation_Right | 4 | 13 |
| Callosal_body | 9 | 21 |
| Cingulum_Left | 4 | 12 |
| Cingulum_Right | 5 | 11 |
| Corticospinal_tract_Left | 7 | 12 |
| Corticospinal_tract_Right | 8 | 15 |
| Fornix | 6 | 15 |
| Inferior_occipito_frontal_fascicle_Left | 4 | 12 |
| Inferior_occipito_frontal_fascicle_Right | 6 | 12 |
| Optic_radiation_Left | 7 | 20 |
| Optic_radiation_Right | 6 | 17 |
| Superior_longitudinal_fascicle_Left | 8 | 11 |
| Superior_longitudinal_fascicle_Right | 8 | 15 |
| Superior_occipito_frontal_fascicle_Left | 8 | 17 |
| Superior_occipito_frontal_fascicle_Right | 6 | 11 |
| Uncinate_fascicle_Left | 6 | 13 |
| Uncinate_fascicle_Right | 5 | 11 |

Table 4.5: Number of outliers detected in the synthesized outlier dataset by MLE and MCD based Mahalanobis distance, out of 251 subjects.

### 4.4.2 Model Testing With Real Data

With the MLE and MCD based Mahalanobis distance outlier detectors tested on the synthesized outlier dataset, the models were then used to identify anomalies within other cohorts. Contour plots were produced for each of the ROIs comparing the estimators, as shown in Figure 4.7. As shown in the model validation stage, the presence of outliers within the data heavily affects the MLE estimations, leading to a poor decision function. In Figure 4.7, the MLE based contours, shown in blue, are spaced further apart due to poor estimations while attempting to include data points far from the distribution.

The Mahalanobis distance of each subject was then plotted with the exclusion line shown in red in order to observe the data points which were excluded by both models. Examples of these plots are demonstrated with Figure 4.8 and Figure 4.9. The MLE consistently identified fewer outliers within each ROI, for all of the cohorts. The MLE and MCD Mahalanobis distances were then plotted against each other to observe the differences in their exclusion boundaries, as shown in Figure 4.10. The bottom left quadrant shows data points which were deemed inliers by both models, the top right quadrant shows the data points deemed outliers by both models, and the top left quadrant shows data points deemed outliers by MCD based Mahalanobis distance thresholding.

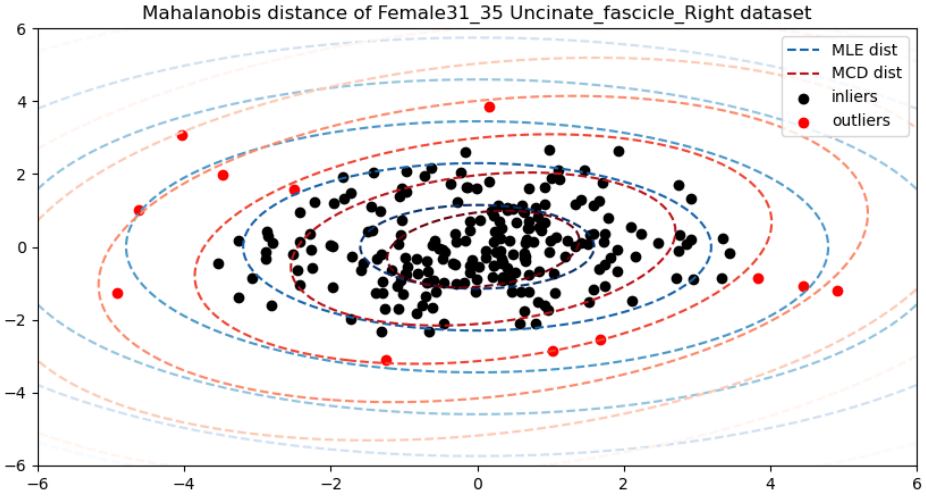



Figure 4.7: Contour plots of the MLE based and MCD based Mahalanobis distance decision functions. This example was for the right uncinate fascicle ROI in the Female 31 to 35 cohort, with the MCD based outliers shown in red

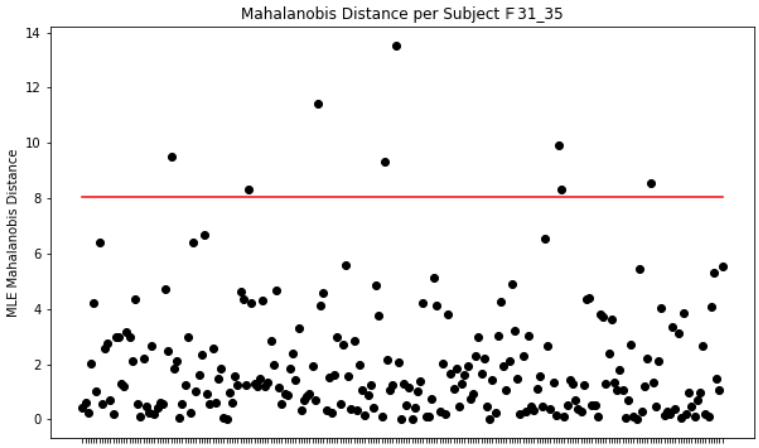



Figure 4.8: Plot showing the MLE based Mahalanobis distances for each subject for the Right corticospinal tract ROI in the Female 31 to 35 cohort, with the cutoff line shown in red

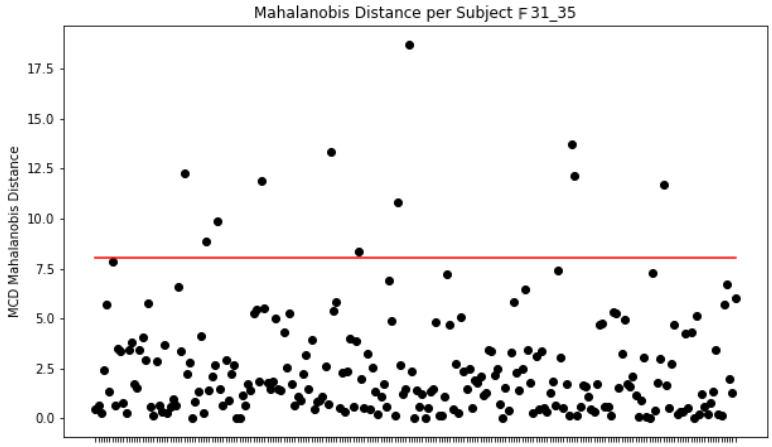



Figure 4.9: Plot showing the MCD based Mahalanobis distances for each subject for the Right corticospinal tract ROI in the Female 31 to 35 cohort, with the cutoff line shown in red

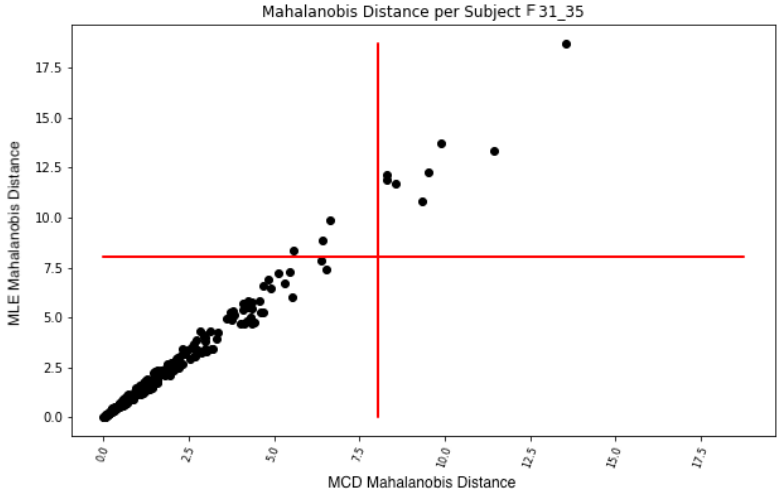



Figure 4.10: Plot showing the MLE based Mahalanobis distances vs. the MCD based Mahalanobis distance for each subject for the right corticospinal tract ROI in the Female 31 to 35 cohort, with the cutoff lines shown in red

The number of outliers identified by each model for the Female cohorts are listed in Table 4.6, while the same for the Male cohorts are in Table 4.7. In the majority of cases, MLE based outlier detection identified significantly fewer possible outliers when compared with MCD based detection. MCD, being the more robust estimator, was able to better estimate the parameters of the data, producing a more reliable exclusion boundary using the Mahalanobis distance.

| | **F22_25** | | **F26_30** | | **F31_35** | |
|---|---|---|---|---|---|---|
| **ROI** | MLE | MCD | MLE | MCD | MLE | MCD |
| Acoustic_radiation_Left | 1 | 7 | 6 | 13 | 4 | 6 |
| Acoustic_radiation_Right | 3 | 5 | 5 | 7 | 6 | 13 |
| Callosal_body | 1 | 1 | 4 | 7 | 5 | 7 |
| Cingulum_Left | 0 | 6 | 3 | 6 | 4 | 7 |
| Cingulum_Right | 0 | 5 | 4 | 5 | 3 | 6 |
| Corticospinal_tract_Left | 1 | 9 | 8 | 11 | 4 | 5 |
| Corticospinal_tract_Right | 2 | 7 | 2 | 5 | 8 | 11 |
| Fornix | 0 | 0 | 3 | 10 | 2 | 5 |
| Inferior_occipito_frontal_fascicle_Left | 2 | 8 | 2 | 12 | 3 | 10 |
| Inferior_occipito_frontal_fascicle_Right | 0 | 2 | 4 | 11 | 2 | 12 |
| Optic_radiation_Left | 1 | 2 | 4 | 5 | 4 | 11 |
| Optic_radiation_Right | 1 | 1 | 7 | 6 | 4 | 5 |
| Superior_longitudinal_fascicle_Left | 4 | 8 | 3 | 13 | 7 | 6 |
| Superior_longitudinal_fascicle_Right | 2 | 6 | 6 | 12 | 3 | 13 |
| Superior_occipito_frontal_fascicle_Left | 4 | 6 | 4 | 11 | 6 | 12 |
| Superior_occipito_frontal_fascicle_Right | 2 | 4 | 5 | 9 | 4 | 11 |
| Uncinate_fascicle_Left | 1 | 1 | 6 | 12 | 5 | 9 |
| Uncinate_fascicle_Right | 0 | 6 | 4 | 4 | 6 | 12 |

Table 4.6: Number of outliers identified by the MLE and MCD based Mahalanobis distance outlier detectors for the Female cohorts.

| | **M22_25** | | **M26_30** | | **M31_35** | |
|---|---|---|---|---|---|---|
| **ROI** | MLE | MCD | MLE | MCD | MLE | MCD |
| Acoustic_radiation_Left | 2 | 4 | 5 | 8 | 0 | 1 |
| Acoustic_radiation_Right | 3 | 5 | 4 | 9 | 3 | 8 |
| Callosal_body | 4 | 6 | 2 | 3 | 2 | 2 |
| Cingulum_Left | 3 | 6 | 4 | 13 | 1 | 3 |
| Cingulum_Right | 1 | 11 | 5 | 11 | 0 | 5 |
| Corticospinal_tract_Left | 2 | 2 | 3 | 6 | 1 | 4 |
| Corticospinal_tract_Right | 0 | 1 | 4 | 4 | 3 | 3 |
| Fornix | 2 | 4 | 3 | 6 | 1 | 1 |
| Inferior_occipito_frontal_fascicle_Left | 4 | 7 | 2 | 3 | 1 | 6 |
| Inferior_occipito_frontal_fascicle_Right | 3 | 4 | 3 | 7 | 3 | 4 |
| Optic_radiation_Left | 1 | 7 | 2 | 3 | 4 | 6 |
| Optic_radiation_Right | 2 | 5 | 3 | 10 | 2 | 5 |
| Superior_longitudinal_fascicle_Left | 2 | 11 | 5 | 7 | 2 | 2 |
| Superior_longitudinal_fascicle_Right | 2 | 12 | 4 | 7 | 2 | 2 |
| Superior_occipito_frontal_fascicle_Left | 1 | 2 | 9 | 14 | 0 | 7 |
| Superior_occipito_frontal_fascicle_Right | 3 | 9 | 5 | 12 | 1 | 6 |
| Uncinate_fascicle_Left | 5 | 6 | 5 | 8 | 2 | 4 |
| Uncinate_fascicle_Right | 5 | 13 | 5 | 9 | 3 | 4 |

Table 4.7: Number of outliers identified by the MLE and MCD based Mahalanobis distance outlier detectors for the Male cohorts.

## 4.5 One-Class Support Vector Machine Outlier Detector

### 4.5.1 Model Validation With Synthesized Outliers

OCSVM outlier detectors were tested on the synthesized outlier dataset to observe how mild to extreme outliers affect a models ability to form a decision function. Since hyperparameter tuning is not a viable option, as there is no ground truth to the data, a range of parameters were tested to observe which best fits the data. In order to

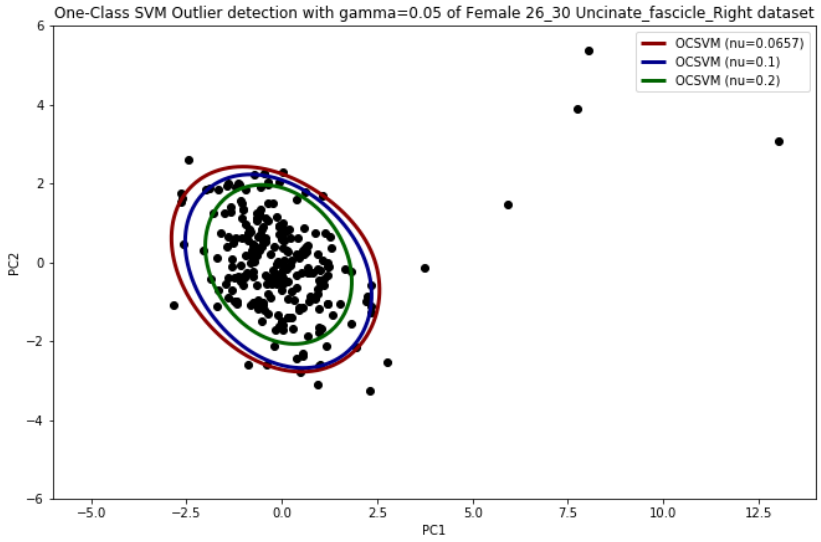



Figure 4.11: Decision boundaries of 3 OCSVMs with a range of $\nu$ values.

test different values of the $\nu$ and $\gamma$ parameters, two sets of plots were made where one parameter was held constant while a range of values were tested for the other. An example plot with a range of $\nu$ values is shown in Figure 4.11. As the $\nu$ value increases, the number of identified possible outliers increases since the parameter $\nu$ is the upper bound of the fraction of data points considered outliers. The dynamic $\nu$ value explained in Chapter 3 forms the largest decision function of the three OCSVMs. An example plot with a range of $\gamma$ values is displayed in Figure 4.14. The complexity of the decision function is increased as $\gamma$ increases.

Each of the models decision functions are unaffected by the presence of extreme outliers, allowing for exclusion of synthesized outlier data from the inliers. For each ROI, all of the OCSVMs were able to identify the synthesized outliers as anomalies.

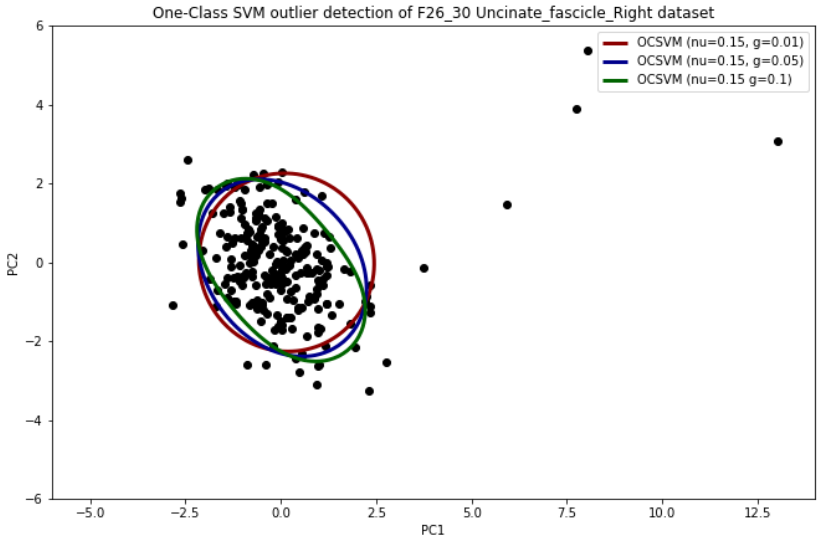



Figure 4.12: Decision boundaries of 3 OCSVMs with a range of $\gamma$ values.

### 4.5.2 Model Testing With Real Data

With the models validated on the synthesized outliers, the OCSVM outlier detectors were tested on the remaining cohorts to investigate the presence of anomalies within the data. The decision functions of the OCSVMs with varying $\nu$ values were plotted for each ROI from the cohorts (Figure 4.13). The list of parameters used is listed in Table 4.8, with the dynamic $\nu$ value being calculated according to approach outlined in Chapter 3. The ouliers in the figure, shown in red, are from the MCD based detector and were only used for visual reference as to how the OCSVM models exclude outliers compared to other methods. Out of the three OCSVMs, the model with the largest $\nu$ excluded the most amount of data points, shown in green, with the dynamic $\nu$ excluding the least, shown in dark red. Since the shape of the distributions and location of anomalies were unique for every ROI, there was no one set of parameter

combinations which best fit the data for every dataset. Upon visual inspection, the dynamic $\nu$ and $\nu = 0.1$ OCSVMs appeared to best fit the data for most cases. The number of outliers detected by each OCSVM model for the Female cohorts is in Table 4.12 and for the Male cohorts in Table 4.13. As noted, the OCSVM with $\nu = 0.2$ identified the most amount of possible outliers. However, when inspecting the decision function plots, such as Figure 4.13, the OCSVM is noted as excluding data points which should be deemed inliers.

Overall, the dynamic $\nu$ and $\nu = 0.1$ OCSVMs produced the most reasonable decision functions out of the three combinations tested. The best selection between these two models is dependant on the dataset being tested, as both models produced the best decision function for different ROIs. With hyperparameter tuning not being a viable option for unsupervised OCSVMs, one must rely on visual inspection of the decision functions to make decisions and conclusions.

A range of $\gamma$ values were tested and the decision functions of the OCSVMs were plotted for each ROI of each cohorts (Figure 4.14), with list of parameters used (Table 4.11). The three OCSVMs all detected similar numbers of possible outliers, as they shared the same $\nu$ value which acts as the upper bound of the fraction of outliers detected. Since the $\gamma$ parameter controls complexity of the decision function, $\gamma = 0.01$ OCSVM produced the simplest decision function, often excluding data points deemed inliers by other OCSVMs and outlier detectors as it could not capture the shape of the data. The $\gamma = 0.1$ OCSVM had the most complex decision function, which lead to oddly shaped decision functions in some cases which were overfitted and stretched to include data points deemed outliers by other methods. Of the three OCSVMs tested with a range of $\gamma$ values, $\gamma = 0.05$ produced the best fitting decision functions,

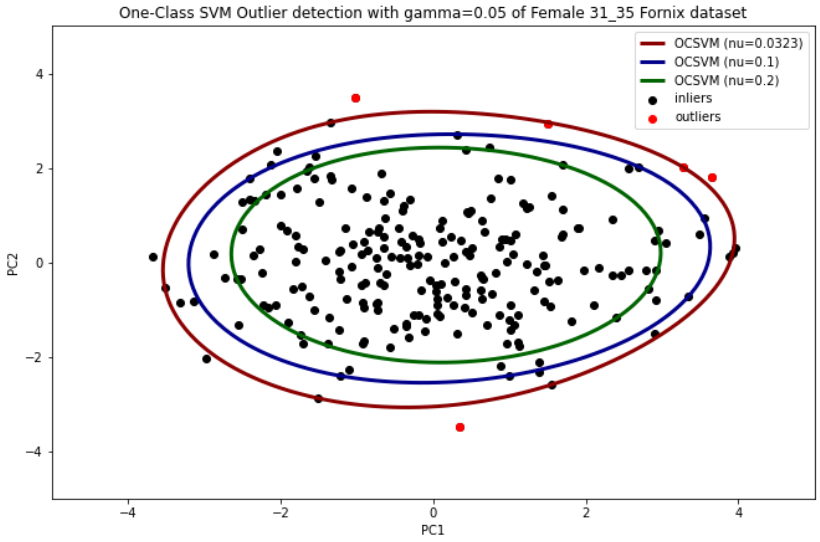



Figure 4.13: Decision boundaries of 3 OCSVMs with a range of $\nu$ values for the Fornix ROI of the Female 31 to 35 cohort, with the MCD based outliers shown in red for reference.

and classified similar data points as outliers similar to other methods tested. The number of outliers detected by each OCSVM model for the Female cohorts is seen in Table 4.12, while that for the Male cohorts is in Table 4.13.

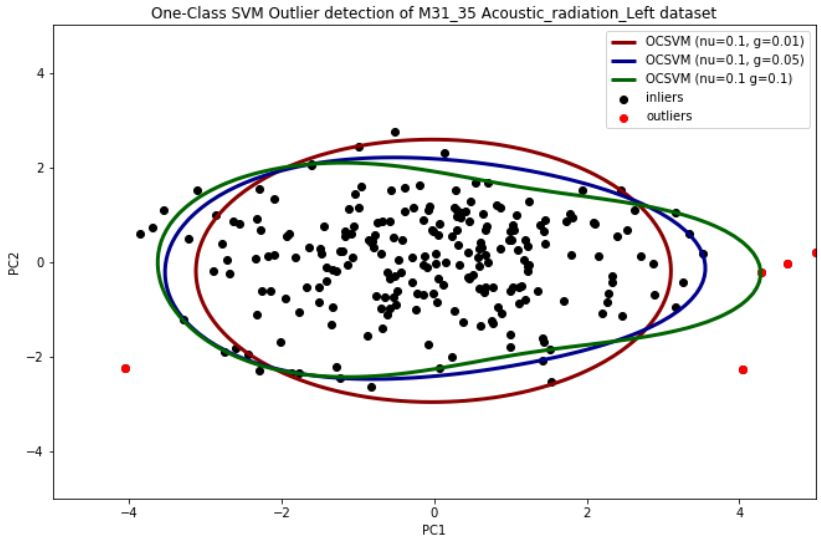



Figure 4.14: Decision boundaries of 3 OCSVMs with a range of $\gamma$ values for the Left Acoustic radiation ROI of the Female 31 to 35 cohort, with the MCD based outliers shown in red for reference.

| **Parameters** | OCSVM-dynnu | OCSVM-nu0.1 | OCSVM-nu0.2 |
|---|---|---|---|
| $\nu$ | $\frac{n_{MCDoutliers}}{n_{subjects}} * 1.5$ | 0.1 | 0.2 |
| $\gamma$ | 0.05 | 0.05 | 0.05 |
| Kernel | RBF | RBF | RBF |
| Tolerance | $1e^{-1}$ | $1e^{-1}$ | $1e^{-1}$ |

Table 4.8: OCSVM parameters for testing a range of $\nu$ values.

| ROI | F22_25 | | | F26_30 | | | F31_35 | | |
|---|---|---|---|---|---|---|---|---|---|
| | OCSVM dynnu | OCSVM nu0.1 | OCSVM nu0.2 | OCSVM dynnu | OCSVM nu0.1 | OCSVM nu0.2 | OCSVM dynnu | OCSVM nu0.1 | OCSVM nu0.2 |
| Acoustic_radiation_Left | 12 | 9 | 15 | 13 | 25 | 50 | 11 | 23 | 46 |
| Acoustic_radiation_Right | 7 | 7 | 15 | 12 | 26 | 50 | 20 | 23 | 46 |
| Callosal_body | 3 | 9 | 16 | 31 | 25 | 50 | 12 | 22 | 45 |
| Cingulum_Left | 9 | 8 | 16 | 12 | 26 | 48 | 10 | 24 | 46 |
| Cingulum_Right | 9 | 8 | 16 | 10 | 24 | 48 | 9 | 25 | 47 |
| Corticospinal_tract_Left | 12 | 8 | 14 | 15 | 27 | 50 | 7 | 23 | 47 |
| Corticospinal_tract_Right | 10 | 7 | 16 | 20 | 25 | 49 | 18 | 22 | 47 |
| Fornix | 3 | 8 | 15 | 16 | 25 | 50 | 9 | 24 | 47 |
| Inferior_occipito_frontal_fascicle_Left | 12 | 7 | 15 | 10 | 26 | 49 | 15 | 23 | 46 |
| Inferior_occipito_frontal_fascicle_Right | 4 | 9 | 16 | 9 | 23 | 49 | 19 | 25 | 46 |
| Optic_radiation_Left | 2 | 8 | 15 | 22 | 23 | 48 | 16 | 23 | 46 |
| Optic_radiation_Right | 4 | 7 | 16 | 21 | 25 | 50 | 8 | 23 | 46 |
| Superior_longitudinal_fascicle_Left | 12 | 9 | 15 | 8 | 26 | 50 | 8 | 25 | 46 |
| Superior_longitudinal_fascicle_Right | 9 | 6 | 16 | 17 | 26 | 48 | 18 | 25 | 45 |
| Superior_occipito_frontal_fascicle_Left | 9 | 7 | 16 | 21 | 26 | 48 | 18 | 24 | 47 |
| Superior_occipito_frontal_fascicle_Right | 6 | 7 | 16 | 12 | 23 | 49 | 16 | 23 | 47 |
| Uncinate_fascicle_Left | 1 | 8 | 15 | 13 | 23 | 50 | 13 | 23 | 47 |
| Uncinate_fascicle_Right | 9 | 8 | 15 | 7 | 24 | 48 | 18 | 24 | 46 |

Table 4.9: Number of outliers identified by each OCSVM for a range of $\nu$ values for the Female cohorts.

| ROI | M22_25 | | | M26_30 | | | M31_35 | | |
|---|---|---|---|---|---|---|---|---|---|
| | OCSVM dynnu | OCSVM nu0.1 | OCSVM nu0.2 | OCSVM dynnu | OCSVM nu0.1 | OCSVM nu0.2 | OCSVM dynnu | OCSVM nu0.1 | OCSVM nu0.2 |
| Acoustic_radiation_Left | 8 | 14 | 27 | 12 | 23 | 41 | 3 | 12 | 25 |
| Acoustic_radiation_Right | 8 | 12 | 28 | 13 | 21 | 41 | 11 | 13 | 26 |
| Callosal_body | 9 | 15 | 27 | 5 | 20 | 42 | 4 | 12 | 25 |
| Cingulum_Left | 8 | 15 | 27 | 19 | 21 | 40 | 4 | 12 | 26 |
| Cingulum_Right | 16 | 13 | 28 | 15 | 21 | 42 | 6 | 13 | 26 |
| Corticospinal_tract_Left | 2 | 15 | 28 | 8 | 22 | 41 | 5 | 13 | 24 |
| Corticospinal_tract_Right | 2 | 12 | 29 | 6 | 20 | 43 | 3 | 14 | 25 |
| Fornix | 5 | 13 | 28 | 9 | 22 | 43 | 2 | 12 | 25 |
| Inferior_occipito_frontal_fascicle_Left | 10 | 14 | 29 | 3 | 21 | 41 | 9 | 12 | 25 |
| Inferior_occipito_frontal_fascicle_Right | 5 | 14 | 28 | 10 | 21 | 41 | 7 | 15 | 25 |
| Optic_radiation_Left | 11 | 16 | 29 | 4 | 21 | 41 | 10 | 13 | 26 |
| Optic_radiation_Right | 6 | 15 | 28 | 17 | 20 | 41 | 7 | 13 | 25 |
| Superior_longitudinal_fascicle_Left | 16 | 12 | 29 | 9 | 20 | 41 | 2 | 11 | 24 |
| Superior_longitudinal_fascicle_Right | 18 | 15 | 28 | 10 | 20 | 40 | 1 | 12 | 24 |
| Superior_occipito_frontal_fascicle_Left | 3 | 15 | 30 | 21 | 22 | 43 | 10 | 12 | 25 |
| Superior_occipito_frontal_fascicle_Right | 13 | 13 | 27 | 18 | 20 | 40 | 9 | 12 | 24 |
| Uncinate_fascicle_Left | 9 | 14 | 26 | 12 | 22 | 41 | 8 | 12 | 24 |
| Uncinate_fascicle_Right | 19 | 14 | 28 | 12 | 20 | 42 | 7 | 13 | 26 |

Table 4.10: Number of outliers identified by each OCSVM for a range of $\nu$ values for the Male cohorts.

| **Parameters** | OCSVM-dynnu | OCSVM-nu0.1 | OCSVM-nu0.2 |
|---|---|---|---|
| $\nu$ | 0.1 | 0.1 | 0.1 |
| $\gamma$ | 0.01 | 0.05 | 0.1 |
| Kernel | RBF | RBF | RBF |
| Tolerance | $1e^{-1}$ | $1e^{-1}$ | $1e^{-1}$ |

Table 4.11: OCSVM parameters for testing a range of $\gamma$ values.

| | **F22_25** | | | **F26_30** | | | **F31_35** | | |
|---|---|---|---|---|---|---|---|---|---|
| **ROI** | OCSVM gamma0.01 | OCSVM gamma0.05 | OCSVM gamma0.1 | OCSVM gamma0.01 | OCSVM gamma0.05 | OCSVM gamma0.1 | OCSVM gamma0.01 | OCSVM gamma0.05 | OCSVM gamma0.1 |
| Acoustic_radiation_Left | 8 | 9 | 8 | 25 | 25 | 26 | 24 | 23 | 25 |
| Acoustic_radiation_Right | 8 | 7 | 9 | 25 | 26 | 24 | 24 | 23 | 23 |
| Callosal_body | 9 | 9 | 7 | 25 | 25 | 26 | 23 | 22 | 23 |
| Cingulum_Left | 8 | 8 | 6 | 25 | 26 | 23 | 24 | 24 | 22 |
| Cingulum_Right | 7 | 8 | 7 | 24 | 24 | 26 | 22 | 25 | 24 |
| Corticospinal_tract_Left | 8 | 8 | 9 | 24 | 27 | 27 | 23 | 23 | 24 |
| Corticospinal_tract_Right | 7 | 7 | 9 | 25 | 25 | 26 | 23 | 22 | 25 |
| Fornix | 9 | 8 | 7 | 25 | 25 | 24 | 22 | 24 | 22 |
| Inferior_occipito_frontal_fascicle_Left | 8 | 7 | 10 | 25 | 26 | 26 | 25 | 23 | 21 |
| Inferior_occipito_frontal_fascicle_Right | 8 | 9 | 9 | 24 | 23 | 25 | 24 | 25 | 23 |
| Optic_radiation_Left | 9 | 8 | 8 | 24 | 23 | 25 | 22 | 23 | 24 |
| Optic_radiation_Right | 8 | 7 | 7 | 25 | 25 | 26 | 24 | 23 | 22 |
| Superior_longitudinal_fascicle_Left | 9 | 9 | 8 | 25 | 26 | 26 | 24 | 25 | 23 |
| Superior_longitudinal_fascicle_Right | 8 | 6 | 8 | 23 | 26 | 24 | 23 | 25 | 24 |
| Superior_occipito_frontal_fascicle_Left | 8 | 7 | 7 | 25 | 26 | 24 | 22 | 24 | 25 |
| Superior_occipito_frontal_fascicle_Right | 9 | 7 | 9 | 24 | 23 | 23 | 23 | 23 | 23 |
| Uncinate_fascicle_Left | 7 | 8 | 8 | 24 | 23 | 24 | 24 | 23 | 22 |
| Uncinate_fascicle_Right | 7 | 8 | 8 | 25 | 24 | 25 | 23 | 24 | 22 |

Table 4.12: Number of outliers identified by each OCSVM for a range of $\gamma$ values for the Female cohorts.

| | **M22_25** | | | **M26_30** | | | **M31_35** | | |
|---|---|---|---|---|---|---|---|---|---|
| **ROI** | OCSVM gamma0.01 | OCSVM gamma0.05 | OCSVM gamma0.1 | OCSVM gamma0.01 | OCSVM gamma0.05 | OCSVM gamma0.1 | OCSVM gamma0.01 | OCSVM gamma0.05 | OCSVM gamma0.1 |
| Acoustic_radiation_Left | 13 | 14 | 15 | 20 | 23 | 20 | 12 | 12 | 13 |
| Acoustic_radiation_Right | 13 | 12 | 13 | 19 | 21 | 19 | 13 | 13 | 14 |
| Callosal_body | 13 | 15 | 15 | 22 | 20 | 19 | 12 | 12 | 13 |
| Cingulum_Left | 15 | 15 | 13 | 20 | 21 | 20 | 13 | 12 | 11 |
| Cingulum_Right | 14 | 13 | 16 | 20 | 21 | 21 | 11 | 13 | 11 |
| Corticospinal_tract_Left | 15 | 15 | 14 | 20 | 22 | 22 | 13 | 13 | 12 |
| Corticospinal_tract_Right | 15 | 12 | 14 | 20 | 20 | 20 | 12 | 14 | 15 |
| Fornix | 14 | 13 | 13 | 20 | 22 | 19 | 13 | 12 | 13 |
| Inferior_occipito_frontal_fascicle_Left | 14 | 14 | 14 | 21 | 21 | 20 | 13 | 12 | 12 |
| Inferior_occipito_frontal_fascicle_Right | 14 | 14 | 16 | 19 | 21 | 20 | 12 | 15 | 13 |
| Optic_radiation_Left | 13 | 16 | 13 | 21 | 21 | 21 | 12 | 13 | 13 |
| Optic_radiation_Right | 14 | 15 | 14 | 20 | 20 | 18 | 13 | 13 | 12 |
| Superior_longitudinal_fascicle_Left | 15 | 12 | 16 | 20 | 20 | 21 | 12 | 11 | 12 |
| Superior_longitudinal_fascicle_Right | 14 | 15 | 15 | 21 | 20 | 21 | 11 | 12 | 12 |
| Superior_occipito_frontal_fascicle_Left | 13 | 15 | 14 | 21 | 22 | 21 | 12 | 12 | 12 |
| Superior_occipito_frontal_fascicle_Right | 13 | 13 | 14 | 21 | 20 | 20 | 13 | 12 | 12 |
| Uncinate_fascicle_Left | 13 | 14 | 14 | 19 | 22 | 19 | 11 | 12 | 12 |
| Uncinate_fascicle_Right | 13 | 14 | 13 | 20 | 20 | 20 | 13 | 13 | 12 |

Table 4.13: Number of outliers identified by each OCSVM for a range of $\gamma$ values for the Male cohorts.

## 4.6 OCSVM Novelty Detection

The MCD based Mahalanobis distance inlier data was used to train a OCSVM novelty detector for each ROI, of each cohort. Since the novelty detector is only trained on in-class data, the inlier data was split 70/30 for training/testing. Hyperparameter tuning was performed using cross validation to optimize the model using $\gamma$ values of 0.0005, 0.001, 0.01, 0.1 and 1, and a range of $\nu$ values between 0.001 and 1, with a 0.001 step increment. The scoring metric chosen for cross validation was f1 score, as it is a reliable measure of incorrectly classified data. The best parameters and classification report for each ROI from cross validation were saved and a decision function was plotted. As shown above in sections 4.4 and 4.5, both MCD based Mahalanobis distance and OCSVMs are unaffected by the presence of extreme outliers within the data. Model validation was performed and all synthesized outliers were excluded from the MCD based training set as well as the OCSVM decision function.

An example of a decision function from a OCSVM novelty detector can be seen in Figure 4.15, with the training data shown in black, the testing data shown in green, and the outliers shown in red. The best parameters from cross validation for this model were $\gamma = 0.0005$ and $\nu = 0.014$, with the classification report shown in Table 4.14 and the confusion matrix shown in 4.16. The low $\gamma$ value of 0.0005 is most likely due to the fact that the OCSVM is being trained on the MCD based dataset, which is elliptical in shape. This introduces a bias which is learned by the novelty detector and influences a low complexity decision function, and in this case, leads to the inclusion of outliers within the decision function as seen in 4.15.

Since this is a heavily unbalanced dataset with only 12 possible outliers in this ROI out of 232 subjects (only 5.1% of the data), the weighted averages of the scores

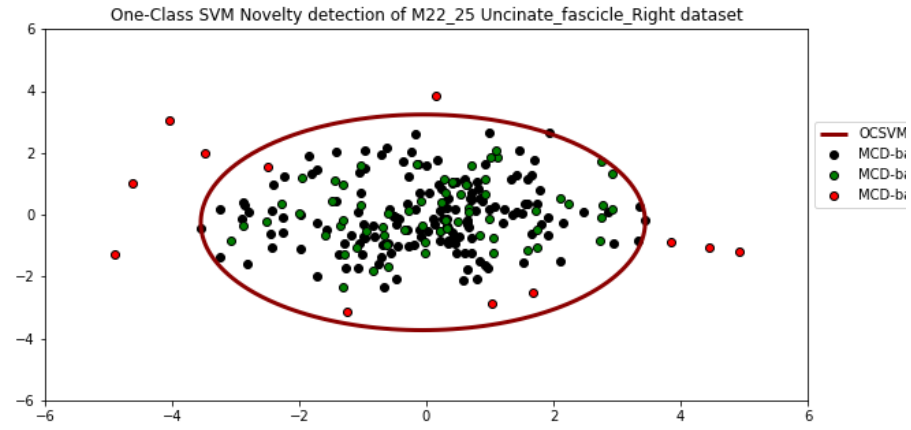



Figure 4.15: Decision function of the OCSVM novelty detector for the Right Uncinate fascicle from the Female 31 to 35 cohort.

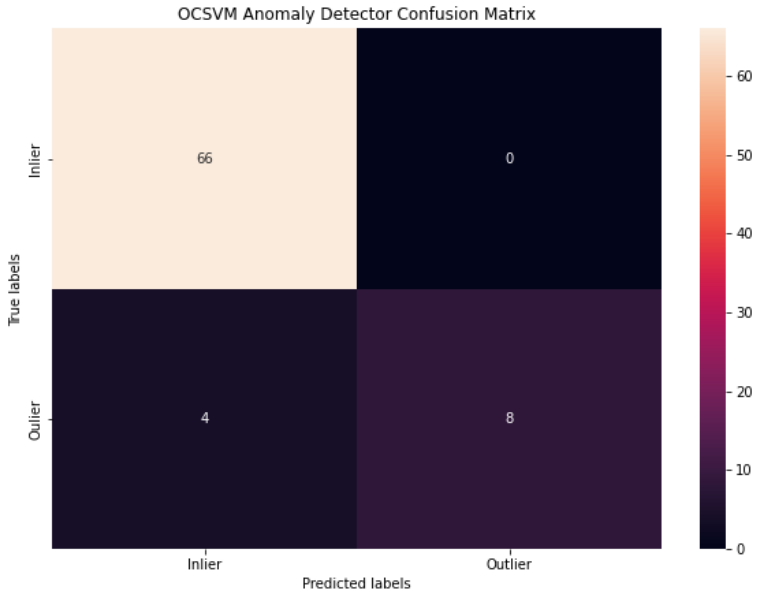



Figure 4.16: Confusion matrix of the OCSVM novelty detector for the right uncinate fascicle from the Female 31 to 35 cohort.

are the best representation of how the model performs. The weighted averages, unlike the macro averages, multiplies the class scores for each metric by the fraction of the data which that class makes up. The OCSVM novelty detector performed exceptionally well on the testing data, as shown in Table 4.15. Though these scores do not accurately represent how the novelty detector would behave on all data, as the classification rates are dependant on the class labels given by the MCD based detector which the model is also trained on, it shows great promise for the use of OCSVMs for novelty detection. For all cohorts, similar scores were observed in the classification reports. The most common parameters determined by cross validation were $\gamma = 0.0005$ and a $\nu$ value between 0.01 and 0.04. The $\nu$ values determined by cross validation for OCSVM novelty detection relate closest to the dynamic $\nu$ values used for OCSVM outlier detection.

| | Precision | Recall | f1-score | Support |
|---|---|---|---|---|
| -1 | 1.00 | 0.67 | 0.80 | 12 |
| 1 | 0.94 | 1.00 | 0.97 | 66 |
| Accuracy | | | 0.95 | 78 |
| Macro avg | 0.97 | 0.83 | 0.89 | 78 |
| Weighted avg | 0.95 | 0.95 | 0.94 | 78 |

Table 4.14: Classification report of the OCSVM novelty detector for the example ROI, the right uncinate fascicle, from the Female 31 to 35 cohort.

| | Weighted Precision | Weighted Recall | Weighted f1-score |
|---|---|---|---|
| Acoustic_radiation_Left | 0.98 | 0.97 | 0.97 |
| Acoustic_radiation_Right | 0.91 | 0.91 | 0.91 |
| Callosal_body | 0.96 | 0.96 | 0.95 |
| Cingulum_Left | 0.96 | 0.96 | 0.95 |
| Cingulum_Right | 0.96 | 0.96 | 0.95 |
| Corticospinal_tract_Left | 0.98 | 0.97 | 0.98 |
| Corticospinal_tract_Right | 0.93 | 0.92 | 0.91 |
| Fornix | 0.95 | 0.92 | 0.93 |
| Inferior_occipito_frontal_fascicle_Left | 0.96 | 0.96 | 0.96 |
| Inferior_occipito_frontal_fascicle_Right | 0.90 | 0.91 | 0.90 |
| Optic_radiation_Left | 0.85 | 0.87 | 0.86 |
| Optic_radiation_Right | 0.99 | 0.99 | 0.99 |
| Superior_longitudinal_fascicle_Left | 0.99 | 0.99 | 0.99 |
| Superior_longitudinal_fascicle_Right | 0.99 | 0.99 | 0.99 |
| Superior_occipito_frontal_fascicle_Left | 0.99 | 0.99 | 0.99 |
| Superior_occipito_frontal_fascicle_Right | 0.97 | 0.96 | 0.96 |
| Uncinate_fascicle_Left | 0.99 | 0.99 | 0.99 |
| Uncinate_fascicle_Right | 0.95 | 0.95 | 0.94 |

Table 4.15: Weighted classification scores for the Female 31 to 35 cohort.

# Chapter 5

# Discussion

Chapters 3 and 4 outline the methodology and results from various outlier and novelty detectors validated and tested on brain DTI datasets for the detection of anomalies. Z-score outlier detection was the simplest method of anomaly detection, however, the presence of extreme outliers during the validation stage affected the standard deviation of the distribution leading to the model missing synthesized outliers. The MLE based Mahalanobis distance was heavily affected by the presence of extreme outliers, leading to poor estimations of the covariance and mean of the distribution. The errors in these estimations lead to inaccurate identification of possible outliers within the data as the contours of the Mahalanobis distance did not fit the shape of the distribution. The MCD based Mahalanobis distance had a far more robust estimator, and its ability to find a pure subset within the data allowed for a more accurate estimation of the covariance and mean of the distribution in the presence of extreme synthesized outliers. The contours of the MCD based Mahalanobis distance outlier detector better fit the distributions than the MLE based detector and was able to consistently identify possible outliers in the data.

OCSVM outlier detection was shown to have the ability to detect outliers in the DTI brain datasets in an unsupervised way. However, due to the lack of automated hyperparameter tuning, it became a cumbersome method requiring visual inspection of multiple combinations of hyperparameter values for each ROI. Parameters which performed well on separating erroneous data in some ROIs did not perform well on others. The dynamic $\nu$ value provided a method for tailoring the upper bound of the fraction of outliers for each ROI. But, it required information from another form of outlier detection. On its own, OCSVM outlier detection does not seem promising as an unsupervised anomaly detector for use in brain MRI data. It became apparent that the more important parameter for tuning to use a OCSVM outlier detector was $\nu$, as changing the value of $\gamma$ while maintaining the value of $\nu$ lead to similar numbers of outliers detected as the fraction is bounded by $\nu$. The models tested with a range of $\gamma$ values struggled to classify a certain number of outliers as determined by the $\nu$ value, leading to some poor decision functions of varying complexity which did not fit most distributions well.

The OCSVM novelty detector had exceptional weighted precision, recall, and f1 scores. However, the classification was based on the output of the MCD based Mahalanobis distance and did not accurately reflect how the model may work on new data that had not been classified by MCD based Mahalanobis distance. The precision and recall scores of 0.99 were suspiciously high and were most likely the result of the novelty detector learning the decision function of the MCD based Mahalanobis distance, as well as a lack of accurately labeled testing data. A wide range of $\gamma$ values were used during the cross validation stage to test whether a higher complexity decision function would best fit the data, however, it was heavily biased by the shape

of the decision function used to classify the subset of inlier data leading to a low $\gamma$ value being selected as the best parameter. OCSVMs show promise as a form of novelty detection for DTI brain datasets, and, these are best fit for situations where new data is being added to a clean dataset. An example of this could be the situation of adding data to the same sex and age cohort from another multisite database to an existing dataset. Instead of performing outlier detection, requiring the analysis of all existing data along side the new data, a trained OCSVM novelty detector could be used to perform novelty detection on only the new data being added.

Although some of the methods tested out performed the others, there was a clear sign that these large, multisite DTI brain datasets contain erroneous data which do not fit the distribution. If used in a classification setting without the removal of these outliers, class separation might become more difficult as data points far outside the distribution could be carrying the incorrect class label. Though the number of outliers per subject is difficult to determine, as each method identified different amounts, there were several data points per ROI which were identified as anomalies by most methods. Outlier detection prior to classification, and even prior to the assembly of multisite datasets, is a necessary part of the preprocessing pipeline which must be included. The simplest and most reliable form beyond Z-score outlier detection would be MCD based Mahalanobis distance, as the robust estimator is unaffected by the presence of outliers in the data, and the Mahalanobis distance outlier detector is based on the point distance from the center of the distribution.

# Chapter 6

# Conclusion

In this thesis, unsupervised machine learning algorithms were used to investigate the presence of anomalies in supposedly healthy brain DTI datasets. The algorithms were tested on their ability to detect synthesized outliers before being utilized for anomaly detection on 6 different datasets. Four different algorithms were used for anomaly detection, including Z-score outlier detection, MLE and MCD based Mahalanobis distance outlier detection, OCSVM outlier detection, and OCSVM novelty detection trained on MCD based Mahalanobis distance data. It was found that the best method for outlier detection on this data was MCD based Mahalanobis distance. Furthermore, each outlier detection method was able to identify erroneous data, many of which consisted of the same data points. The results from this study reinforce the need for outlier or novelty detection as part of a preprocessing pipeline as anomalies were detected in each cohort.

## 6.1 Future Work

With the study concluding that anomaly detection is necessary for working with these multisite datasets, future work is needed to further investigate the presence of erroneous data. This includes using additional information from the ROIs in addition to the mean, such as the min and max of the voxels, to better understand the range of values in each region. Gathering more information from the ROIs may provide additional insight useful for detection of anomalies. For Mahalanobis distance calculation, additional estimators such as the MM-estimator and the Orthogonalized Gnanadesikan–Kettenring estimator could be tested. Additionally, in order to avoid having to perform MCD based Mahalanobis distance in order to calculate the dynamic $\nu$ value used in OCSVM outlier detection, the formula could be adjusted to use the number of outliers detected by the Z-score as standardization is necessary prior to performing PCA.

For additional methods testing, subjects from one of the online databases with varying levels of disease progression (for example, the three stages of Alzheimer's disease) could be used as anomaly data to observe how the models classify them. Furthermore, healthy control data from other online databases could be used to test how the OCSVM novelty detector behaves with new data, as well as adding the subjects with varying levels of disease as labeled outliers for further testing of the model.

## 6.2 Limitations

During this study, one of the biggest limitations was the lack of an expertly analyzed dataset with all possible outliers removed. Though the purpose of the study was to investigate large, multisite, brain DTI datasets containing healthy controls for the presence of anomalies, there lacked a concrete way of comparing the unsupervised models performance prior to exploratory analysis. The creation of synthesized outliers helped validate that the models were able to identify anomalies and observe how they behaved in the presence of highly erroneous data. However, the datasets which they were added to still contained anomalies.

Another limitation of the study was the need for visual inspection of the plots to determine whether the decision functions fit the data. With 6 cohorts and 18 ROIs, visually inspecting each plot became a time consuming process. Specifically for the OCSVM outlier detectors, comparing the decision functions between multiple parameter choices was difficult as the best parameters differed from one ROI to the next. The number of outliers detected for these models were not always representative of the amount of erroneous data as some of the decision functions excluded data points which were deemed inliers by all other methods.

# Appendix A

# Code

Your appendix goes here.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

@author: neilmacphee
"""

#%% Import libraries
    ###############################################################

import numpy as np
import pandas as pd
```

```
np.set_printoptions(precision=2, suppress=True)
import scipy
import os
import math
import nibabel as nib
from matplotlib import pyplot as plt
from matplotlib import cm
#from nibabel.testing import data_path

np.random.seed(777)

### Search directories for NifTi files
    ##########################################
#os.chdir('/Users/neilmacphee/Dropbox/Grad_School/Grad_School
    /Data/M22_25_proc/') #MacOS
os.chdir('/home/macphena/brainDATA/DTI_CSV') #radon clean
    data
#os.chdir('/home/macphena/brainDATA/DTI_CSV_out') #radon
    OUTLIER
#os.chdir('/Users/neilmacphee/Dropbox/Grad_School/Grad_School
    /Data/DTI_CSV') #MacOS

### Create file lists and dictionary
pathlist = list()
```

```
filelist = list()
dirslist = list()
oddlist = list()
#oddstore = dict()
datastore = dict()
cool = 0

#%% Import Data
    ##########################################################################

# import StandardScaler from sklearn
from sklearn.preprocessing import StandardScaler

DTI_mods = ['FA', 'AD', 'RD', 'MD']
#ROIs = ['Acoustic_radiation_L', 'Acoustic_radiation_R', '
    Callosal_body', 'Cingulum_L', 'Cingulum_R', '
    Corticospinal_tract_L', 'Corticospinal_tract_R', 'Fornix',
     'Inferior_occipito-frontal_fascicle_L', '
    Inferior_occipito-frontal_fascicle_R', 'Optic_radiation_L
    ', 'Optic_radiation_R', 'Superior_longitudinal_fascicle_L
    ', 'Superior_longitudinal_fascicle_R', '
    Superior_occipito_frontal_fascicle_L', 'Superior_occipito-
    frontal_fascicle_R', 'Uncinate_fascicle_L', '
    Uncinate_fascicle_R']
```

```
ROIs = ['Acoustic_radiation_Left', 'Acoustic_radiation_Right'
    , 'Callosal_body', 'Cingulum_Left', 'Cingulum_Right', '
    Corticospinal_tract_Left', 'Corticospinal_tract_Right', '
    Fornix', 'Inferior_occipito_frontal_fascicle_Left', '
    Inferior_occipito_frontal_fascicle_Right', '
    Optic_radiation_Left', 'Optic_radiation_Right', '
    Superior_longitudinal_fascicle_Left', '
    Superior_longitudinal_fascicle_Right', '
    Superior_occipito_frontal_fascicle_Left', '
    Superior_occipito_frontal_fascicle_Right', '
    Uncinate_fascicle_Left', 'Uncinate_fascicle_Right']
ages = ['22_25', '26_30', '31_35', '36_40']
df_odd = pd.DataFrame()
sex = ['Male','Female']

for j in range(0,len(ROIs)):
    # read in CSV file of ROI means and remove unwanted first
        column
    #df_DTI = pd.read_csv('Female_{}_{}.csv'.format(ages[0],
        ROIs[j]))
    df_DTI = pd.read_csv('Male_{}_{}.csv'.format(ages[0],ROIs
        [j]))
    df_DTI = df_DTI.rename(columns={'Unnamed:_0': 'Subject'})
```

```
#df_DTIF = pd.read_csv('Female_{}_{}.csv'.format(ages[3],
    ROIs[j]))
#df_DTIF = df_DTIF.rename(columns={'Unnamed: 0': 'Subject
    '})
#df_DTI = df_DTI.append(df_DTIF)
df_DTI = df_DTI.set_index([pd.Index(range(0,len(df_DTI)))
    ])
for lmao in range(0,len(df_DTI['Subject'])):
    df_DTI.loc[lmao, 'Subject'] = ('M{}'.format(lmao))

# Z-Scores
    ##########################################################################



# create 'to be scaled' dataframe without subject IDs
df_tbs = df_DTI.drop('Subject', axis=1)
# scale features of dataframe
scaled_features = StandardScaler().fit_transform(df_tbs.
    values)

# Add index and column names as well as subject IDs back
    into dataframe
df_scaled = pd.DataFrame(scaled_features, index=df_tbs.
    index, columns=df_tbs.columns)
```

```
df_scaled['Subject'] = df_DTI['Subject']
df_scaled = df_scaled[['Subject', 'FA', 'AD', 'RD', 'MD'
    ]]
#df_scaled = df_scaled[['FA', 'AD', 'RD', 'MD']]



# Plot histograms of z-scores
for i in range(0,len(DTI_mods)):
    # add subjects with a STD > 2 to a list
    for n in range(0,len(df_scaled[:])):
        if (df_scaled.loc[n,DTI_mods[i]] >= 2.807) or (
            df_scaled.loc[n,DTI_mods[i]] <= -2.807):
            oddlist.append('{} {} {}'.format(df_scaled.
                loc[n,'Subject'], DTI_mods[i], ROIs[j]))
            #oddstore.update({'{}'.format(df_scaled.loc[n
                ,'Subject']) : '{} {}'.format(DTI_mods[i],
                ROIs[j])})
        else:
            cool = cool + 1
    _ = plt.hist(df_scaled[DTI_mods[i]],20)
    plt.title('Histogram of M{} {} {} z-scores'.format(
        ages[0], DTI_mods[i],ROIs[j]))
```

```
        plt.savefig('/home/macphena/Outputs/{}_{}/hist_{}_{}_
            {}.png'.format(sex[0],ages[0], ages[0], DTI_mods[i
            ], ROIs[j]))
        plt.show()



#%% Create dataframe to store the number of outlier values
    per subject ##########
df_odd = pd.DataFrame(0, index=df_scaled.index, columns=
    df_scaled.columns)
df_odd['Subject'] = df_DTI['Subject']
for m in range(0,len(oddlist)):
    temp = oddlist[m].split()
    num = df_odd.index[df_odd['Subject'] == temp[0]].tolist()
    df_odd.loc[num, temp[1]] = df_odd.loc[num, temp[1]] + 1

#%% Plot the number of oddities per subject
   ######################################
index_bar = df_odd['Subject']

#df_bar = pd.DataFrame({'FA' : df_odd['FA'], 'AD' : df_odd['AD
    '], 'RD' : df_odd['RD'], 'MD' : df_odd['MD']}, index=
    df_odd.index, columns=df_temp.columns)
```

```
df_bar = pd.DataFrame(df_odd, index=df_odd.index, columns=
    df_odd.columns)
df_bar['Subject'] = df_bar['Subject'].apply(str)
df_bar = df_bar.set_index('Subject')
#ax = df_bar.plot.bar(stacked=True, xlabel='Subject ID',
    ylabel='# of occurances', rot=60, fontsize=8, title='
    Number of Outlier ROIs per Subject', figsize=(16,11))
df_bar.plot.bar(stacked=True, figsize=(16,11))
plt.title('Number of Outlier ROIs per Subject M{}, Threshold
    of 2.807 SD (99.5%)'.format(ages[0]))
plt.xlabel('Subject ID')
plt.ylabel('# of Occurances')
plt.xticks(rotation=70, fontsize=8)
plt.legend(title='DTI metrics', bbox_to_anchor=(1.05,1), loc=
    'upper left')

#%% PCA
    ##########################################################################

from sklearn.decomposition import PCA
# features already scaled from z-score
df_scree = pd.DataFrame(0, index=ROIs, columns=['PC1', 'PC2',
     'PC3', 'PC4'])
```

```
for p in range(0,len(ROIs)):
    # read in CSV file of ROI means and remove unwanted first
      column
    #df_DTI = pd.read_csv('Female_{}_{}.csv'.format(ages[0],
      ROIs[p]))
    df_DTI = pd.read_csv('Male_{}_{}.csv'.format(ages[0],ROIs
      [p]))
    df_DTI = df_DTI.rename(columns={'Unnamed: 0': 'Subject'})
    #df_DTIF = pd.read_csv('Female_{}_{}.csv'.format(ages[3],
      ROIs[p]))
    #df_DTIF = df_DTIF.rename(columns={'Unnamed: 0': 'Subject
      '})
    #df_DTI = df_DTI.append(df_DTIF)
    df_DTI = df_DTI.set_index([pd.Index(range(0,len(df_DTI)))
      ])

    for lmao in range(0,len(df_DTI['Subject'])):
        df_DTI.loc[lmao, 'Subject'] = ('F{}'.format(lmao))


    # Z-Scores
    ##########################################################################
```

```
# create 'to be scaled' dataframe without subject IDs
df_tbs = df_DTI.drop('Subject', axis=1)
# scale features of dataframe
scaled_features = StandardScaler().fit_transform(df_tbs.
    values)

# Add index and column names as well as subject IDs back
    into dataframe
df_scaled = pd.DataFrame(scaled_features, index=df_tbs.
    index, columns=df_tbs.columns)
df_scaled['Subject'] = df_DTI['Subject']
df_scaled = df_scaled[['Subject', 'FA', 'AD', 'RD', 'MD'
    ]]

x = df_scaled.loc[:, DTI_mods].values
y = df_scaled.loc[:, 'Subject'].values

pca = PCA()
pca_DTI = pca.fit_transform(x)

df_pca = pd.DataFrame(data=pca_DTI, columns=['PC1', 'PC2'
    , 'PC3', 'PC4'], index=df_bar.index)

# explained variance
```

```
        explained_variance = pca.explained_variance_ratio_
        print(explained_variance)
        for ll in range(0,len(df_scree.columns)):
            df_scree.loc[ROIs[p], df_scree.columns[ll]] =
                explained_variance[ll]




        df_DTI.to_csv('/home/macphena/Outputs/{}_{}/
            explained_variance_{}_{}_{}.csv'.format(sex[0],ages
            [0], ages[0], DTI_mods[i], ROIs[j]))
        # PC1 = 65% PC2 = 32% PC3 = 3% PC4 = 0%, reduce to 2
            components for visualization
        #ax2 = df_pca.plot.scatter(x='PC1', y='PC2', c='red',
            colormap='viridis')


# scatter plot of PC1 and PC2
#ax2 = df_pca.plot.scatter(x='PC1', y='PC2', c='red',
    colormap='viridis')


#%% Scree plot
PC_values = np.arange(pca.n_components_) + 1
# set up colour iteration
colour = iter(cm.rainbow(np.linspace(0, 1, len(ROIs))))
```

```
plt.figure(figsize=(11.5,8))
for jj in range(0,len(df_scree.index)):
    c = next(colour)
    plt.plot(PC_values, df_scree.loc[ROIs[jj],:], 'ro-',
        linewidth=2, label='Line {}'.format(jj), c=c)#'{}'.
        format(jj*0.05+0.05))
plt.title('Scree Plot of ROIs')
plt.xlabel('Principal Component')
plt.ylabel('Proportion of Explained Variance')
plt.legend(ROIs, loc='center left', bbox_to_anchor=(1, 0.5))
plt.ylim([0,1])
plt.show()

plt.savefig('/home/macphena/Outputs/{}_{}/scree_{}.png'.
    format(sex[0],ages[0], ages[0]))



#
    ================================================================


# for v in range(0,len(df_pca.index)):
#     fig = plt.figure(figsize=(8,6))
#     with plt.style.context(('ggplot')):
```

```
#             plt.scatter(df_pca.loc[df_pca.index[v], 'PC1'],
    df_pca.loc[df_pca.index[v], 'PC2'], edgecolors='k', cmap='
    jet')
#             plt.xlabel('PC1')
#             plt.ylabel('PC2')
#             plt.title('Score Plot')
#       plt.show()
#
```

---

```
#%% Euclidean Distance
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.covariance import MinCovDet, EmpiricalCovariance



#### Euclidean distance = sqrt(sum for i to N(v1[i] - v2[i])
    ^2)
##

for zz in range(0,len(ROIs)):
    # read in CSV file of ROI means and remove unwanted first
        column
```

```
df_DTI = pd.read_csv('Male_{}_{}.csv'.format(ages[0],ROIs
    [zz]))
#df_DTI = pd.read_csv('Male_{}_outlier_{}.csv'.format(
    ages[0],ROIs[zz]))
df_DTI = df_DTI.rename(columns={'Unnamed: 0': 'Subject'})
df_DTIF = pd.read_csv('Female_{}_{}.csv'.format(ages[3],
    ROIs[zz]))
df_DTIF = df_DTIF.rename(columns={'Unnamed: 0': 'Subject'
    })
df_DTI = df_DTI.append(df_DTIF)
df_DTI = df_DTI.set_index([pd.Index(range(0,145))])
for lmao in range(0,len(df_DTI['Subject'])):
    df_DTI.loc[lmao, 'Subject'] = ('M{}'.format(lmao))
# Z-Scores
    ##########################################################################



# create 'to be scaled' dataframe without subject IDs
df_tbs = df_DTI.drop('Subject', axis=1)
# scale features of dataframe
scaled_features = StandardScaler().fit_transform(df_tbs.
    values)
```

```
    # Add index and column names as well as subject IDs back
        into dataframe
    df_scaled = pd.DataFrame(scaled_features, index=df_tbs.
        index, columns=df_tbs.columns)
    df_scaled['Subject'] = df_DTI['Subject']
    df_scaled = df_scaled[['Subject', 'FA', 'AD', 'RD', 'MD'
        ]]
    x = df_scaled.loc[:, DTI_mods].values
    y = df_scaled.loc[:, 'Subject'].values
    pca = PCA(n_components=(3))
    pca_DTI = pca.fit_transform(x)
    df_pca = pd.DataFrame(data=pca_DTI, columns=['PC1', 'PC2'
        , 'PC3'], index=df_bar.index)
    euc = euclidean_distances(df_pca.loc[:,:], df_pca.loc
        [:,:])

    fig,ax5 = plt.subplots(figsize=(10,5))
    data_plot = ax5.scatter(df_pca['PC1'], df_pca['PC2'],
        color='black')

#
    ================================================================

#       for ll in range(0,len(df_scree.columns)):
```

```
#            df_scree.loc[ROIs[zz], df_scree.columns[ll]] =
    explained_variance[ll]
#
#
    ================================================================


#%% Minimum Covaraince Determinant estimation and Mahalanobis
    distance ############
from sklearn.covariance import MinCovDet, EmpiricalCovariance
    , EllipticEnvelope
from sklearn.svm import OneClassSVM
from sklearn.linear_model import SGDOneClassSVM
from scipy.stats import f
from sklearn.model_selection import train_test_split,
    GridSearchCV
import random

for p in range(0,len(ROIs)):
    # read in CSV file of ROI means and remove unwanted first
        column
    #df_DTI = pd.read_csv('Female_{}_{}.csv'.format(ages[0],
        ROIs[p]))
```

```
    df_DTI = pd.read_csv('Male_{}_{}.csv'.format(ages[0],ROIs
        [p]))
    df_DTI = df_DTI.rename(columns={'Unnamed: 0': 'Subject'})

#
    ================================================================

#       for aaa in range(0,len(ages)):
#           df_DTIF = pd.read_csv('Female_{}_{}.csv'.format(
    ages[aaa],ROIs[p]))
#           df_DTIF = df_DTIF.rename(columns={'Unnamed: 0': '
    Subject'})
#           df_DTI = df_DTI.append(df_DTIF)
#
#       for bbb in range(1,len(ages)):
#           df_DTIF = pd.read_csv('Male_{}_{}.csv'.format(ages[
    bbb],ROIs[p]))
#           df_DTIF = df_DTIF.rename(columns={'Unnamed: 0': '
    Subject'})
#           df_DTI = df_DTI.append(df_DTIF)
#
#
    ================================================================
```

```
#df_DTIG = pd.read_csv('Female_{}_{}.csv'.format(ages[2],
    ROIs[p]))
#df_DTIG = df_DTIG.rename(columns={'Unnamed: 0': 'Subject
    '})
#df_DTI = df_DTI.append(df_DTIG)

df_DTI = df_DTI.set_index([pd.Index(range(0,len(df_DTI)))
    ])
for lmao in range(0,len(df_DTI['Subject'])):
    df_DTI.loc[lmao, 'Subject'] = ('M{}'.format(lmao))


# Z-Scores
    ##########################################################################


# create 'to be scaled' dataframe without subject IDs
df_tbs = df_DTI.drop('Subject', axis=1)
# scale features of dataframe
scaled_features = StandardScaler().fit_transform(df_tbs.
    values)

# Add index and column names as well as subject IDs back
    into dataframe
```

```
df_scaled = pd.DataFrame(scaled_features, index=df_tbs.
    index, columns=df_tbs.columns)
df_scaled['Subject'] = df_DTI['Subject']
df_scaled = df_scaled[['Subject', 'FA', 'AD', 'RD', 'MD'
    ]]

x = df_scaled.loc[:, DTI_mods].values
y = df_scaled.loc[:, 'Subject'].values

pca = PCA()
pca_DTI = pca.fit_transform(x)

#df_pca_m = pd.DataFrame(data=pca_DTI[:,0:3], columns=['
    PC1', 'PC2', 'PC3'], index=df_bar.index)
df_pca = pd.DataFrame(data=pca_DTI[:,0:2], columns=['PC1'
    , 'PC2'], index=df_DTI['Subject'])

# fit a MCD estimator to both the PCA and raw data
mcd_cov_pca = MinCovDet(random_state=0).fit(df_pca.values
    )
#mcd_cov_dti = MinCovDet(random_state=0).fit(x)
# fit a MLE estimator to both the PCA and raw data
mle_cov_pca = EmpiricalCovariance().fit(df_pca.values)
#mle_cov_dti = EmpiricalCovariance().fit(x)
```

```
    ### square root the MLE dist
#
    ================================================================================


#       mle_sqrt_pca = list()
#       for qwer in range(0,len(df_pca)):
#           hold_sqrt = math.sqrt(mle_cov_pca.mahalanobis(
    df_pca)[qwer])
#           mle_sqrt_pca.append(hold_sqrt)
#
    ================================================================================



    print('The Estimated Covariance matrices for the PCA
        values of M{} {}:\nMLE:\n{} \nMCD:\n{}'.format(ages
        [0],ROIs[p],mle_cov_pca.covariance_,mcd_cov_pca.
        covariance_))
```

```
#print('The Estimated Covariance matrices for the mean
    DTI values of M{} {}:\nMLE:\n{} \nMCD:\n{}'.format(
    ages[0],ROIs[p],mle_cov_dti.covariance_,mcd_cov_dti.
    covariance_))

# Calculate F-distribution Score
dfn, dfd = (len(df_pca.columns) - 1), (len(df_pca) - len(
    df_pca.columns))
f_ppf = f.ppf(0.995, dfn, dfd)
f_line = [f_ppf] * len(df_DTI)
#f_line = [f_ppf] * 146

### Plot MLE Mahalanobis distance vs index

fig,ax6 = plt.subplots(figsize=(10,6))
data_plot = ax6.scatter(df_pca.index, mle_cov_pca.
    mahalanobis(df_pca.values), color='black', )
plt.plot(df_pca.index, f_line, color='red')
plt.title('Mahalanobis Distance per Subject M{}'.format(
    ages[0]))
plt.xlabel('Subject ID')
plt.ylabel('MLE Mahalanobis Distance')
plt.xticks(rotation=70, fontsize=8)
```

```
#plt.legend(title='DTI metrics', bbox_to_anchor=(1.05,1),
    loc='upper left')

#plt.savefig('/home/macphena/Outputs/{}_{}/MLE_M{}_{}.png
    '.format(sex[0],ages[0],ages[0], ROIs[p]))

### Plot MCD Mahalanobis distance vs index

fig,ax6 = plt.subplots(figsize=(10,6))
data_plot = ax6.scatter(df_pca.index, mcd_cov_pca.dist_,
   color='black', )
plt.plot(df_pca.index, f_line, color='red')
plt.title('Mahalanobis Distance per Subject M{}'.format(
   ages[0]))
plt.xlabel('Subject ID')
plt.ylabel('MCD Mahalanobis Distance')
plt.xticks(rotation=70, fontsize=8)
#plt.legend(title='DTI metrics', bbox_to_anchor=(1.05,1),
    loc='upper left')

#plt.savefig('/home/macphena/Outputs/{}_{}/MCD_M{}_{}.png
    '.format(sex[0],ages[0],ages[0], ROIs[p]))
```

```
### Plot MCD Mahalanobis distance vs MLE Mahal D.


fig,ax6 = plt.subplots(figsize=(10,6))
data_plot = ax6.scatter(mle_cov_pca.mahalanobis(df_pca.
    values), mcd_cov_pca.dist_, color='black', )
#data_plot = ax6.scatter(mcd_cov_pca.reweight_covariance(
    df_pca), mcd_cov_pca.dist_, color='black', )
plt.plot(mcd_cov_pca.dist_, f_line, color='red')
plt.plot(f_line, mcd_cov_pca.dist_, color='red')
plt.title('Mahalanobis Distance per Subject M{}'.format(
    ages[0]))
plt.xlabel('Subject ID')
plt.ylabel('MCD Mahalanobis Distance')
plt.xticks(rotation=70, fontsize=8)
#plt.legend(title='DTI metrics', bbox_to_anchor=(1.05,1),
     loc='upper left')

#plt.savefig('/home/macphena/Outputs/{}_{}/MLE_vs_MCD_M{}
    _{}.png'.format(sex[0],ages[0],ages[0], ROIs[p]))



outie = 0
```

```
df_inliers_mcd = pd.DataFrame(columns=['PC1','PC2'])
df_outliers_mcd = pd.DataFrame(columns=['PC1','PC2'])
df_inliers_mle = pd.DataFrame(columns=['PC1','PC2'])
df_outliers_mle = pd.DataFrame(columns=['PC1','PC2'])



for qq in range(0,len(df_pca)):
    if mcd_cov_pca.dist_[qq] > f_ppf:
        outie = outie +1
        df_outliers_mcd.loc['M{}'.format(qq)] = df_pca.
            loc['M{}'.format(qq)]
    else:
        df_inliers_mcd.loc['M{}'.format(qq)] = df_pca.loc
            ['M{}'.format(qq)]

for qq in range(0,len(df_pca)):
    if mle_cov_pca.mahalanobis(df_pca.values)[qq] > f_ppf
        :
        outie = outie +1
        df_outliers_mle.loc['M{}'.format(qq)] = df_pca.
            loc['M{}'.format(qq)]
    else:
```

```
            df_inliers_mle.loc['M{}'.format(qq)] = df_pca.loc
                ['M{}'.format(qq)]

#print('Number of Mahalanobis distances over 99.5%
    confidence level of {}: {}'.format(f_ppf, outie))
print('Percentage of outliers for M{} {}\nMCD:{}
    MLE:{}'.format(ages[0],ROIs[p],(len(df_outliers_mcd)/
    len(df_pca)),(len(df_outliers_mle)/len(df_pca))))

fig,ax3 = plt.subplots(figsize=(10,5))
#data_plot = ax3.scatter(df_pca['PC1'], df_pca['PC2'],
    color='black')
inlier_plot = ax3.scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black')
inlier_plot.set_label('Inliers')
outlier_plot = ax3.scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red')
outlier_plot.set_label('Outliers')
ax3.set_title('Mahalanobis distance of M{} {} dataset'.
    format(ages[0],ROIs[p]))
ax3.set_xlim(-6,6)
ax3.set_ylim(-6,6)
plt.xlabel('PC1')
plt.ylabel('PC2')
```

```
xx, yy = np.meshgrid(
    np.linspace(plt.xlim()[0], plt.xlim()[1], 100),
    np.linspace(plt.ylim()[0], plt.ylim()[1], 100),
)

zz = np.c_[xx.ravel(), yy.ravel()]

mahal_mle_cov = mle_cov_pca.mahalanobis(zz)
mahal_mle_cov = mahal_mle_cov.reshape(xx.shape)

# plot MLE contour in BLUE
emp_cov_contour = plt.contour(xx, yy, np.sqrt(
   mahal_mle_cov), levels=[1,2,3,4,5,6], cmap='Blues_r',
   linestyles='dashed')
#emp_cov_contour.set_label('MLE')


mahal_mcd_cov = mcd_cov_pca.mahalanobis(zz)
mahal_mcd_cov = mahal_mcd_cov.reshape(xx.shape)

# plot MCD contour in RED
```

```
mcd_cov_contour = plt.contour(xx, yy, np.sqrt(
    mahal_mcd_cov), levels=[1,2,3,4,5,6], cmap='Reds_r',
    linestyles='dashed')
#mcd_cov_contour.set_label('MCD')

ax3.legend(
    [
    emp_cov_contour.collections[1],
    mcd_cov_contour.collections[1],
    inlier_plot,
    outlier_plot,
    ],
    ["MLE dist", "MCD dist", "inliers", "outliers"],loc='
        upper right')

#plt.savefig('/home/macphena/Outputs/{}_{}/contour_M{}_
    {}.png'.format(sex[0],ages[0],ages[0], ROIs[p]))


### OCSVM
    ##################################################################
```

```
df_train_mcd, df_test_mcd = train_test_split(
    df_inliers_mcd, train_size=0.75)

OCSVMa = OneClassSVM(nu=0.1, gamma=0.05)
OCSVMa.fit(df_pca.values)
OCSVMb = OneClassSVM(nu=0.1, gamma=0.01)
OCSVMb.fit(df_pca.values)
OCSVMc = OneClassSVM(nu=0.1, gamma=0.1)
OCSVMc.fit(df_pca.values)

#y_pred_train = OCSVM.predict(df_train_mcd)
#y_pred_test = OCSVM.predict(df_test_mcd)
#y_pred_outliers = OCSVM.predict(df_outliers_mcd)

SVM_dec = OCSVMa.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_dec = SVM_dec.reshape(xx.shape)
SVM_decb = OCSVMb.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_decb = SVM_decb.reshape(xx.shape)
SVM_decc = OCSVMc.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_decc = SVM_decc.reshape(xx.shape)
```

```
fig,ax7 = plt.subplots(figsize=(11,7))
data_plot = ax7.scatter(df_pca['PC1'], df_pca['PC2'],
    color='black')
data_plot.set_label('Inlier')
#train_plot = ax7.scatter(df_train_mcd['PC1'],
    df_train_mcd['PC2'], color='black')
#train_plot.set_label('Trained')
#test_plot = ax7.scatter(df_test_mcd['PC1'], df_test_mcd
    ['PC2'], color='purple')
#test_plot.set_label('Test')
outlier_plot = ax7.scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'], color='red')
outlier_plot.set_label('Outlier')
ax7.set_title('One-Class SVM Outlier detection of M{} {}
    dataset'.format(ages[0],ROIs[p]))
ax7.set_xlim(-5,5)
ax7.set_ylim(-5,5)
plt.xlabel('PC1')
plt.ylabel('PC2')

# plot SVM contour in GREEN
SVM_contour = plt.contour(xx, yy, SVM_dec, levels=0,
    colors='darkred', linewidths=3)
```

```
SVM_contourb = plt.contour(xx, yy, SVM_decb, levels=0,
    colors='darkblue', linewidths=3)
SVM_contourc = plt.contour(xx, yy, SVM_decc, levels=0,
    colors='darkgreen', linewidths=3)

ax7.legend(
    [
    SVM_contour.collections[1],
    SVM_contourb.collections[1],
    SVM_contourc.collections[1],
    inlier_plot,
    outlier_plot,
    ],
    ["OCSVM (nu=0.1, g=0.05)", "OCSVM (nu=0.1, g=0.01)",
        "OCSVM (nu=0.1 g=0.1)", "inliers", "outliers"],loc
        ='upper right')#, bbox_to_anchor=(1, 0.5))

plt.savefig('/home/macphena/Outputs/{}_{}/OCSVM_gamma_M{}
    _{}.png'.format(sex[0],ages[0],ages[0], ROIs[p]))


#plt.savefig('/home/macphena/brainDATA/Outputs/Contour/
    contour_M{}_{}.png'.format(ages[0], ROIs[p]))
```

```
    #plt.savefig('/Users/neilmacphee/Dropbox/Grad_School/
        Grad_School/Outputs/contour_F{}_{}.png'.format(ages
        [0], ROIs[p]))
#
    ================================================================

#       # Calculate the MLE based Mahalanobis distances
#       mahal_mle_cov = mle_cov_pca.mahalanobis()
#
#       # Create meshgrid of feature 1 and feature 2 values
#       xx, yy = np.meshgrid(
#           np.linspace(plt.xlim()[0], plt.xlim()[1], 100),
#           np.linspace(plt.ylim()[0], plt.ylim()[1], 100),
#       )
#
    ================================================================


#%% Support Vector Machine
    ##########################################################
    from sklearn.kernel_approximation import Nystroem
    from sklearn.pipeline import make_pipeline
    from sklearn.model_selection import GridSearchCV, KFold
```

```
from sklearn.metrics import f1_score, make_scorer,
    classification_report, confusion_matrix
import warnings
warnings.filterwarnings('once')

#f1sc = make_scorer(f1score)
nus = [0.0001, 0.001, 0.01, 0.1, 1]
gammas = [0.0001, 0.001, 0.01, 0.1, 1]
#nus = np.arange(0.01,1,0.01).tolist()
#gammas = np.arange(0.01,1,0.01).tolist()
tuned_parameters = {'kernel':['rbf'], 'gamma':gammas, 'nu
    ':nus}
scores = ['precision', 'recall']

#splits = KFold(n_splits=5).split(df_inliers_mcd)
# split inlier data into training and test sets
# constructs a generator of tupils of train, test where
    they contain the indices of training and test data

Xsvm = np.concatenate([df_inliers_mcd, df_outliers_mcd],
    axis=0)
# combine into one dataset to pass to GridSearchCV

ysvm_ = np.repeat(1.0, len(df_inliers_mcd.values))
```

```
ysvm_out = np.repeat(-1.0, len(df_outliers_mcd.values))


ysvm = np.concatenate([np.repeat(1.0, len(df_inliers_mcd.
    values)), np.repeat(-1.0, len(df_outliers_mcd.values))
    ])
# OCSVM uses +1 for inlier values and -1 for outlier
    values

nn, mm = len(df_inliers_mcd), len(df_outliers_mcd)

#splits = ((train, np.concatenate([test, np.arange(nn, nn
    +mm)], axis=0)) for train, test in splits)
# make a new generator of (train, test) splits with
    indices for the anomalous examples included in the
    test folds


X_train, X_test, y_train, y_test = train_test_split(
    df_inliers_mcd, ysvm_, train_size=0.75, random_state
    =(7))

y_test_out = np.concatenate((y_test, ysvm_out))
```

```
X_test_out = np.concatenate((X_test, df_outliers_mcd.
    values))

grid_search = GridSearchCV(OneClassSVM(),
    tuned_parameters, scoring='f1', refit=True, verbose=3)
grid_search.fit(X_train,y_train)



OCSVM = OneClassSVM(nu=0.15, gamma=0.1)
OCSVM.fit(df_pca.values)

transform = Nystroem(gamma=0.12, random_state=777)
OCSVM_sgd = SGDOneClassSVM(nu=0.05, random_state=777, tol
    =1e-6, fit_intercept=True)
pipe_sgd = make_pipeline(transform, OCSVM_sgd)
pipe_sgd.fit(X_train.values)

OCSVM_nov = OneClassSVM(nu=0.01, gamma=0.01)
OCSVM_nov.fit(X_train.values)

y_pred_train = OCSVM_nov.predict(df_train_mcd.values)
y_pred_test = OCSVM_nov.predict(df_test_mcd.values)
```

```
y_pred_outliers = OCSVM_nov.predict(df_outliers_mcd.
    values)

grid_search.best_params_



SVM_dec = OCSVM.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_dec = SVM_dec.reshape(xx.shape)

SVM_dec_nov = OCSVM_nov.decision_function(np.c_[xx.ravel
    (), yy.ravel()])
SVM_dec_nov = SVM_dec_nov.reshape(xx.shape)

SVM_dec_sig = pipe_sgd.decision_function(np.c_[xx.ravel()
    , yy.ravel()])
SVM_dec_sig = SVM_dec_sig.reshape(xx.shape)

fig,ax8 = plt.subplots(figsize=(10,5))
#data_plot = ax3.scatter(df_pca['PC1'], df_pca['PC2'],
    color='black')
train_plot = ax8.scatter(X_train['PC1'], X_train['PC2'],
    color='black', edgecolors='k')
train_plot.set_label('Trained')
```

```
test_plot = ax8.scatter(X_test['PC1'], X_test['PC2'],
    color='green', edgecolors='k')
test_plot.set_label('Test')
outlier_plot = ax8.scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
ax8.set_title('One-Class SVM vs SGD-OCSVM Novelty
    detection of M{} {} dataset'.format(ages[0],ROIs[p]))
ax8.set_xlim(-6,6)
ax8.set_ylim(-6,6)

# plot SVM contour in GREEN
#SVM_contour = plt.contour(xx, yy, SVM_dec, levels=4,
    colors='darkred')
SVM_contour_nov = plt.contour(xx, yy, SVM_dec_nov, levels
    =0, colors='darkred', linewidths=[3])
SVM_contour_sig = plt.contour(xx, yy, SVM_dec_sig, levels
    =0, colors='darkblue', linewidths=[3])

ax8.legend(
     [
     SVM_contour_nov.collections[1],
     SVM_contour_sig.collections[1],
     train_plot,
```

```
    test_plot,
    outlier_plot
    ],
    ["OCSVM", "SGD-OCSVM", "MCD-based training set", "MCD
        -based testing set", "MCD-based outliers"],loc='
        lower left', bbox_to_anchor=(1, 0.5))



OCSVM_poly = OneClassSVM(kernel='poly', nu=0.25, gamma
    =0.1)
OCSVM_poly.fit(df_pca.values)

OCSVM_sigmoid = OneClassSVM(kernel='sigmoid', nu=0.1,
    gamma=0.1)
OCSVM_sigmoid.fit(df_pca.values)

OCSVM_linear = OneClassSVM(kernel='linear', nu=0.1, gamma
    =0.1)
OCSVM_linear.fit(df_pca.values)

SVM_dec_ex1 = OCSVM_poly.decision_function(np.c_[xx.ravel
    (), yy.ravel()])
SVM_dec_ex1 = SVM_dec_ex1.reshape(xx.shape)
```

```
SVM_dec_ex2 = OCSVM_sigmoid.decision_function(np.c_[xx.
    ravel(), yy.ravel()])
SVM_dec_ex2 = SVM_dec_ex2.reshape(xx.shape)

SVM_dec_ex3 = OCSVM_linear.decision_function(np.c_[xx.
    ravel(), yy.ravel()])
SVM_dec_ex3 = SVM_dec_ex3.reshape(xx.shape)



fig,ax9 = plt.subplots(figsize=(10,5))
#data_plot = ax3.scatter(df_pca['PC1'], df_pca['PC2'],
    color='black')
train_plot = ax9.scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = ax9.scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
ax9.set_title('Poly and Sigmoid OCSVM outlier detection
    of M{} {} dataset'.format(ages[0],ROIs[p]))
ax9.set_xlim(-6,6)
ax9.set_ylim(-6,6)
plt.xlabel('PC1')
```

```
plt.ylabel('PC2')

# plot SVM contour in GREEN
#SVM_contour = plt.contour(xx, yy, SVM_dec, levels=4,
    colors='darkred')
SVM_contour_ex1 = plt.contour(xx, yy, SVM_dec_ex1, levels
   =0, colors='darkgreen', linewidths=[3])
SVM_contour_ex2 = plt.contour(xx, yy, SVM_dec_ex2, levels
   =0, colors='darkblue', linewidths=[3])

ax9.legend(
    [
    SVM_contour_ex1.collections[1],
    SVM_contour_ex2.collections[1],
    train_plot,
    outlier_plot
    ],
    ["OCSVM - poly", "OCSVM - sigmoid", "MCD-based
       Inliers", "MCD-based outliers"],loc='lower left',
       bbox_to_anchor=(1, 0.5))


fig,axs = plt.subplots(2, 2, figsize=(10,10))
```

```
#data_plot = ax3.scatter(df_pca['PC1'], df_pca['PC2'],
    color='black')
train_plot = axs[0,0].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = axs[0,0].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
train_plot = axs[0,1].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = axs[0,1].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
train_plot = axs[1,0].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = axs[1,0].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
train_plot = axs[1,1].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
```

```
outlier_plot = axs[1,1].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
fig.suptitle('Comparison of OCSVM Kernel decision
    functions', fontsize=20)
axs[0,0].set_title('Poly OCSVM outlier detection')
axs[0,1].set_title('Sigmoid OCSVM outlier detection')
axs[1,0].set_title('Linear OCSVM outlier detection')
axs[1,1].set_title('RBF OCSVM outlier detection')
SVM_contour_ex1 = axs[0,0].contour(xx, yy, SVM_dec_ex1,
    levels=0, colors='darkgreen', linewidths=[3])
SVM_contour_ex2 = axs[0,1].contour(xx, yy, SVM_dec_ex2,
    levels=0, colors='darkblue', linewidths=[3])
SVM_contour_ex3 = axs[1,0].contour(xx, yy, SVM_dec_ex3,
    levels=0, colors='darkred', linewidths=[3])
SVM_contour_ex4 = axs[1,1].contour(xx, yy, SVM_dec,
    levels=0, colors='orange', linewidths=[3])


OCSVMa = OneClassSVM(nu=0.15, gamma=1)
OCSVMa.fit(df_pca.values)
OCSVMb = OneClassSVM(nu=0.15, gamma=0.1)
OCSVMb.fit(df_pca.values)
OCSVMc = OneClassSVM(nu=0.15, gamma=0.05)
```

```
OCSVMc.fit(df_pca.values)
OCSVMd = OneClassSVM(nu=0.15, gamma=0.01)
OCSVMd.fit(df_pca.values)

SVM_dec = OCSVMa.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_dec = SVM_dec.reshape(xx.shape)
SVM_decb = OCSVMb.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_decb = SVM_decb.reshape(xx.shape)
SVM_decc = OCSVMc.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_decc = SVM_decc.reshape(xx.shape)
SVM_decd = OCSVMd.decision_function(np.c_[xx.ravel(), yy.
    ravel()])
SVM_decd = SVM_decd.reshape(xx.shape)

fig,axs = plt.subplots(2, 2, figsize=(10,10))
#data_plot = ax3.scatter(df_pca['PC1'], df_pca['PC2'],
    color='black')
train_plot = axs[0,0].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
```

```
outlier_plot = axs[0,0].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
train_plot = axs[0,1].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = axs[0,1].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
train_plot = axs[1,0].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = axs[1,0].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
train_plot = axs[1,1].scatter(df_inliers_mcd['PC1'],
    df_inliers_mcd['PC2'],color='black', edgecolors='k')
train_plot.set_label('Inlier')
outlier_plot = axs[1,1].scatter(df_outliers_mcd['PC1'],
    df_outliers_mcd['PC2'],color='red', edgecolors='k')
outlier_plot.set_label('Outlier')
fig.suptitle('Comparison of RBF OCSVM gamma values',
    fontsize=20)
axs[0,0].set_title('OCSVM gamma=1')
```

```
axs[0,1].set_title('OCSVM gamma=0.1')
axs[1,0].set_title('OCSVM gamma=0.01')
axs[1,1].set_title('OCSVM gamma=0.001')
SVM_contour_ex5 = axs[0,0].contour(xx, yy, SVM_dec,
    levels=0, colors='darkgreen', linewidths=[3])
SVM_contour_ex6 = axs[0,1].contour(xx, yy, SVM_decb,
    levels=0, colors='darkblue', linewidths=[3])
SVM_contour_ex7 = axs[1,0].contour(xx, yy, SVM_decc,
    levels=0, colors='darkred', linewidths=[3])
SVM_contour_ex8 = axs[1,1].contour(xx, yy, SVM_decd,
    levels=0, colors='orange', linewidths=[3])
```

```
#%%
```

```
# Define classifiers
classifiers = {
    "Empirical Covariance": EllipticEnvelope(
        support_fraction=1.0,contamination=0.25),
    "Minimum Covariance Determinant": EllipticEnvelope(
        contamination=0.25),
    "OCSVM": OneClassSVM(nu=0.25, gamma=0.35)}
```

```
colors = ["m", "g", "b"]
legend1 = {}
legend2 = {}

fig,ax7 = plt.subplots(figsize=(10,6))
xx1, yy1 = np.meshgrid(
    np.linspace(-6, 6, 100),
    np.linspace(-6, 6, 100)
    )

for ii, (clf_name, clf) in enumerate(classifiers.items())
   :
    plt.figure(1)
    clf.fit(df_pca)
    z1 = clf.decision_function(np.c_[xx1.ravel(), yy1.
       ravel()])
    z1 = z1.reshape(xx1.shape)
    legend1[clf_name] = plt.contour(
        xx1, yy1, z1, levels=[0], linewidths=2, colors=
           colors[ii]
        )
```

```
#%% Reshaping the data 2D->3D
   ##########################################################
zscr_3d = np.zeros(hold_data.shape)
dim = 0

for g in range(0,len(z_scr)):
    if g == 0:
        for a in range(0,len(zscr_3d)):
            for b in range(0,len(zscr_3d[0])):
                zscr_3d[a,b,0] =+ z_scr[g,b]
    if (g+1)%182 == 0:
        dim =+ 1
        for a in range(0,len(zscr_3d)):
            for b in range(0,len(zscr_3d[0])):
                zscr_3d[a,b,dim] =+ z_scr[g,b]



for x in range(1,182):
    # Display the background
    plt.imshow(zscr_3d[..., x].T, origin='lower',
       interpolation='nearest', cmap='gray')
    # Mask background values of activation map
    masked_act = np.ma.masked_equal(act, 0.)
```

```
    plt.imshow(masked_act[..., x].T, origin='lower',
        interpolation='nearest', cmap='hot')
    # Cosmetics: disable axis
    plt.axis('off')
    plt.show()


# [85:87,12323:12325]

from scipy.spatial import distance
eu_dist = distance.cdist(data_2d[filelist[4]], data_2d[
    filelist[4]], 'euclidean')



### PCA
    ##########################################################################

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
pca_datastore = dict()
hold_pca = float()
```

```
for p in range(0,len(filelist)):
    hold_pca = PCA().fit_transform(sc.fit_transform(data_2d[
        filelist[p]]))
    pca_datastore.update({filelist[p]:hold_pca})

#explained_variance = pca.explained_variance_ratio_

# Score plot of the first 2 PC
for v in range(0,len(filelist)):
    fig = plt.figure(figsize=(8,6))
    with plt.style.context(('ggplot')):
        plt.scatter(pca_datastore[filelist[v]][:, 0],
            pca_datastore[filelist[v]][:, 1], edgecolors='k',
            cmap='jet')
        plt.xlabel('PC1')
        plt.ylabel('PC2')
        plt.title('Score Plot (%s)' % filelist[v])
    plt.show()


with plt.style.context(('ggplot')):
```

```
    plt.scatter(pca_datastore[filelist[1]][:, 0],
        pca_datastore[filelist[1]][:, 1], edgecolors='k', cmap
        ='jet')
    plt.xlabel('PC1')
    plt.ylabel('PC2')
    plt.title('Score Plot (%s)' % filelist[1])
plt.show()

### outlier vs original ###
with plt.style.context(('ggplot')):
    plt.scatter(pca_datastore[filelist[3]][:, 0],
        pca_datastore[filelist[3]][:, 1], edgecolors='k', cmap
        ='jet')
    plt.scatter(pca_datastore[filelist[5]][:, 0],
        pca_datastore[filelist[5]][:, 1], edgecolors='k', cmap
        ='jet', c='green')
    plt.xlabel('PC1')
    plt.ylabel('PC2')
    plt.title('Score Plot (%s)' % filelist[1])
plt.show()


# Euclidean distance for score plots
euclidean_data = dict()
```

```
for z in range(0,len(filelist)):
    for i in range(75):
        euclidean_data.update({filelist[z]:(pca_datastore[
            filelist[z]][:, i] - np.mean(pca_datastore[
            filelist[z]][:,:75]))**2/np.var(pca_datastore[
            filelist[z]][:, :75])})

for p in range(0,len(filelist)):
    colour = [plt.cm.jet(float(i)/max(euclidean_data[filelist
        [p]])) for i in euclidean_data[filelist[p]]]
    fig = plt.figure(figsize=(8,6))
    with plt.style.context(('ggplot')):
        plt.scatter(pca_datastore[filelist[p]][:, 0],
            pca_datastore[filelist[p]][:, 1], c=colour,
            edgecolors='k', s=60)
        plt.xlabel('PC1')
        plt.ylabel('PC2')
        plt.title('Score Plot (%s)' % filelist[p])
        plt.show()

### Mahalanobic Distance
    ##########################################################
from sklearn.covariance import EmpiricalCovariance, MinCovDet
robust_cov = dict()
```

```
mahal_datastore = dict()

# fit a Minimum Covariance Determinant (MCD) robust estimator
    to data
robust_cov = MinCovDet(support_fraction=1).fit(pca_datastore[
   filelist[0]][:,:5])
# Get the Mahalanobis distance
m = robust_cov.mahalanobis(pca_datastore[filelist[1]][:,:5])


for b in range(0,len(filelist)):
    # fit a Minimum Covariance Determinant (MCD) robust
       estimator to data
    robust_cov = MinCovDet(support_fraction=0.9).fit(
       pca_datastore[filelist[b]][:, :5])
    # Get the Mahalanobis distance
    mahal_datastore.update({filelist[b]:robust_cov.
       mahalanobis(pca_datastore[filelist[b]][:, :5])})

for p in range(0,len(filelist)):
    colour = [plt.cm.jet(float(i)/max(mahal_datastore[
       filelist[p]])) for i in mahal_datastore[filelist[p]]]
    fig = plt.figure(figsize=(8,6))
    with plt.style.context(('ggplot')):
```

```
        plt.scatter(pca_datastore[filelist[p]][:, 0],
            pca_datastore[filelist[p]][:, 1], c=colour,
            edgecolors='k', s=60)
        plt.xlabel('PC1')
        plt.ylabel('PC2')
        plt.title('Score Plot (%s)' % filelist[p])
        plt.show()

### Local outlier factor
    ##########################################################
from sklearn.neighbors import LocalOutlierFactor
lof_datastore = dict()

for q in range(0,len(filelist)):
    lof_test = LocalOutlierFactor(n_neighbors=4)
    lof_datastore.update({filelist[q]:lof_test.fit_predict(
        pca_datastore[filelist[q]])})

for d in range(0,len(filelist)):
    colour = [plt.cm.jet(float(i)/max(lof_datastore[filelist[
        d]])) for i in lof_datastore[filelist[d]]]
    fig = plt.figure(figsize=(8,6))
    with plt.style.context(('ggplot')):
```

```
        plt.scatter(pca_datastore[filelist[d]][:, 0],
            pca_datastore[filelist[d]][:, 1], c=colour,
            edgecolors='k', s=60)
        plt.xlabel('PC1')
        plt.ylabel('PC2')
        plt.title('Score Plot (%s)' % filelist[d])
        plt.show()

lof_test = LocalOutlierFactor(n_neighbors=3)
lof_test.fit_predict(pca_datastore[filelist[0]])

#
    ##########################################################################
```

# Bibliography

(2020). 2020 alzheimers disease facts and figures. *Alzheimers & Dementia*, **16**(3), 391–460.

Andersson, J. L., Graham, M. S., Zsoldos, E., and Sotiropoulos, S. N. (2016). Incorporating outlier detection and replacement into a non-parametric framework for movement and distortion correction of diffusion mr images. *NeuroImage*, **141**, 556–572.

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Filho, W. J., Lent, R., and Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *The Journal of Comparative Neurology*, **513**(5), 532–541.

Basser, P., Mattiello, J., and LeBihan, D. (1994). Mr diffusion tensor spectroscopy and imaging. *Biophysical Journal*, **66**(1), 259–267.

Behrens, T., Woolrich, M., Jenkinson, M., Johansen-Berg, H., Nunes, R., Clare, S., Matthews, P., Brady, J., and Smith, S. (2003). Characterization and propagation of uncertainty in diffusion-weighted mr imaging. *Magnetic Resonance in Medicine*, **50**(5), 1077–1088.

Behrens, T., Berg, H. J., Jbabdi, S., Rushworth, M., and Woolrich, M. (2007). Probabilistic diffusion tractography with multiple fibre orientations: What can we gain? *NeuroImage*, **34**(1), 144–155.

Chapman, E. H., Weintraub, R. J., Milburn, M. A., Pirozzi, T. O., and Woo, E. (1999). Homeopathic treatment of mild traumatic brain injury: A randomized, double-blind, placebo-controlled clinical trial. *Journal of Head Trauma Rehabilitation*, **14**(6), 521–542.

Chiao, C.-C., Lin, C.-I., and Lee, M.-J. (2019). Multiple approaches for enhancing neural activity to promote neurite outgrowth of retinal explants. *Retinal Development Methods in Molecular Biology*, page 65–75.

Comper, P., Bisschop, S. M., Carnide, N., and Tricco, A. (2005). A systematic review of treatments for mild traumatic brain injury. *Brain Injury*, **19**(11), 863–880.

Dolz, J., Desrosiers, C., Wang, L., Yuan, J., Shen, D., and Ayed, I. B. (2020). Deep cnn ensembles and suggestive annotations for infant brain mri segmentation. *Computerized Medical Imaging and Graphics*, **79**, 101660.

Filippi, M., Cercignani, M., Inglese, M., Horsfield, M., and Comi, G. (2001). Diffusion tensor magnetic resonance imaging in multiple sclerosis. *Neurology*, **56**(3), 304–311.

Hanseeuw, B. J., Betensky, R. A., Jacobs, H. I. L., Schultz, A. P., Sepulcre, J., Becker, J. A., Cosio, D. M. O., Farrell, M., Quiroz, Y. T., Mormino, E. C., and et al. (2019). Association of amyloid and tau with cognition in preclinical alzheimer disease. *JAMA Neurology*, **76**(8), 915.

Hardin, J. and Rocke, D. M. (2005). The distribution of robust distances. *Journal of Computational and Graphical Statistics*, **14**(4), 928–946.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, **585**(7825), 357–362.

Holloway, R. L. (1968). The human brain in figures and tables: A quantitative handbook. by s. m. blinkov and i. i. glezer. basic books, new york. 482 pp. and 305 tables. 1968. $25.00. *American Journal of Physical Anthropology*, **29**(3), 449–450.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, **24**(6), 417–441.

Hubert, M., Debruyne, M., and Rousseeuw, P. J. (2017). Minimum covariance determinant and extensions. *WIREs Computational Statistics*, **10**(3).

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, **9**(3), 90–95.

Jack, C. R., Lowe, V. J., Weigand, S. D., Wiste, H. J., Senjem, M. L., Knopman, D. S., Shiung, M. M., Gunter, J. L., Boeve, B. F., Kemp, B. J., and et al. (2009). Serial pib and mri in normal, mild cognitive impairment and alzheimers disease: implications for sequence of pathological events in alzheimers disease. *Brain*, **132**(5), 1355–1365.

Jenkinson, M., Beckmann, C. F., Behrens, T. E., Woolrich, M. W., and Smith, S. M. (2012). Fsl. *NeuroImage*, **62**(2), 782–790.

Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., Wang, Y., Dong, Q., Shen, H., Wang, Y., and et al. (2017). Artificial intelligence in healthcare: Past, present and future. *Stroke and Vascular Neurology*, **2**(4), 230–243.

Kay, T., Harrington, D. E., and Adams, R. (1993). Definition of mild traumatic brain injury. *Journal of Head Trauma Rehabilitation*, **8**(3), 86–87.

Kingsley, P. B. (2006). Introduction to diffusion tensor imaging mathematics: Part i. tensors, rotations, and eigenvectors. *Concepts in Magnetic Resonance Part A*, **28A**(2), 101–122.

Kubicki, M., MCCARLEY, R., WESTIN, C., PARK, H., MAIER, S., KIKINIS, R., JOLESZ, F., and SHENTON, M. (2007). A review of diffusion tensor imaging studies in schizophrenia. *Journal of Psychiatric Research*, **41**(1-2), 15–30.

Li, X., Morgan, P. S., Ashburner, J., Smith, J., and Rorden, C. (2016). The first step for neuroimaging data analysis: Dicom to nifti conversion. *Journal of Neuroscience Methods*, **264**, 47–56.

Madden, D. J., Bennett, I. J., and Song, A. W. (2009). Cerebral white matter integrity and cognitive aging: Contributions from diffusion tensor imaging. *Neuropsychology Review*, **19**(4), 415–435.

Mazziotta, J. C., Toga, A. W., Evans, A., Fox, P., and Lancaster, J. (1995). A probabilistic atlas of the human brain: Theory and rationale for its development. *NeuroImage*, **2**(2), 89–101.

McCarthy, P. (2021). Fsleyes.

Mccrea, M., Iverson, G. L., Mcallister, T. W., Hammeke, T. A., Powell, M. R., Barr, W. B., and Kelly, J. P. (2009). An integrated review of recovery after mild traumatic brain injury (mtbi): Implications for clinical management. *The Clinical Neuropsychologist*, **23**(8), 1368–1390.

Mcdonald, B. C., Saykin, A. J., and Mcallister, T. W. (2012). Functional mri of mild traumatic brain injury (mtbi): progress and perspectives from the first decade of studies. *Brain Imaging and Behavior*, **6**(2), 193–207.

McKinney, W. *et al.* (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Mejia, A. F., Nebel, M. B., Eloyan, A., Caffo, B., and Lindquist, M. A. (2017). Pca leverage: Outlier detection for high-dimensional functional magnetic resonance imaging data. *Biostatistics*, **18**(3), 521–536.

Mergenthaler, P., Lindauer, U., Dienel, G. A., and Meisel, A. (2013). Sugar for the brain: the role of glucose in physiological and pathological brain function. *Trends in Neurosciences*, **36**(10), 587–597.

Moldovanu, S., (Vişan) Pungă, M., and Moraru, L. (2015). Evaluation of outliers in acquired brain mr images. *Journal of Physics: Conference Series*, **574**, 012121.

Mori, S., Wakana, S., Nagae-Poetscher, L. M., and Zijl, P. C. V. (2005). Three-dimensional atlas of brain white matter tracts. *MRI Atlas of Human White Matter*, page 15–31.

Mourão-Miranda, J., Hardoon, D. R., Hahn, T., Marquand, A. F., Williams, S. C., Shawe-Taylor, J., and Brammer, M. (2011). Patient classification as an outlier detection problem: An application of the one-class support vector machine. *NeuroImage*, **58**(3), 793–804.

Niethammer, M., Bouix, S., Aja-Fernández, S., Westin, C.-F., and Shenton, M. E. (2007). Outlier rejection for diffusion weighted imaging. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, page 161–168.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11), 559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.* (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, **12**(Oct), 2825–2830.

Petersen, R. C., Aisen, P. S., Beckett, L. A., Donohue, M. C., Gamst, A. C., Harvey, D. J., Jack, C. R., Jagust, W. J., Shaw, L. M., Toga, A. W., and et al. (2009). Alzheimers disease neuroimaging initiative (adni): Clinical characterization. *Neurology*, **74**(3), 201–209.

Raybaut, P. (2009). Spyder-documentation. *Available online at: pythonhosted. org.*

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, **79**(388), 871–880.

Sairanen, V., Leemans, A., and Tax, C. (2018). Fast and accurate slicewise outlier detection (solid) with informed model estimation for diffusion mri data. *NeuroImage*, **181**, 331–346.

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. (1999). Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA. MIT Press.

Smith, S. M. (2002). Fast robust automated brain extraction. *Human Brain Mapping*, **17**(3), 143–155.

Stejskal, E. O. and Tanner, J. E. (1965). Spin diffusion measurements: Spin echoes in the presence of a time-dependent field gradient. *The Journal of Chemical Physics*, **42**(1), 288–292.

Thornhill, S. (2000). Disability in young people and adults one year after head injury: Prospective cohort study. *BMJ*, **320**(7250), 1631–1635.

Torrey, H. C. (1956). Bloch equations with diffusion terms. *Physical Review*, **104**(3), 563–565.

Tortora, G. J. and Derrickson, B. (2014). *Principles of anatomy and physiology*. Wiley.

Travers, B. G., Adluru, N., Ennis, C., Tromp, D. P., Destiche, D., Doran, S., Bigler, E. D., Lange, N., Lainhart, J. E., Alexander, A. L., and et al. (2012). Diffusion tensor imaging in autism spectrum disorder: A review. *Autism Research*, **5**(5), 289–313.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, **17**, 261–272.

Vos, P. E., Battistin, L., Birbamer, G., Gerstenbrand, F., Potapov, A., Prevec, T., Stepan, C. A., Traubner, P., Twijnstra, A., Vecsei, L., and et al. (2002). Efns guideline on mild traumatic brain injury: report of an efns task force. *European Journal of Neurology*, **9**(3), 207–219.

Westlye, L. T., Walhovd, K. B., Dale, A. M., Bjornerud, A., Due-Tonnessen, P., Engvig, A., Grydeland, H., Tamnes, C. K., Ostby, Y., Fjell, A. M., and et al. (2009). Life-span changes of the human brain white matter: Diffusion tensor imaging (dti) and volumetry. *Cerebral Cortex*, **20**(9), 2055–2068.