MACHINE LEARNING FOR CLASSIFICATION OF PEDIATRIC CONCUSSION RECOVERY STAGES

MACHINE LEARNING FOR CLASSIFICATION OF PEDIATRIC CONCUSSION RECOVERY STAGES

BY

LAUREN ANDERSON, B.Eng.

A THESIS

SUBMITTED TO THE 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 (2021) McMaster University

(biomedical engineering) Hamilton, Ontario, Canada

TITLE:	Machine Learning for Classification of Pediatric Concussion Recovery Stages
AUTHOR:	Lauren Anderson B.Eng. (Biomedical Engineering), University of Guelph, Guelph, Canada
SUPERVISOR:	Michael D. Noseworthy
NUMBER OF PAGES:	xxi, 170

Lay Abstract

Concussions are recorded in approximately 300,000 athletes annually and are estimated to affect up to 3.8 million individuals per year in the United States alone. Understanding when its safe to return to normal routine after an injury is important but challenging. Therefore, a series of stages have been developed to lead children through a safe and timely return to sport and activity after concussion. The goal of this study was to develop machine learning (ML) algorithms which predict these return stages using symptom recordings and gross body movement data. Algorithms could be incorporated into a smartphone application (APP) to provide accessible return guidelines for children with concussions. Algorithms were created and model performance was tested using symptom and body movement data collected from children after a concussive injury. The results of this study show that it is possible to predict return to school and return to activity stages with ML, and with improvements, can be used to facilitate return from injury.

Abstract

Mild traumatic brain injury (mTBI), or concussion, results from sudden acceleration or deceleration of the brain and subsequent complex tissue propagation of shock waves that disrupt structure and function. Concussions can cause many symptoms including headache, dizziness, and difficulty concentrating. These can be detrimental to children, affecting their participation in school, sport, and social activities. Therefore, return to school (RTS) and return to activity (RTA) protocols have been developed to help safely return children to these activities without risking further injury. The goal of this study was to develop machine learning (ML) algorithms to predict RTA and RTS stages, that can easily be incorporated into a smartphone application (APP). Ideally this would assist children in tracking and determining their RTA and RTS progression leading them to a safe and timely return.

Support vector machine classifier (SVC) and random forest (RF) algorithms were developed to predict RTA/RTS stages. Both were modeled on previously acquired data, and on newly acquired data, and results were compared. Models were trained and tested using accelerometry and symptom data from pediatric concussion patients. A sliding window technique and feature extraction were performed on raw acceleration data to extract suitable features, which were combined with yes/no symptom recordings as ML inputs. The dataset consisted of 67 participants aged 10 to 18, 42

female and 25 male, with a total of 844408 samples.

The best results for RTS prediction showed average accuracy of 83% for RF and 66% for SVC. For RTA predictions, the best results had average accuracy of 60% for RF and 58% for SVC. For new data, RTS predictions showed an accuracy of 45% for RF and 41% for SVC. RTA predictions had an accuracy of 35% for RF and 30% for SVC. RF models had superior performance on all data. These results show that predicting RTA/RTS is possible with ML. However, improvements to these models can be made by training on more data prior to APP implementation. More data is needed, as recruitment during this study was limited due to Covid-19 restrictions.

To Leanne

For being my #1 fan.

Acknowledgements

Firstly, I would like to thank Dr. Noseworthy for all of the help and support throughout this project. I would not have been able to make it through without your guidance. I greatly appreciate you taking a chance on me and accepting me into your lab, I have learned so much from you and it has been a wonderful few years! I would also like to thank Dr. Doyle and Professor DeMatteo, who have helped in so many ways. None of this would have been possible without any of you!

I was lucky to have had such great help from members of CanChild, specifically Kathy and Josie. Both of you never failed to be available to answer any questions or concerns I had, and fully supported me over the past two years. I am very grateful for your help!

Finally, to all of my wonderful lab mates who have provided assistance in any way they could. To Calvin and Ama, who were always willing to help when I was stuck, and to Elyse, who patiently sat through far too many rants. I feel very lucky to have had such a strong support system over the years, so thank you all!

Contents

List of Figures

List of Tables

Abbreviations

Abbreviations

ADC Apparent Diffusion Coefficient

AI Artificial intelligence

ANS Autonomic Nervous System

APP Application

ATP Adenosine Triphosphate

AUC Area Under Curve

BESS Balance Error Scoring System

BOLD Blood Oxygen Level Dependant

CART Classification and Regression Trees

CPU Central Processing Unit

CRT Concussion Recognition Tool

CV	Cross Validation
DMN	Default Mode Network
DTI	Diffusion Tensor Imaging
ECG	Electrocardiogram
FA	Fractional Anisotropy
FFT	Fast Fourier Transform
fMRI	Functional Magnetic Resonance Imaging
GCS	Glasgow Coma Scale
GPU	Graphics Processing Unit
HAR	Human Activity Recognition
HF	High Frequency
HFnu	High Frequency Normalized Units
HRV	Heart Rate Variability
K-D	King-Devick
LF	Low Frequency
MAD	Median Aboslute Deviation
ML	Machine Learning
MLA	Machine Learning Algorithm

mPFC	Medial Prefrontal Cortex
MRI	Magnetic Resonance Imaging
mTBI	Mild Traumatic Brain Injury
PCC	Posterior Cingulate Cortex
PCS	Post Concussive Symptoms
PCSS	Post Concussion Symptom Scale
pNN50	Percent intervalls Varying by more than 50 milliseconds
PPCS	Persistant Post Concussion Symptoms
RBF	Radial Basis Function
RF	Random Forest
RMSSD	Root Mean Square of Successive R-R Differences
ROC	Receiver Operating Characteristic Curve
rs-fMRI	Resting State Functional Magentic Resonance Imaging
RTA	Return to Activity
RTS	Return to School
SCAT	Sport Concussion Assessment Tool
SD	Standard Deviation
SDNN	standard deviation of total variance

SMA	Signal Magnitude Area
SMART	Self-Monitoring Activity Restriction and Relaxation Training
SOT	Sensory Organization Test
SRC	Sport Related Concussion
SVC	Support Vecotr Machine Classifier
SVM	Signal Vector Magnitude
ULF	Ultra Low Frequency
VLF	Very Low Frequency

Chapter 1

Introduction

Mild traumatic brain injury (mTBI), formally known as concussion, is recorded in approximately 300 000 athletes annually, however, is estimated to affect up to 3.8 million individuals a year in the United States (Halstead *et al.*, 2010). According to the government of Canada, 46 000 children had recorded concussions in 2016-2017 (Canada, 2020), and only 1 in 4 people are aware of how concussions are treated. Concussions can cause many symptoms including headache, dizziness, and difficulty concentrating. These can be detrimental to children, affecting their participation in school, sport, and social activities. Finding the balance between returning too soon and staying out from activities for too long can be challenging, but is important to keep children from isolating from their normal routines for too long(DeMatteo *et al.*, 2019). Providing children with a simple solution to understanding the return from their injury could increase understanding of return protocols, and facilitate the recovery process.

The current research proposes the use of two machine learning algorithms (MLA), random forest and support vector machine classifiers, to predict the pre-determined

return to school (RTS) and return to activity (RTA) protocols for children with concussion(Dematteo *et al.*, 2015a)(Dematteo *et al.*, 2015b). Using self-recorded symptoms and recorded body movement as the basis for inputs to the algorithms, stages could be predicted and easily be portrayed to the child assisting them in moving through the stages leading to a safe and timely recovery. Alongside this study, a phone application (APP) called Back2Play has been developed by developers with the company Medic associated with Mohawk College, where children can record their symptoms, and movement is tracked using an Apple watch. The end goal for this research is to incorporate the ML model into the APP to predict and display which stage a child is in at any given time throughout the day. APP results could then be correlated to brain recovery found from functional medical imaging techniques. This study was part of the Back2Play study presented by research company CanChild.

This thesis describes in detail the methodology, results, and conclusions from finding the best ML model to incorporate into the APP for use of children with concussions. Two algorithms were used and compared to find the best model for predicting RTA and RTS stages. Chapter 2 highlights important background information on concussions, currently available tools and technologies for assessing and intervening with concussions, and introduces the machine learning models that are used in this study. Chapter 3 describes the methods for this research, including data processing and ML modelling. Chapters 4 through 7 describes the results for models created on different processing methods for the initial study data. Each method taken was a new variation from the former, with the hopes of improving algorithm performance with each test. Chapter 8 describes the results for the models created on a newly collected set of data. Chapter 9 discusses the results presented in previous chapters.

Finally, chapter 10 summarizes the findings of this study, while highlighting potential limitations, and describing potential future work prior to APP implementation.

Chapter 2

Background

This chapter provides a brief background into mild traumatic brain injuries, including assessment tools, protocols of intervention, and currently available products to assist children with concussions.

2.1 Mild Traumatic Brain Injury

A concussion occurs when there is sudden acceleration or deceleration of the brain inside the skull, which can be caused by traumatic forces to the body or head producing impulsive forces which are transferred to the brain. A concussive injury results in a rapid onset of impairment and neurological function which usually resolves quickly, however, these impairments can last from minutes to hours (McCrory *et al.*, 2017). The signs and symptoms of concussion largely reflect a functional disturbance as opposed to a structural injury (McCrory *et al.*, 2017)(Chen *et al.*, 2004) however, imaging techniques have found axonal injury in research settings. Shrey *et al.* suggest that more work must be done to apply these findings clinically(Shrey *et al.*,

2011). Following an injury to the brain, there is a release of neurotransmitters that result in an efflux of potassium and influx of calcium. These ionic fluxes then cause changes in cellular physiology (Giza and Hovda, 2014). In order to normalize these shifts, the Na+-K+ATPase pump must work harder, requiring an increase of adenosine triphosphate (ATP). Dramatic increases in the level of ATP causes an imbalance between glucose supply and demand thus causing a cellular energy crisis (Giza and Hovda, 2014). This crisis creates a vulnerability and in the brain's response to a second injury, which increases the possibility of longer-lasting deficits. Additionally, after injury, axonal stretching occurs which can cause membrane disruption and depolarization. This can lead to increased calcium influx, mitochondrial swelling, and altered neurofilament stability (Giza and Hovda, 2014). The injury can be considerably magnified and exacerbated calcium is often a second messenger in biochemical pathways. Based on these pathophysological effects, Shrey *et al.* suggests that youth could be more susceptible to second injury(Shrey *et al.*, 2011). Furthermore, individuals with previous history of concussion are reportedly up to 5.8 times more likely to sustain another injury(Zemper, 2003). Additional injury prior to healing from the initial can result in second-impact syndrome in youth under 18 years producing cerebral vascular congestion that can lead to severe morbidity, and death in severe cases(Halstead *et al.*, 2018).

Concussions cause several physical, cognitive and emotional symptoms including, but not limited to, sensitivity to light or noise, headaches, dizziness, fogginess, depression, and difficulty concentrating (Halstead *et al.*, 2010). Concussion greatly affects youth in sports, and there has been a 40% increase in incidence noted in emergency departments for sport related mTBIs in the past decade alone (DeMatteo *et al.*, 2019).

A study done by Zuckerman *et al.* found that the recovery time for children ages 13-18 is longer than youth aged 18-22 (Zuckerman *et al.*, 2012). Symptoms occurring due to concussion can significantly impact the participation of children in school, social, and physical activities. A major focus in concussion research is providing children with the understanding of when it is safe for them to return to these activities. Returning to activities too soon can risk more severe injuries or a longer recovery from the initial injury, and it is known that children have a higher risk of re-injury (Giza and Hovda, 2014)(Dematteo *et al.*, 2015c). Furthermore, isolating children from school and activities for longer than necessary can cause anxiety or higher incidence of depression due to isolation from normal routine (DeMatteo *et al.*, 2019). Therefore, return to school (RTS) and return to activity (RTA) protocols have been developed and can be used to identify when children can continue participation without risking further injury (DeMatteo *et al.*, 2019).

2.2 Concussion Assessment

The following section describes several tools that are currently used to assess concussions either on the sidelines, or in clinical settings.

2.2.1 PCSS

The post concussion symptom scale (PCSS) is a self-assessment used to rank symptoms based on their severity. The list consists of 22 symptoms, which can be seen in Table 2.1, and each symptom is ranked from 0-6 where 0 is no symptom, 3 is moderate, and 6 is severe.

Table 2.1: PCSS Symptoms (Hawaii Concussion, 2017)

Symptoms		
Headache	Nausea	Vomitting
Balance problems	Dizziness	Fatigue
Trouble falling asleep	Excessive sleep	Loss of sleep
Drowsiness	Light sensitivity	Noise sensitivity
Irritability	Sadness	Nervousness
More emotional	Numbness	Difficulty concentrating
Feeling 'foggy'	Feeling 'slow'	Difficulty remembering
	Visual problems	

PCSS is a good benchmark for individuals to self assess symptoms and can show important information on the severity of the concussion.

2.2.2 King Devick Test

Visual testing can be important to determining and assessing a brain injury such as concussion. The pathways connecting the eyes to the visual system are quite complex with many intersections in the frontal, parietal and temporal lobes of the brain (Galetta *et al.*, 2016). Many subcortical structures, such as the thalamus, are involved in eye movement causing an interconnection between eye movement and neural activity in these regions. Based on this relationship, saccade testing is a highly appropriate option for testing the neurophysiological effects of concussion (Galetta *et al.*, 2016). The King-Devick (K-D) test was developed to assess visual performance measures

such as rapid eye movement. The K-D test is a two-minute assessment where patients rapidly read numbers from a card or computer-based system, measuring eye movement, attention and language function (Galetta *et al.*, 2016). The results of this test have been shown to correlate with suboptimal brain function in concussion patients. The K-D test is mainly used in sport as a sideline tool for assessing concussion acquired during an activity.

2.2.3 Balance Testing

Balance and related problems are a major symptom of concussion(Guskiewicz, 2011). Measuring balance can assist in tracking patient recovery and can be used to aid in determining safe return to school and activity. Several methods exist to measure balance, including the Sensory Organization Test (SOT) and the Balance Error Scoring System (BESS).

The SOT method implements a specially designed force plate that alters the users somatosensory and visual inputs, in real time, to assess how the individual can maintain their stance (Guskiewicz, 2011). Expensive technology is required for this test and therefore it is not a feasible manner of assessing concussion recovery, especially if considering assessment on the sidelines. More commonly, BESS testing is used to assess balance. For BESS testing, a series of 3 stances are performed on flat ground followed by on a foam pad for a total of 6 trials. The stances are shown in Figure 2.1.

Figure 2.1: BESS testing stances.

2.2.4 Sport Concussion Assessment Tool

The Sport Concussion Assessment Tool (SCAT) was first developed by the Concussion in Sport Group at a meeting in Prague in 2004 and was developed as a tool for the public to assist in detecting sport related concussion (Echemendia *et al.*, 2017c). Since its introduction, the SCAT tool has continued to be refined, with the current definition being the SCAT5. SCAT5 was designed for use by healthcare practitioners on individuals 13 years or older who are believed to have had a sport related concussion (SRC). A child version of the SCAT5 was also designed for individuals younger than 13 (Echemendia *et al.*, 2017c). For the non-medically trained, a separate concussion recognition tool (CRT5) was developed to assess SRC. The SCAT5 test is done in two parts: first an on-field assessment, followed secondly by in-office assessment. A child version of the SCAT called ChildSCAT5 was created for use on children 12 and under. This version has minor changes to gear the assessments towards the children,

and includes a parent symptom report (Davis *et al.*, 2017).

On-Field Assessment

The on-field assessment is separated into 4 parts(Echemendia *et al.*, 2017b): 1) red flags (listed in Table 2.2), 2) observable signs, 3) Maddock memory assessment, and 4) the Glasgow Coma Scale (GCS). Additionally, there is also a cervical spine assessment to aid in determining whether the individual has sustained a spinal injury.

Table 2.2: SCAT5 red flag symptoms (Echemendia *et al.*, 2017b)

Red Flag Symptoms
Neck pain or tenderness
Double vision
Weakness or tingling/burning in arms or legs
Severe or increasing headache
Seizure or convulsion
Deteriorating conscious state
Vomitting
Increasingly restless, agitated or combative

In-Office Assessment

The SCAT5 in-office assessment is conducted in a quiet distraction-free area (Echemendia *et al.*, 2017b). This part of the assessment includes 1) a background on the individual including previous concussions and recovery for those injuries, 2) record symptoms and their severity, 3) cognitive screening to assess orientation, memory, and concentration, 4) a neurological examination is performed using a modified version of

the Balance Error Scoring System (BESS), which will be discussed in section 2.2.3, and 5) a delayed recall, which asks the participant how many words they remember from the given list in the cognitive screening section(Echemendia *et al.*, 2017b).

2.2.5 Concussion Recognition Tool

The CRT5 is a modified version of the SCAT5 which can be used by non-medically trained individuals to assess if an athlete has sustained a concussion and therefore should be removed from gameplay. This version consists of only on-field assessment, however, should any red flag symptom be observed then emergency assistance should be phoned immediately (Echemendia *et al.*, 2017a).

2.2.6 Heart rate variability

Heart rate variability (HRV) is a physiological marker that measures the variation in time intervals between individual heart beats, measuring between consecutive R peaks in the QRS complex. Electrocardiogram (ECG) is the gold standard in measuring HRV (Bishop *et al.*, 2018). Heart rate is primarily controlled by a balance between sympathetic and parasympathetic neural activity, and therefore HRV provides insight into the state of the autonomic nervous system (ANS) (Bilchick and Berger, 2006). Various parameters exist for calculating HRV, and they include but are not limited to averaging the R-R intervals (RR mean), a standard deviation representing total variance (SDNN) and a percent of intervals that vary by more than 50 millimeters (pNN50). Furthermore, a fast Fourier transform (FFT) can be used to calculate a power spectral density of the R-R intervals at different frequencies (Bishop *et al.*, 2018). The Task Force of the European Society of Cardiology and the North

American Society of Pacing and Electrophysiology divided heart rate into four frequencies: ultra-low frequency (ULF), very-low frequency (VLF), low frequency (LF) and high frequency (HF) bands, and these frequency bands can be used as metrics for measuring HRV (Shaffer and Ginsberg, 2017). Research has been done to show that HRV can potentially be used as a marker to assess concussion recovery, as many studies have shown a correlation between TBI patients and low HRV when compared to controls (Senthinathan *et al.*, 2017). Autonomic nervous system (ANS) dysfunction has been seen in early and late stages of mTBI recovery, and HRV is commonly used as an index to measure ANS function and its reflection on cardiovascular health (Purkayastha *et al.*, 2019).

A study done by Purkayastha *et al.* evaluated HRV using pNN50 of concussed athletes compared to healthy controls, measuring at 3 times post injury; 3 days, 21 days and 90 days (Purkayastha *et al.*, 2019). Results showed that the pNN50 values were significant lower in concussed individuals compared to controls at 3 days post injury, however the pNN50 values were comparable for both groups at 21- and 90-days post injury. This study supports that HRV can be used as a marker during early stages of recovery. A recent study done by Paniccia *et al.* observed how HRV differed in youth with concussion versus controls (Paniccia *et al.*, 2018). The study recruited participants aged 13 to 18 from community sports programs and performed baseline testing. Participants who sustained a concussion were then matched with an age-matched control for comparison. The study looked at SDNN, root mean square of successive R-R differences (RMSSD), pNN50, HF and HF normalized units (HFnu) as measures of HRV. Results showed that SDNN had no significant difference between injured and control groups. RMSSD showed a decrease 15 days post

injury followed by a continuous decrease until 30 days post-injury, whereas pNN50 measures showed increases with increasing days post injury. Furthermore, individuals with more symptoms showed higher pNN50. Finally, both HF and HFnu showed increasing values as days elapsed post injury, with significant differences seen between injured participants and healthy controls. This study also showed a relationship between increase in HRV in some measures with more reported symptoms, however this is mentioned to be inconsistent with other literature, which found lower HRV for individuals with more symptoms (Paniccia *et al.*, 2018). A study performed by Abaji *et al.* observed the effects of concussion on HRV in collegiate athletes, looking both at resting state and during physical exertion (Abaji *et al.*, 2016). Concussed athletes were paired with controls matched based on age, sex, and weight. The study found no significant differences in HRV during resting state, however during physical exertion is was found that concussed athletes had significantly lower HF power bands when compared to the control. Other measures of HRV such as RMSSD and RMNN did not show significant differences between concussed athletes and control. The authors suggest that concussed individuals show modifications in cardiac autonomic activity due to increased ANS stimulation (Abaji *et al.*, 2016). The studies highlighted above show that HRV can potentially be used as a marker for concussion recovery, however more research is required. Overall, there are inconsistencies in the literature which indicates more research must be conducted to further understand the relationship between concussion symptoms, recovery and HRV.

2.2.7 Neurological Imaging

As previously mentioned, concussions result in a functional injury more so than a structural injury, and therefore routine (i.e. clinical) structural neuroimaging does not greatly assist in concussion detection and recovery. Functional magnetic resonance imaging (fMRI) is a form of functional neuroimaging that detects brain activity by measuring relative changes of blood oxygenation and may be of more assistance in concussion diagnosis. As neural activity in regions of the brain increase, more ATP is required, leading to an increase in oxygenated blood flow to that area (Glover, 2011). The change in oxygenation drives the so called blood oxygen level dependent (BOLD) signal and can be measured using MRI. Depending on haemoglobin percent oxygen saturation (%O2sat) there will be varied modulation in the local MRI magnetic field homogeneity ($(\triangle B_0)$) surrounding the cells, and hence differing BOLD contrast, allowing fMRI to display regions of activation (Glover, 2011). fMRI can either be task-mediated or measured during what is termed resting state. During task-mediated fMRI, the participant performs a task, such as tapping their fingers, or is subjected to a form of stimulation (e.g. visual or sensory stimulation) and neural activity is measured. For resting state fMRI (rs-fMRI), the participant is asked to relax and 'think of nothing in particular', with eyes open. The resting state is used to show spontaneous neurological activity of the brain and is often used to analyze connectivity between various brain regions (Lv *et al.*, 2018). The default mode network (DMN) is commonly analyzed in concussion studies as it shows high functional connectivity during resting state (Chamard and Lichtenstein, 2018a). Both task-based and resting state methods have been used as approaches for evaluating concussion, with varying degrees of success (detailed below).

Task Based fMRI

Many studies have been done to compare fMRI results from concussed athletes with healthy/aged matched controls. Many studies have reported abnormal fMRI results which coincide with post concussive symptoms (PCS) (Chamard and Lichtenstein, 2018b). A study by Chen *et al.* showed athletes experiencing PCS had abnormal fMRI scans during working memory tasks, whereas concussed individuals without symptoms presented scans similar to an uninjured control (Chen *et al.*, 2004)(Chamard and Lichtenstein, 2018b). This demonstrates that fMRI could potentially be used to assess symptoms and symptom patterns throughout the recovery process. Jantzen *et al.* found a relationship between hyperactivation on fMRI scans and longer recovery periods for athletes with concussions (Chamard and Lichtenstein, 2018b)(Jantzen *et al.*, 2004). Athletes who showed hyperactivity during finger-tapping compared to baseline tests had a longer recovery period compared to those who had lower activation patterns. This information could be used to help track recovery periods however it would require frequent scans (not feasible due to cost and limited access to MRI scanners). Other studies have been done to test changes in activation patterns from concussed, compared to uninjured individuals, however most studies have been done on adults and not children. A study by Keightley *et al.* tested working memory in concussed children using fMRI (Keightley *et al.*, 2014). The results showed lower performance for visual and verbal working memory tasks in concussed children, as well as lower activation in numerous brain regions (Keightley *et al.*, 2014). This study shows a good start in analyzing functional deficits caused by concussion, however more research is required.

Resting State fMRI

Recently, rs-fMRI has been more frequently used to evaluate concussions. This approach is advantageous compared to task-mediated fMRI due to limitations in task-based studies, such as activity constraints, and variability between participants (Murdaugh *et al.*, 2018). Iyer *et al.* performed a study to look at connectivity between two regions of the DMN, the posterior cingulate cortex (PCC) and the medial prefrontal cortex (mPFC), as well as grey matter volume (Iyer *et al.*, 2019a). This study compared children with persistent post concussion symptoms (PPCS) to an age-matched control group. It was found that decreased functional connectivity within the DMN, and grey matter volume was linked to sleep disturbances, fatigue, and increased symptoms. It was also noted that normalization of the DMN was linked to improvements in cognitive function (Iyer *et al.*, 2019a). These results provided promise in the use of rs-fMRI alongside symptom scales for analyzing recovery in children with concussion. Furthermore, it was concluded by Iyer *et al.* that assessment of the connectivity between the PCC and mPFC can provide information on the likelihood of recovery from symptoms, and the evolution of PPCS (Iyer *et al.*, 2019a). A previous study by the same group noted the same results, that decreased functional connectivity in the DMN is indicative of increased persistent post concussion symptoms and decreased cognitive ability (Iyer *et al.*, 2019b). Churchill *et al.* performed a study on university level athletes to determine how functional connectivity corresponds to SCAT3 scores of concussed athletes compared to a non-injured controls (Churchill *et al.*, 2018). The results showed that decreased connectivity correlated with worse symptoms and lower scores on the SCAT3. It was also seen that most athletes with high scores showed similar functional connectivity to the control group.

Furthermore, functional connectivity was seen to be most affected in regions consistent with post-concussive symptoms such as cognition and memory (Churchill *et al.*, 2018). This study further supports the results found in the aforementioned studies, where decreased neural connectivity is suggestive of more severe symptoms. A study by Murdaugh *et al.* was done to assess alteration in functional activity and connectivity using rs-fMRI, and white matter integrity using DTI (Murdaugh *et al.*, 2018). The study was done on male high school football players with post concussion symptoms, and the goal was to compare acute results with those 21 days later, and to compare against healthy controls. The results showed concussed athletes had decreased functional connectivity in anterior regions of the brain but increased connectivity in posterior regions when compared to a control. Furthermore, hyperconnectivity was found in the left precuneus. The precuneus is an important node in the DMN as it mediates connectivity across the other nodes, and is involved in many cognitive tasks (Murdaugh *et al.*, 2018). [35]. It was suggested that hyperconnectivity is a compensatory mechanism to maintain similar neurological function to non-concussed peers (Murdaugh *et al.*, 2018). This study defends the argument that DMN connectivity is affected in symptomatic individuals. Rs-fMRI results in literature have proven to be a valuable method of evaluating post concussion symptoms and severity and can therefore be a key tool in assessing the recovery patterns of children with concussions.

Diffusion Tensor Imaging

Many studies have been done to assess changes in white matter integrity, and for evidence of diffuse axonal injury, after concussion (Wu *et al.*, 2018). Using DTI Murdaugh *et al.* found that upon initial scanning the concussed group had increased

isotropic diffusion compared to controls. This showed concussed individuals could have abnormal diffusion patterns, suggesting DTI could be a tool in assessing concussion recovery. (Murdaugh *et al.*, 2018). Wu *et al.* evaluated white matter FA and apparent diffusion coefficient (ADC; also called MD from tensor based measures) in concussed children in a longitudinal study, imaging at both 96 hours and 3 months post injury (Wu *et al.*, 2018). Concussed individuals were compared to two control groups, one with no injury and one with orthopedic injury. An orthopedic control group is regularly recruited to compare to TBI groups in order to rule out any factors or non-specific effects found within both groups that were caused by the injury, treatment, or emotional impact of the traumatic injury, and are therefore not specific to TBI (Wu *et al.*, 2018). A good example, but not the only factor, is the elevation in inflammatory markers (e.g. IL-6) seen following both mTBI and surgery. This study found that at 96 hours post injury, no difference was seen in FA and ADC, however at 3 months the concussed group showed significantly lower FA than both controls (Wu *et al.*, 2018). Furthermore, they found that certain brain regions showed higher ADC in the concussed group compared to the control (Wu *et al.*, 2018). Satchell *et al.* performed a study testing whether FA and MD were different in patients with a sports related concussion compared to age and sex matched controls(Satchell *et al.*, 2019). This study was done on patients ages 9 to 17. The results showed that there was no significant difference in FA or MD between concussed individuals and healthy controls. It was suggested that this may be due to the small sample size of the study, or the time after injury as they were scanned around 4 weeks post injury. This study highlights the fact that some DTI has shown changes in white matter whereas other studies do not, indicating that more research is required to assess the use of DTI as a

biomarker for concussion (Satchell *et al.*, 2019). Based on the studies analyzed, DTI could be useful in determining recovery patterns due to FA, ADC and white matter integrity, however it is still unclear as to what patterns are consistent in children with concussion.

2.3 Concussion Protocols of Intervention

Currently, many physical and cognitive tests are used to determine the appropriateness of returning to activity (RTA) or school (RTS). Alongside these tests, a series of stages have been determined to assist in the recovery process and provide a guide for individuals to follow to assist in their return. These guidelines have been developed by the Zurich Consensus and updated in the Berlin Consensus 2016(McCrory *et al.*, 2017), and CanChild for pediatric cases(CanChild, 2017). A primary limitation with these assessments is requirement for clinicians to oversee the progress, and most of the test administration. Providing a smart device app (i.e. smart phone) incorporated with ML that guides children through recovery stages would allow for more accessible determination of RTA/RTS without the constant requirement of clinician assistance, yet still provide clinician support through data tracking.

2.3.1 Stages of Recovery

A set of RTA guidelines were developed at the 4th International Conference on Concussion in Sport (the Zurich Consensus Guidelines), and RTS guidelines were added in the 5th Conference in Berlin (McCrory *et al.*, 2017)(Mccrory *et al.*, 2013). These guidelines were developed for adults suffering from sport related concussions

and define stages to guide in the recovery process. For both RTA and RTS, it is recommended that the stages commence after 24-48 hours of cognitive and physical rest, and patients spend a minimum of 24 hours on each stage prior to advancing. Should the individual have worsening symptoms at any point it is recommended they return to the previous stage (McCrory *et al.*, 2017). RTA guidelines consist of six stages leading up to a full return to activity, and are outlined in Table 2.4. RTS guidelines consist of four stages leading up to a full return to school, and are outlined in Table 2.3.

Table 2.3: Zurich Consensus Return to School Guidelines(McCrory *et al.*, 2017)(Mccrory *et al.*, 2013)

Stage	**Goal**
1	Participation in daily activities at home that do not induce any symptoms, such as reading or texting. It is recommended to start at 5-15 minutes and increase the time given there are no symptoms
2	Participating in school activities at home, such as homework or reading.
3	Part time return to school, with a gradual reintroduction to schoolwork.
4	Full return to school, by gradually returning to full participation until a full school day is achieved

Table 2.4: Zurich Consensus Return to Activity Guidelines(McCrory *et al.*, 2017)(Mccrory *et al.*, 2013)

Stage	**Goal**
1	Participation in light daily activities that do not invoke any symptoms.
2	Light exercise such as walking or light cycling.
3	Returning to light sport-specific training with no contact.
4	Full return to non-contact sport specific training, which can include resistance training.
5	Participation in full-contact sport-specific training.
6	Full return to activity.

One major limitation with the Zurich Consensus Guidelines is that they have been developed for adults, and therefore they do not apply to children under the age of 13 (Mccrory *et al.*, 2013). Understanding appropriate RTA and RTS guidelines is imperative for a safe recovery of each child. Current literature recommends that these guidelines should be conservative, cautious, and specific to children and youth (Dematteo *et al.*, 2015c). Returning to school or activity too soon can cause increased symptoms, longer recovery time, and a higher risk of re-injury. However, staying away from school for too long can severely affect academic standing, cause social isolation, and even increased depression (Dematteo *et al.*, 2015a). This juxtaposition requires careful determination of balance of activity with symptamology. Specific guidelines are required for these children as well as for toddlers. Based upon the Zurich consensus

and Berlin updates, DeMatteo *et al.* developed RTS and RTA guidelines, that are designed specifically for children (Dematteo *et al.*, 2015a)(CanChild, 2017)(Dematteo *et al.*, 2015b).The RTS protocol consists of five stages resulting in a full return to school, and RTA protocol consists of six stages resulting in a full return to activity. The RTS and RTA protocols can be seen in Tables 2.5 and 2.6.

Table 2.5: CanChild Return to School Guidelines (Dematteo *et al.*, 2015a)(CanChild, 2017)

Stage	**Goal**	**Activities**
1	Short term physical and cognitive rest	Regular daily activities that do not cause any symptoms. Limited screen time.
2	Simple cognitive activities	Walking, 15 minutes of screen time twice a day, socialization for 30 minutes a day.
3	Return to simple school routines	Attend class with a reduced class time, no tests and completing homework in 15-minute blocks for a maximum of 45 minutes. 15-minute blocks of screen time for a maximum of 1 hour. Regular school activities avoiding high stress situations such as the bus or lunchroom.
4	Normal routine, with some restrictions	Return to school fully, can miss one day a week if necessary. Can return to full homework, but limit to 1 test a week.
5	Return to school	Gradual return to full school activities.

Table 2.6: CanChild Return to Activity Guidelines(CanChild, 2017)(Dematteo *et al.*, 2015b)

Stage	**Goal**	**Activities**
1	Short term physical and cognitive rest	Regular daily activities that do not cause any symptoms
2	Light exercise	15-30 minutes of light exercise without increased symptoms, up to 1 hour a day.
3	Non-contact sport-specific activity	Continue stage 2 activity and add two 30-minute sessions of moderate physical activity
4	Non-contact sport specific practice	Continue stage 2 and 3 activity, add two 30-minute sessions of vigorous physical activity
5	Return to practice	Participate in practice and training without worsening symptoms
6	Return to sport	Guidelines are complete

2.3.2 Existing Technology for Concussion Tracking

Many tools and protocols described above exist to assist individuals dealing with a concussion. However, other approaches that assist in monitoring concussion and injury recovery exist, and could be useful for users to track their injury. Examples of these technologies are described below.

EQ Active Brain Tracking

A phone application (app) has been developed by Highmark Interactive which was designed to allow further insight into concussion and symptoms that are correlated with concussion (Starr, 2020). This app is available free for download on Apple and Android devices, however is subscription based (Highmark Interactive, 2020). The application tracks brain function by using a series of games as tests, which can then be used to assess the individual's brain function based on statistical norms. The concept surrounding this application is to give physicians an idea of how an individual is performing between visits, thereby providing guidance for that person's specific brain injury. Real-time results are also shown directly in the application which can provide areas of improvement for the individual to focus on (Highmark Interactive, 2020). Furthermore, the application has different settings for different age classes of people, for example kids versus adults. This product provides insight into symptoms and areas of improvement related to brain injury; however, it does not provide feedback on recovery or assist in the recovery patterns for individuals with concussions.

SMART Web Application

Kurowski *et al.* developed a web-based "Self-Monitoring Activity-restriction and Relaxation Training (SMART)" web application to be used as a tool in self-assessment of concussion symptoms and recovery (Kurowski *et al.*, 2016). This application promotes self-monitoring by incorporating guidance and psychoeducation of concussion, as well as self-management for cognitive and physical activities, and cognitive behavioral principles for aid in managing concussion for adolescents (Kurowski *et al.*, 2016).

The program consists of two parts: symptom and activity monitoring, and psychoeducational modules containing information related to concussion. The aim is for users to input symptoms and activity levels using a trajectory based PCSS, allowing input in text boxes. Furthermore, the application requests the user's plans for the next day of activities (i.e. increase, decrease, or maintain levels) (Kurowski *et al.*, 2016). This promotes self-assessment and provides the user with valuable skills, useful in recovery. Psychoeducational modules are available, providing information regarding the recuperation process, again promoting self-management of their recovery (Kurowski *et al.*, 2016). This application assists in managing and monitoring symptoms and activities throughout recovery, however it does not provide specific recommendations for RTA or RTS which can be highly beneficial for youth with this injury. Furthermore, this web application remains in the research stages and is not publicly available.

Concussion Tracker App

The concussion tracker app was developed by "Complete Concussion Management" (https://completeconcussions.com) to be used by coaches, clinicians, teachers, and patients. It was developed as a recovery tool to help assess concussion, report symptoms to clinicians, and track recovery (Complete Concussion Management, 2020). The main goal of the app is to connect the patient to their teachers, coaches and clinicians by providing updates on symptoms and milestones. The app also conveniently allows users to contact clinics and book appointments. Furthermore, it allows patients to update their teachers or coaches on their return to school or activity timelines by uploading documents from clinicians such as doctors notes or progress updates (Complete Concussion Management, 2020). The app is free for download and is available

on Apple and Android devices. This app requires complete clinician input for any milestone or RTA/RTS progress, and therefore it does not promote self-management throughout the recovery process.

2.4 Machine Learning

Machine learning (ML) is a type of artificial intelligence that learns based on previous experience to predict future outputs for new input data. ML uses a series of variables called "input variables" which represent measurable features of something, for example temperature, or size (de Mello and Ponti, 2018). Input variables are paired with an output variable, which represents the classification of the features. For example, if the model is trying to predict whether a patient has cancer, the input variables may include age, sex, and symptoms, and the output variable would be either yes or no. A dataset is split into two separate sets: the training set and the testing set. A training set is required to train a machine learning algorithm. It is a set of input/output variables that is used to train the algorithm to recognize patterns in the inputs that correlate to the output variable, allowing the algorithm to classify based on the input variables. Finally, a testing set is a set of inputs where the output is known but removed, and the algorithm classifies these inputs based on what it has learned from the training set. The results are then compared to the actual output to determine how well the algorithm performed. This type of ML is called supervised machine learning because the algorithm receives labelled data (de Mello and Ponti, 2018).

Many machine learning algorithms (MLAs) exists for classification and clustering of data, however some perform better than others given the data being used.

Prior to passing data through the algorithms, data must be analyzed and massaged using tools such as feature extraction. The two algorithms that were used in this study are Support Vector Machine Classifiers (SVC) and Random Forest Classifiers (RF). Both methods are supervised classifiers and can be used to predict the stages of concussion as classes. A study done by Bergeron *et al.* tested various MLAs to determine which algorithms performed best in analysis of recovery timelines for students having suffered a sports related concussion (Bergeron *et al.*, 2019). The study found that RF with 500 trees showed the most promise in predicting recovery times (Bergeron *et al.*, 2019), and therefore can be a good potential algorithm for predicting concussion recovery as well. SVC is the most commonly used technique in artificial intelligence in healthcare (Jiang *et al.*, 2017). Previous studies have used SVCs for applications such as neurological disease, diagnosis of cancer and early detection of Alzheimer's disease (Jiang *et al.*, 2017). Based on these examples, it is evident that SVCs are used in a wide variety of situations, and therefore can be a good potential algorithm for concussion recovery detection. A beneficial property of SVCs is that the solution is a global optimum and therefore always the best possible results (Jiang *et al.*, 2017).

2.4.1 Support Vector Machine Classifier

SVC is a supervised machine learning technique originally presented by Vapnik and Cortes (Cortes and Vapnik, 1995). SVC work by separating a binary set of data with a hyper-plane, generally called the 'maximal margin hyper-plane', that is maximally distant from both classes (Furey *et al.*, 2000). The hyperplane for a data

set $N = (x_i, y_i ... x_N, y_N)$ is defined by the function (Hastie *et al.*, 2009)

$$x : f(x) = x^T\beta + \beta_0 = 0 \tag{2.4.1}$$

where β is a unit vector. The margin, M, of the hyperplane is calculated by the following optimization:

$$max_{\beta,\beta_0}||\beta|| \tag{2.4.2}$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, i = 1...N \tag{2.4.3}$$

where $M = (1/||\beta||)$, and is the distance on either side of the hyperplane, making the full boundary 2M.

It is possible that some classes will overlap in the feature space, making it impossible to have a linear separation of the data. To account for this overlap, techniques have been developed to maximize M while allowing some points, each point represented by ξ_i, to fall on the wrong side of the margin (Hastie *et al.*, 2009). The following equation accounts for these overlapping points.

$$\min ||\beta|| \text{ subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall \\ \xi_i \geq 0, \Sigma\xi \geq \text{constant} \end{cases} \tag{2.4.4}$$

The constant in the equation above can be replaced by a 'cost' parameter C, which is inputted by the user and can be changed to improve the output of the algorithm.

SVC classification can then be performed by using an inner dot product of transformed feature vectors $h(x)$. The overall decision function is derived from Lagrange multipliers and is as follows

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \tag{2.4.5}$$

where α is a constant between 0 and the cost parameter (Hastie *et al.*, 2009).

Kernels

Support vector machines are designed to separate data that is linearly separable, however this is not always the case. Kernels were therefore designed as a method of mapping data into a new space that can be separated linearly as much as possible (de Mello and Ponti, 2018). The optimization of SVC is calculated using a dot product (Equation 2.4.5), and these dot products are easily replaced by the kernel function to transform the data to a higher dimensional feature space (de Mello and Ponti, 2018). The three most common kernel functions are the polynomial kernel, the radial basis kernel, and the sigmoid kernel.

The polynomial kernel transforms features to a higher dimension using the following equation(de Mello and Ponti, 2018)

$$k(x, y) = \langle x, y + c \rangle^d \tag{2.4.6}$$

where d is the kernel order and c is a parameter that trades off the influence of lower and higher order terms of the polynomial. Both parameters are set by the user. This kernel maps the data to a new feature space, where the dimension of that

feature space is determined by the order of the kernel. For example, a 2-D feature space, using the polynomial kernel of order 2, will be mapped to a 4-D space (de Mello and Ponti, 2018).

The RBF kernel transforms the data using the following equation(de Mello and Ponti, 2018)

$$k(x, y) = exp(-\gamma||x - y||^2), \gamma > 0 \tag{2.4.7}$$

where γ is the kernel parameter, which is inversely related to the standard deviation of the Gaussian distribution, and measures how significant the length of the difference vector between examples is to the similarity of those examples. A higher gamma value would require examples to be closer together for them to be deemed similar, whereas a low gamma would find similarity between examples that are further apart(de Mello and Ponti, 2018).

Finally, the signoid kernel maps data to a higher dimension using the following equation(de Mello and Ponti, 2018).

$$K(x, y) = tanh(-kx, y\rangle + c) \tag{2.4.8}$$

Where k $>$ 0 and c $<$ 0. K is a modifier to the magnitudes of the dot product, whereas c is a shifting parameter for the curve (de Mello and Ponti, 2018). Sigmoid kernels are popular for SVC models since they originate from neural networks (Lin and Lin, 2003).

2.4.2 Random Forest

Classification and regression trees (CART) are a powerful classification tool and is the foundation algorithm from which random forest classifiers are built. CART splits observations at the 'nodes' of the tree based on certain conditions of the predictor variables, and makes predictions to classify these observations. A common method that is used in classification trees for splitting the data at the nodes is called the Gini Index (Hastie *et al.*, 2009). First, the proportion of class k observations is found at the node, m, with the following equation

$$\hat{P}_{mk} = \frac{1}{N_m} \sum_{x_i \epsilon R_m} I(y_i = k) \tag{2.4.9}$$

where R_m represents the region of the node and N_m represents the observations. The Gini Index is then calculated for each proportion as a measure of impurity of the node, and is calculated by

$$\sum_{k \neq k'} \hat{P}_{mk} \hat{P}_{mk}{}'(1 - \hat{P}_{mk}) \tag{2.4.10}$$

The class with the lowest Gini Index would be the splitter for that node and the observations would then be split based on that class. This allows for each split in the decision tree to be as pure as possible with the lowest misclassification.

RF classifiers are a type of decision tree classifier that is built on a method called bagging (Zhou, 2012). The bagging method is a combination of bootstrap and aggregating techniques. Bootstrap sampling is done to obtain a subset of data from the training set. This method starts with a training set of n samples and creates a new subset of n samples by sampling with replacement. Each subset is used to train

a base classifier, creating a series of decision trees with a predicted output. A voting method is then used to aggregate the subsets into a final predictor, meaning that the algorithm will vote on which predictor sample is the best overall prediction of the system (Zhou, 2012). The RF algorithm differs from bagging by using a randomly selected set of features to predict the output. At each split in the decision tree, the algorithm will select a random subset of features from the total number of features in the data, and will proceed with the normal conventional split selection based on the randomly selected features. At each split in the tree, the randomly selected features will be different (Zhou, 2012). The number of features selected is specified by the user when modelling the algorithm, and this value can be manipulated to get the best possible results from the model. The value can be calculated as approximately

$$M = \sqrt{P} \tag{2.4.11}$$

where M is the subset of features randomly selected, and P is the total number of features in the dataset(Zhou, 2012).

Chapter 3

Methods

This chapter describes the overall methodology taken throughout the project from pre-processing to result visualization.

3.1 Data

Data used in this study was previously collected by Professor DeMatteo and her team at CanChild. The study was called "Safely returning children and youth to activity after concussion", and was done to develop the RTA and RTS stages. Two separate datasets were collected from children with concussion. Dataset A consisted of information about the symptoms each child was experiencing. This dataset initially had data from 134 participants, 61 male and 73 female. Of those 134 participants, 94 had data acquired for dataset B, which consisted of raw accelerometry data. Participant ages ranged from 6 to 18, however data collected from children under 10 years of age was removed for the present study. This was done because the initial scope of the current research was participants aged 10 to 18, and the initial target population of

the APP was children over 10 years of age. Usable participants were determined by whether they had data in both datasets A and B, both recorded on the same day. In total, 67 participants had overlapping data which could be used for this study. Research ethics approval (Hamilton Integrated Research Ethics Board, HiREB) was attained prior to any data collection.

3.2 Software

Programming was done on two separate graphics processing unit (GPU) computing clusters. The first machine was used for all work done on the initial data including data processing and modelling. This machine had an Intel Core i9-9820x 3.30GHz central processing unit (CPU), with 64GB RAM, and an NVIDIA GeForce RTX 2080 Ti graphics card with 12GB memory. All coding on this machine was done using the open-source programming language Python version 3.8 (Python Software Foundation, https://www.python.org/). The editor used to write and execute code was Jupyter Notebook, an open-source interactive environment derived from iPython(Jupyter, 2021).

The second machine used was used for all work done on the new data including data processing and modelling. This system had an AMD Threadripper 3960X 3.8 GHz 24-Core CPU, with 64GB RAM. All programming on this machine was done using Python 3.8, running scripts from the command line.

Many packages were used, all of which were open source and accessed directly through Python. The main packages used can be found in Table 3.1.

Several functions included in the packages mentioned above were used, and are mentioned in the appropriate sections below.

Package	**Purpose**
Numpy	Used to work with numerical data. (Harris *et al.*, 2020)
Pandas	Used for data manipulation of time series and tabular data.(pandas development team, 2020)(Wes McKinney, 2010)
Matplotlib	Used for plotting and visualizing results.(Hunter, 2007)
Sklearn	Used for pre-processing and machine learning functions. (Pedregosa *et al.*, 2011)
Scipy	Used for statistic calculations. (Virtanen *et al.*, 2020)
Seaborn	Used for statistical data visualisation. (Waskom, 2021)

Table 3.1: Main Python packages used and their purpose for use.

Early stage programming was done using MATLAB (MathWorks, https:// www.mathworks.com/ products/matlab.html). Only symptom data cleaning was done using this software. All other programming was done with Python.

3.3 Data Processing

Data collected in the study included accelerometer data, symptom recordings and stage of recovery self reported by the participants. Symptom and accelerometer data was processed and combined in preparation for modelling and stage prediction. The flow of the project is detailed in Figure 3.1, and the methods for each component are described in this chapter.

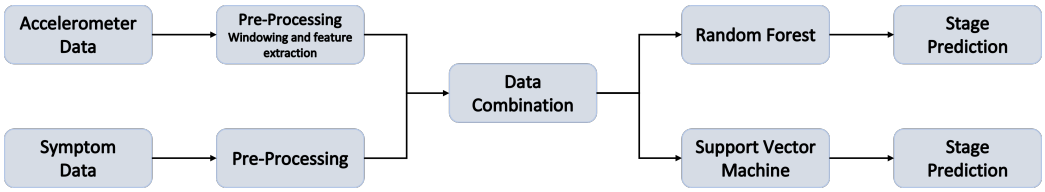


Figure 3.1: Flow chart displaying modelling pipeline from data pre-processing to stage prediction.

3.3.1 Symptom Data

The acquired symptom data was recorded every 48 hours by participants using the PCSS. In addition, a few other questions regarding their concussion symptoms, such as the level of cognitive activity they did, or how their school day went, were also recorded. A total of 27 questions from the symptom surveys were used as features for the machine learning algorithms. Table 3.2 lists the features and possible responses initially recorded with each feature. All other entries recorded in the initial study were removed as they were not relevant to the current Back2Play study. Furthermore, some participants had data recorded once biweekly. This was removed for all participants since very few users had entries, and therefore there was not enough useful data.

Participants could specify that they had not experienced any symptoms in the past 48 hours, and if this was the case they did not record 0 for all of the symptoms. Therefore, if a participant specified they had no symptoms then all 'no entry' spaces, represented by a NaN, were changed to 0. This was done using nested "for-loops" with a conditional statement, which checked if the value at the index for 'no symptoms in the last 48 hours' was zero, then all symptom values for that row marked as NaN would be set to 0. Entries that had some missing information were removed since a complete dataset is required for machine learning. Participant sex and age initially were only recorded with the first entry, and therefore were added to each entry to fill

Feature	**Possible Response**
Headache Nausea Balance Problems Dizziness Fatigue Drowsiness Sensitivity to Light Sensitivity to Noise Irritability Sadness Nervousness Feeling Emotional Feeling Slowed Down Feeling Foggy Difficulty Concentrating Difficulty Remembering Visual Problems Getting Confused with Direction or Tasks More Clumsy Difficulty Answering Questions Neck Pain Sleep Problems	Number from 0 to 6 where 0 is no symptom and 6 is severe
Degree of Feeling Differently	Number from 0 to 4 where 4 is a major difference
Total Score	Sum of all symptom recordings
Cognitive Activity Level	Score 1 to 5 where 1 is none and 5 is full.
School Day Level	Score 1 to 5, 1 is school not in session and 2 to 5 range from not going to school to a full return.
Treatment	1 or 0 if they received non-study treatment in the past 48 hours

Table 3.2: Symptom Features

in those missing values. This was done using a function 'fillmissing' in Matlab which fills the NaN value of that column with the value above it. Any symptom response

from 1 to 6 was changed to a 1 representing that the participant was experiencing that symptom. This was done to limit the possible responses for the majority of the questions, and to keep the features as consistent as possible with the phone application, which asks participants which symptoms they are experiencing and not a ranking. Values were changed manually for each column in Microsoft Excel. Code for this section can be found in Appendix A.1.1.

3.3.2 Accelerometry Data

Participants in the study wore an ActiGraph (ActiGraph, https://actigraphcorp.com/) model wGT3XBT ActiLife Software v6.13.3 with firmware v1.2.0 around their waist throughout the day. The Actigraph collected accelerations in the x, y, and z axis in gravitational units, at a sampling frequency of 30Hz. The Actigraph was worn continuously throughout the day, unless removed for activities such as showering, swimming, or sleeping. A wear journal was kept by participants recording when the device was removed. Non-wear times recorded in the journal were used to remove data recorded when the device was not worn. Journal recordings for non-wear had length of time in minutes for each period of time the watch was worn and removed. An example of the wear time journal can be seen in Figure 3.2, and the time between each period of wear and non-wear is labelled a 'Length'. This value was used to remove the non-wear samples from the accelerometer data. The length was added with each previous value to get a vector with the cumulative time from the beginning that the watch had been worn/removed. This was done using the 'cumsum' function in Python. Each value was then multiplied by 60 to convert from minutes to seconds, and again by the sampling frequency of 30 to convert to samples.The last value was

	Start_time	Stop_time	Wear	Length	Use
3	12/11/2015 5:00	12/11/2015 7:31	Non-Wear	151.5	FALSE
4	12/11/2015 7:31	12/11/2015 21:58	Wear	866.5	TRUE
5	12/11/2015 21:58	12/12/2015 8:40	Non-Wear	642.5	FALSE
6	12/12/2015 8:40	12/12/2015 15:45	Wear	424.5	TRUE
7	12/12/2015 15:45	12/13/2015 7:43	Non-Wear	958	FALSE
8	12/13/2015 7:43	12/13/2015 18:28	Wear	645	TRUE
9	12/13/2015 18:28	12/22/2015 16:58	Non-Wear	12870	FALSE

Figure 3.2: Sample Wear-Time Journal

changed to the total length of the signal to ensure all samples after the last time of removal were removed. A "for-loop" was then used to create an index vector that would be used to label the bounds of which values needed to be removed. The "for-loop" iterated over every 2nd value of the wear length vector and took the current value an as the start bound and the next value in the wear length vector as the stop bound. Once all bounds for removal were recorded as a vector, these indices were used to remove all values in the accelerometer signal between those bounds. Figure 3.3 shows this process. Code for non-wear removal can be found in Appendix A.2.1 .

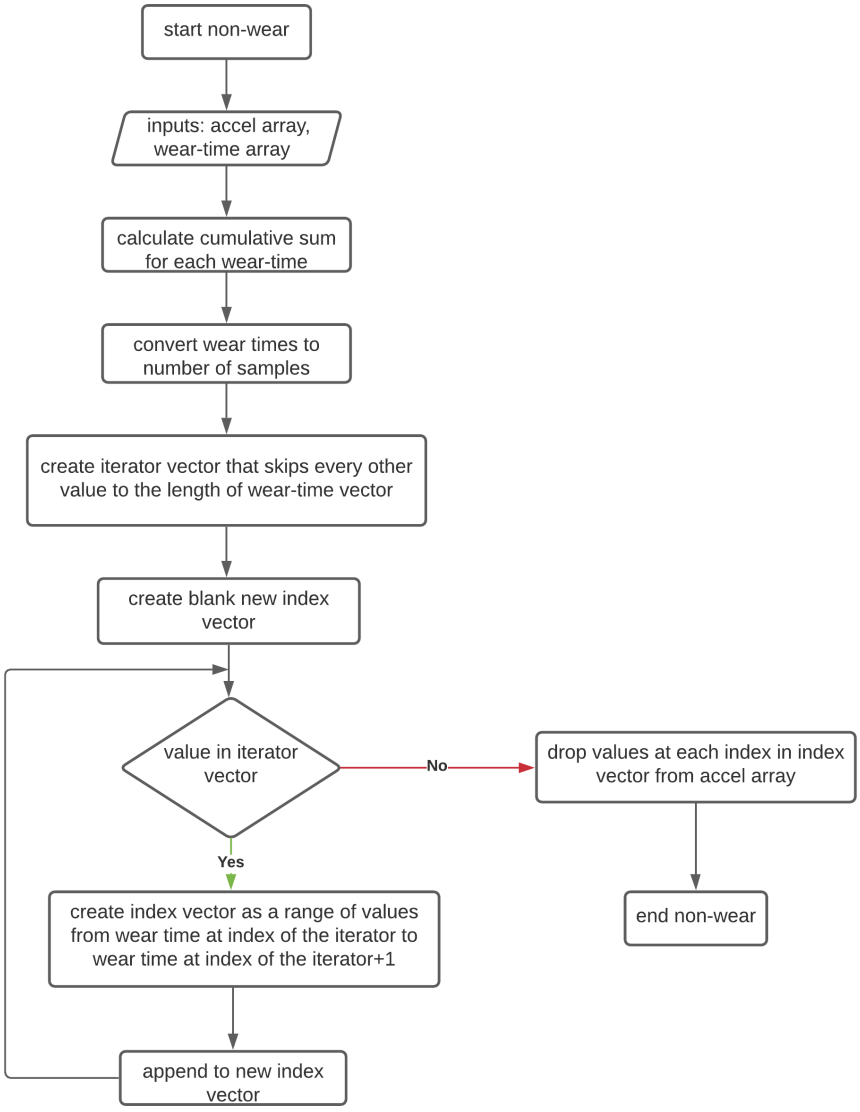


Figure 3.3: Flowchart showing process for removing non-wear times.

Once all non wear times were removed, the remaining dates with accelerometer recordings were compared to dates in which that participant had both symptom and stage recordings. Given the overall goal of the study being stage prediction, accelerometer recordings with no corresponding symptoms or stages were not useful.

Therefore, only dates in which participants had all 3 types of data were windowed for feature extraction. This was done in Python by comparing the date stamps of the symptom data to the date stamps on the accelerometer data. First, a date vector was created of all of the dates that had symptom recordings, and each date was saved as a unique variable. The dates from the accelerometer data were then converted into a date value without a timestamp, and saved in a separate vector. The accelerometer data was then split into each individual day by finding the dates that matched the unique signal date variables. An index vector was then created to find the indices for all of the accelerometer samples that needed to be kept. The index vectors were combined using numpy's 'concatenate' function creating a final vector of which indices from the original accelerometer file needed to be kept for windowing. Using the python function 'loc', all indices stored in the index vector were taken from the accelerometer file and saved as a new matrix which contained all useful data. This data could then be used for windowing. Code for this methodology can be found in Appendix A.2.2

Windowing

A sliding window technique was used to split the data and prepare it for feature extraction. A sliding window is a technique where a window time is chosen and data is separated in segments where each segment contains the defined time worth of data. The window is then moved with a specified overlap and then groups the next segment of data. Many studies use a 50% overlap between windows to ensure no movement is lost between windows.This technique often used in human activity recognition machine learning studies when accelerometer data is being analyzed to predict body movement (Abdull Sukor *et al.*, 2018)(Lavanya and Mallappa, 2019). Window sizes

often vary, and in literature it has been seen to range from 1 second to upwards of 60 seconds (de Almeida Mendes *et al.*, 2018). To window the data, a function provided by a colleague of the lab was used (A. Simons, personal communication, September 15, 2020). This function uses an initial condition to determine if there is a shift, and if so a "while-loop" is used to iterate over the signal vector as long as there are samples left in the signal to window. If so, then for each iteration of the loop the number of samples in the window is taken from the signal vector and added to an array using 'numpy.concatenate', and the iterator is increased by the window shift each time. Once completed, an array containing all of the windows concatenated together as well as a variable set as the total number of windows is returned from the function. Figure 3.4 illustrates this function. Each x, y, z axis and a vector containing the date for each sample is sent as inputs into the function individually, as well as the sampling frequency, window length, and window shift. The code used for windowing the signals can be found in Appendix A.2.2.

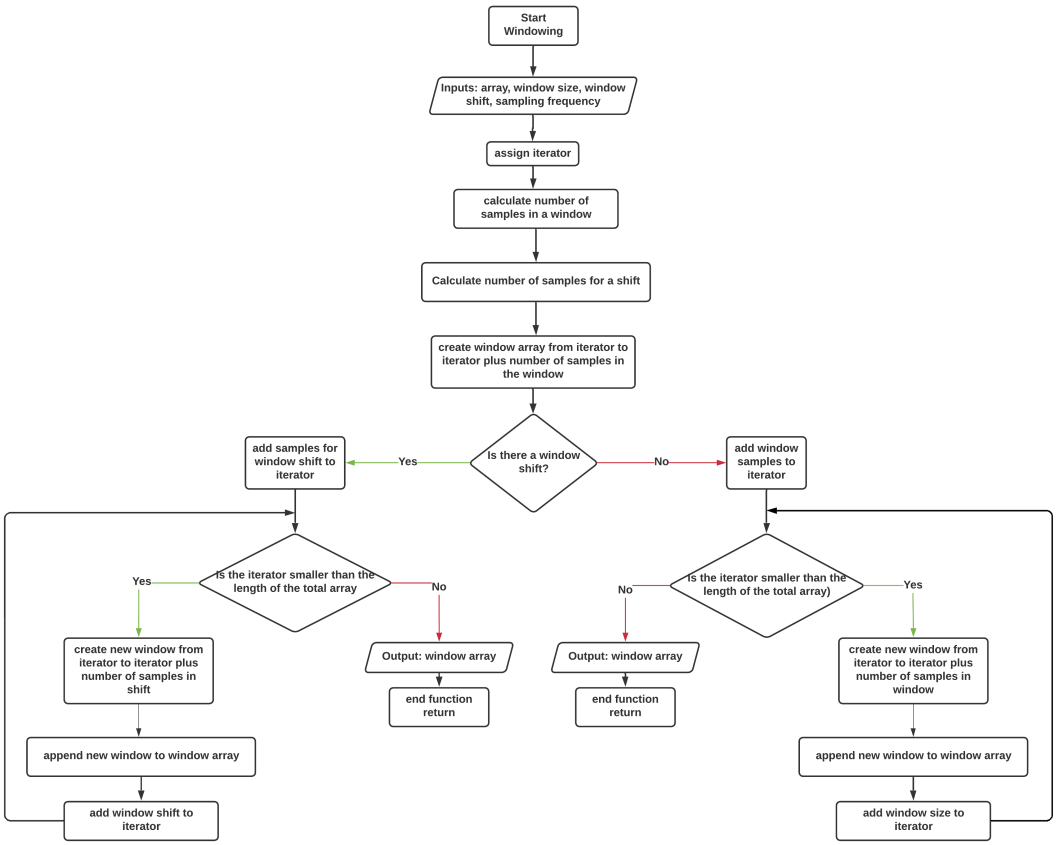


Figure 3.4: Flowchart describing windowing function process.

The window size selected was 30 seconds with a 50% overlap between windows. This time was selected by consulting Professor Carol Dematteo, principal investigator in the Back2Play study, determining that 30 seconds was an appropriate amount of time to capture the overall type of movement the child is doing in each window. A shorter window could potentially misinterpret the nature of the movement the

child is doing, whereas a longer window could be too vague and miss smaller bouts of movement during analysis. The 50% overlap was selected to ensure that no movement is lost between windows, and it also provides more data points for machine learning. Once the data was segmented into windows, feature extraction could be performed.

Feature Extraction

An integral part in machine learning is extracting features that provide good data for the models to predict from. Human activity recognition (HAR) papers were searched to determine features which have been seen to work well to predict body movement. A total of 17 statistical features were chosen to calculate over each window of data, all of which have shown promise in predicting human activity in previous studies(Figo *et al.*, 2010) (Abdull Sukor *et al.*, 2018) . All features are listed in Table 3.3. Furthermore, these features were selected as they were relatively simple to calculate and would be easily transferable to a smartphone application without requiring too much computational power, which is the eventual end goal of the Back2Play study. The features were calculated using a "for-loop" to loop through the windowed signal and calculate all features for each window of data. The specific method for calculating each feature is described below, and the code for this section can be found in Appendix A.2.2.

Feature	**Axis**
Standard Deviation	One for each x, y, z axis
Mean	One for each x, y, z axis
Maximum	One for each x, y, z axis
Minimum	One for each x, y, z axis
Mean Absolute Deviation (MAD)	One for each x, y, z axis
Signal Magnitude Area (SMA)	One for all axes combined
Signal Vector Magnitude (SVM)	One for all axes combined

Table 3.3: Accelerometry Features

The standard deviation (SD) was calculated in Python using the 'numpy.std()' function, which finds the square root of the variance, and is calculated using Equation 3.3.1(Ebner *et al.*, 2021). One SD was calculated for each window in the x, y, and z axis.

$$SD = \sqrt{1/N(\sum_{i=1}^{N}(x_i - \overline{x_i})^2)} \tag{3.3.1}$$

where $N = number\ of\ samples\ in\ the\ dataset$

$x_i = individual\ values\ of\ the\ dataset$

and $\overline{x_i} = mean\ of\ x_i$

The mean was calculated in Python using the 'numpy.mean()' function, which calculates the arithmetic average of the dataset and is calculated using Equation 3.3.2 (NumPy, 2021). One mean was calculated for each window in the x, y, and z axis.

$$MEAN = 1/N \sum_{i=1}^{N}(x_i) \tag{3.3.2}$$

where $N = number\ of\ samples\ in\ the\ dataset$

and $x_i = individual\ values\ of\ the\ dataset$

The maximum and minimum value from each window in the x, y, and z axis were also taken as features. The Python functions max() and min() were used to find the largest and smallest values in the window.

The median absolute deviation (MAD) was calculated in python using the function 'scipy.stats.median_absolute_deviation()', which calculates the average absolute difference between all of the values in the window using Equation 3.3.3. This statistical measure was used as an extracted feature in some publicly available HAR datasets including UCI HAR Dataset (Davide Anguita and Reyes-Ortiz, 2013) and the WISDM dataset (Weiss *et al.*, 2019), and has shown promise as a feature in predicting HAR. One MAD was calculated for each window in the x, y, and z axis.

$$MAD = 1/N(\sum_{i=1}^{N} |x_i - \overline{x_i}|) \tag{3.3.3}$$

where $N = number\ of\ samples\ in\ the\ dataset$

$x_i = individual\ values\ of\ the\ dataset$

and $\overline{x_i} = mean\ of\ x_i$

The signal magnitude area (SMA) and signal vector magnitude (SVM) were both chosen as metrics to analyze the complete signal by combining all three axes. Both features have been found useful in previous HAR studies (Figo *et al.*, 2010)(Abdull Sukor *et al.*, 2018). Both the SMA and SVM were calculated directly in Python using

Equations 3.3.4 and 3.3.5(Abdull Sukor *et al.*, 2018).

$$SMA = 1/N(\sum_{i=1}^{N} |x_i| + \sum_{i=1}^{N} |y_i| + \sum_{i=1}^{N} |z_i|) \tag{3.3.4}$$

where $N = number\ of\ samples\ in\ the\ dataset$

and $x_i, y_i, z_i = individual\ values\ of\ the\ dataset\ in\ each\ axis$

$$SVM = \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{3.3.5}$$

where $x_i, y_i, z_i = individual\ values\ of\ the\ dataset\ in\ each\ axis$

All features mentioned above were calculated for each window and could then be combined with the daily symptom and stage recordings. The code used to extract features can be found in Appendix A.2.2.

3.3.3 Data Fusion

Once feature extraction was performed for every window of data, the final data set for machine learning was completed by combining the symptom data with the accelerometer data, and attaching the appropriate RTA and RTS label to each feature set. Features were encoded as integer values of 0 or 1 for all symptoms and whether or not they received treatmet, values between 0 and 120 for total score, and values between 0 and 7 for ranking of cognitive activity, school day, and degree of feeling differently. Since symptoms were recorded by day, feature sets for each 30 seconds of one day were all labelled with the symptoms and RTA/RTS labels for that day. Features were standardized to a value between -1 and 1 using the python function "StandardScalar", which subtracts the mean and normalizes by the variance (Equation 3.4.1. To combine the data sets, a nested "for-loop" with nested conditional

statements were used. The first "for-loop" iterated over the length of the symptom data set, and the inside "for-loop" iterated over the length of the feature data set. Within both "for-loops", the first conditional compared participant numbers. If the participant was the same at that value, then another conditional compared the dates. If the date was also the same, then the feature and symptom data at that row were combined using the 'pandas.concat' function, and added to a new vector using the 'append' function. This was done to compare each row in both data sets creating one final matrix containing all necessary data. Figure 3.5 shows this process. The final data set included 67 participants with a total of 844408 rows of data, where each row is a feature set associated with one label for RTA and one for RTS. Participant ages ranged from 10 to 18, and there were 42 female and 25 male participants. A sample of data used as inputs to the algorithms can be see in Figure 3.6. Code for this section can be found in Appendix A.2.3.

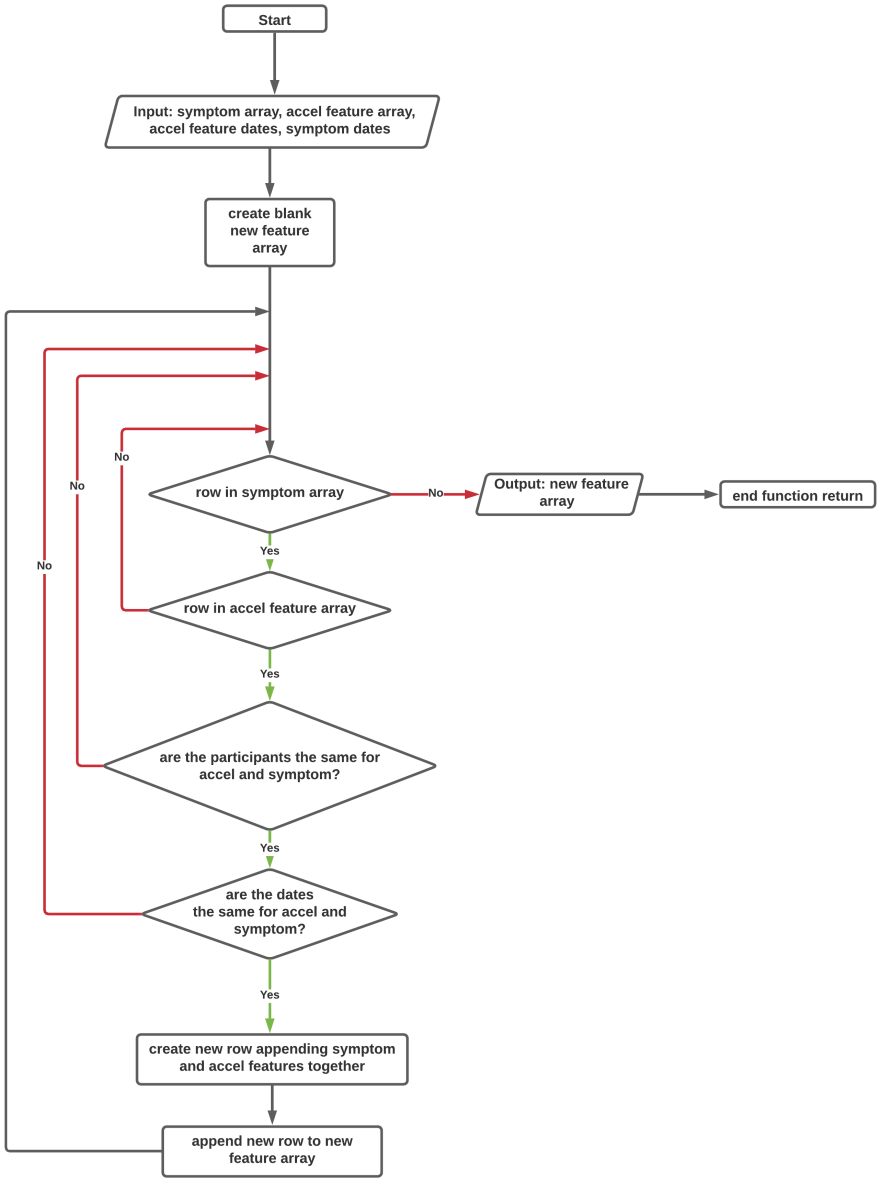


Figure 3.5: Flowchart describing data fusion process.

Std x	Mean x	Max x	Min x	MAD x	Std y	Mean y	Max y	Min y	MAD y	...	pcsiy48_cf	pcsiy48_cl	pcsiy48_sl	pcsiy48_dif
4.090481	-1.146812	0.199950	-7.688496	-0.750525	1.505694	-0.548711	0.172015	-1.656775	-0.584650	...	1.878802	-0.518542	-0.501429	-0.109442
3.088875	-2.405167	0.199950	-7.688496	2.678420	2.003085	-0.539163	0.202874	-1.656775	2.721896	...	1.878802	-0.518542	-0.501429	-0.109442
8.953492	0.056823	3.201097	-2.750049	15.091016	2.075050	-0.226851	0.894719	-1.184735	3.267306	...	1.878802	-0.518542	-0.501429	-0.109442
4.426546	1.749259	3.201097	-0.882472	4.720999	2.405106	0.186220	1.136616	-1.184735	2.713374	...	1.878802	-0.518542	-0.501429	-0.109442
2.936800	0.352018	1.892946	-0.882472	3.648876	4.963804	-0.377144	1.136616	-0.960229	11.942728	...	1.878802	-0.518542	-0.501429	-0.109442
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.325833	1.100561	0.321225	0.868073	-0.306888	-0.263016	1.068421	0.584136	1.097748	-0.311945	...	-0.532254	-0.518542	-0.501429	1.819330
-0.257969	1.312693	0.269481	0.914226	-0.602646	-0.474627	1.024113	0.564226	1.097748	-0.516474	...	-0.532254	-0.518542	-0.501429	1.819330
0.289930	0.962759	0.269481	0.894446	-0.159009	-0.276962	1.091771	0.564226	1.109261	-0.260813	...	-0.532254	-0.518542	-0.501429	1.819330
-0.755348	0.671219	-0.115364	0.887853	-0.602646	-0.642539	1.154384	0.541331	1.194650	-0.465342	...	-0.532254	-0.518542	-0.501429	1.819330
0.175505	0.754098	0.308289	0.340602	-0.491737	-0.320094	1.134555	0.751373	1.068006	-0.516474	...	-0.532254	-0.518542	-0.501429	1.819330

Figure 3.6: Sample of data from the training set, after standard scaling.

3.4 Modelling

Once the feature matrix was completed, modelling could be done. Data was split into a training and testing set, ensuring that no participants had data in both sets. To split accordingly, the 'GroupKFold' function from the SKLearn Python library was used. This function splits data into a desired number of folds, ensuring that no groups occur in multiple folds. A split value of 3 was used to separate data into 3 folds, where 2 folds were the training set and one was the test. This value of folds was chosen to have approximately 70% of the data in the training set and 30% of data in the testing set. Labels were separated from the data and saved in a unique vector, one for the training set and one for the testing set. This was done for both RTA and RTS data.

Features were then normalized using the function 'StandardScalar' from the SKLearn pacakge This function standardizes the data by subtracting the mean of the training set and dividing by the standard deviation of the training set, as seen in Equation 3.4.1. Standardizing was done to ensure all features were approximately all scaled the same to ensure no feature had more weighting in the classifications than others.

$$z = (x_i - u)/s \tag{3.4.1}$$

$$\text{where } x = individual\ value\ of\ the\ dataset$$

$$u = mean\ of\ the\ training\ set$$

$$\text{and } s = standard\ deviation\ of\ the\ training\ set$$

Once standardization was complete, data was ready for modelling. The 'RandomForestClassifier' function was used form the SKLearn 'ensemble' package to create a RF classifier, and the 'SVC' function from the SKLearn 'svm' package was used for the SVC classifier.

3.4.1 Hyperparameter Tuning

Hyperparameter tuning was done for each model to find the best combination of parameters that would yield the best results for the data. A cross-validation grid search was used to tune hyperparameters. For the random forest classifier, a randomized grid search was first used to narrow down the parameters, followed by a full grid search. K-Fold Cross validation (CV) is a method of training a model by splitting the training set into K folds, where K-1 folds act as the training data, and 1 fold acts as the validation set to test the accuracy of the model. This occurs K times, where each fold acts as the validation set one time (Hastie *et al.*, 2009). A CV grid search does a CV for each possible combination of hyperparameters, and determines which set of parameters yields the best model accuracy. This method of hyperparameter tuning was determined to be the best to get highest model accuracy for SVC in a study done by Duarte and Wainer(Duarte and Wainer, 2017). The number of folds for a CV is manually selected, however is usually either 5 or 10(Hastie *et al.*, 2009). In this study, 5-fold was chosen for all models to decrease length of time required for training model to run, which was necessary due to time and resource limitations.

For both RF and SVC, the parameter 'class_weight' was set to balanced, allowing the models to weight the classes differently depending on the amount of data was available for each class. This was done to try and level out the imbalance between classes.

Random Forest

For the RF classifier, a randomized grid search was used prior to doing the grid search. The randomized search did 50 combination trials to find approximately which values for each parameter yielded the best results. Values around those numbers were then used to do a full grid search to find the overall best set of hyperparameters for the data-set. This was done to decrease the computational time, since the grid searches were long processes. To do both the randomized grid search and the full grid search, the functions 'RandomizedSearchCV' and 'GridSearchCV' from the SKLearn library were used. Parameters that were tuned are described in Table 3.4.

Support Vector Machine

For the SVC, just a grid search was done using the function 'GridSearchCV from the SKLearn library. Table 3.5 describes the parameters tuned in the grid search. The Radial Basis Function Kernel was chosen as this is one of the most popular kernels used, and works well when dealing with non-linear and large data-sets(Prajapati and Patle, 2010).

Parameter	Description	Possible Values
n_estimators	Number of decision trees in the forest.	10 evenly spaced values between 10 and 1500
max_features	Number of considered features at each split.	'sqrt'
max_depth	Maximum number of levels in the tree	11 evenly spaced values between 10 and 110, or No maximum
min_samples_split	Minimum number of samples required to split a node.	2, 5, 10
min_samples_leaf	Minimum number of samples required for a leaf node	1, 2, 4
bootstrap	Method for selecting samples from the training set	True, False
class_weight	Weights the classes to balance the models	'balanced'

Table 3.4: Random Forest Hyperparameters Tuned for Modelling

Parameter	Description	Possible Values
Kernel	Kernel used for SVC	'rbf' - radial basis function
Gamma	Kernel parameter	1e-1, 1e-2, 1e-3, 1e-4
C	Represents how many values can be misclassified	1, 10, 100, 1000, 10000
class_weight	Weights the classes to balance the models	'balanced'

Table 3.5: Support Vector Machine Hyperparameters Tuned for Modelling

3.4.2 Modelling and Results

Once hyperparameters were selected and trained on the training set, the models were tested using the testing data set. Models were first made to predict all five RTS stages and all six RTA stages. Predictions made by the model using the test set were found using the 'decision_function' feature of the SVC model, which returns an

array for each set of samples showing a score of how far the sample for each class would be from the decision boundary(scikit learn, 2021b). The largest positive value is decided as the class most likely to be predicted, and therefore the overall prediction for each sample set is determined using "np.argmax". For the RF models, the function "predict_proba", which computes the probability for each class to be predicted(scikit learn, 2021a). "argmax" is also used in this instance to determine which class is predicted for each instance in the test set.

After predictions were made, the results were analyzed using multiple methods. Firstly, a confusion matrix was used to display classification results. A confusion matrix is an n by n table, where n is the number of classes being predicted. The confusion matrix has a value in each index of the matrix, showing for each true case which classes were predicted. The values along the diagonal are the correct predictions, and all other values are false predictions. An example confusion matrix can be seen in Figure 3.7, where green shows the correct predictions and red shows the incorrect predictions. The confusion matrix for each model was printed using the python function "confusion_matrix" from the "SKlearn.metrics" package. Confusion matrices were plotted as a heat map for visualization purposes using Seaborns 'heatmap' function.

From the confusion matrix, values such as accuracy, f1-score, recall and precision can be calculated. Precision and recall were found using the "classification_report" function from the "SKLearn.metrics" package. Precision measures the accuracy of the model by calculating how many positive predictions were actually correct, whereas recall measures how many of the actual positives were predicted (Developers, 2020). Recall can also be called sensitivity. The formulae for precision and recall can be seen

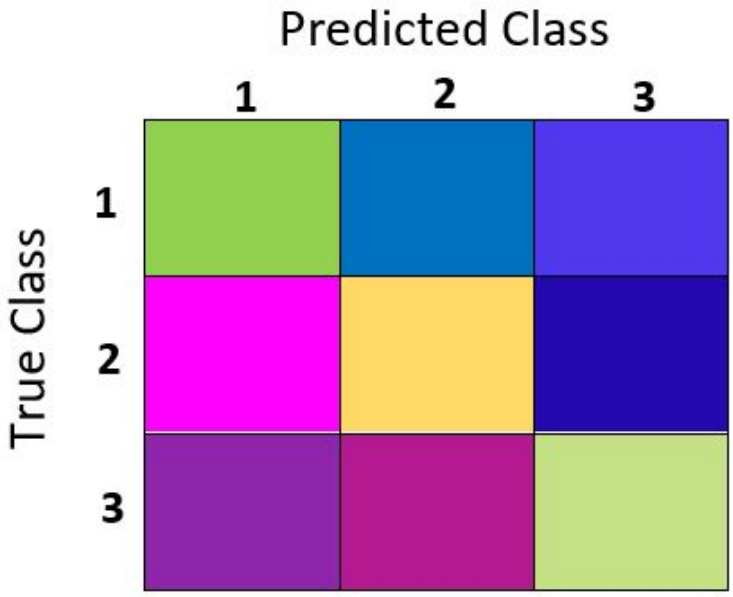


Figure 3.7: Example Confusion Matrix

in equations 3.4.2 and 3.4.3 (Developers, 2020).

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3.4.2}$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.4.3}$$

Precision is mostly important when having a false positive could be costly, whereas recall is mostly important when having a false negative could be costly. However, both are important to understanding the overall effectiveness of the model, and therefore a metric that evaluates both is needed. The f1-score is a measure of both precision and accuracy, providing a metric that balances both. F1-score is calculated with equation

3.4.4, and was found using the "classification_accuracy" function in python.

$$F1 = \frac{Precision * Recall}{Precision + Recall} \tag{3.4.4}$$

Precision, recall and F1 score were calculated for each class, as this gives a better understanding on how the model works as opposed to an overall accuracy that averages all of the classes together. An overall F1 score and accuracy were also calculated for the entire classification, however the values are not as telling as they are an average across all classes.

Receiver operator characteristic curves (ROC) were used to show the performance of the models, and the area under the curve (AUC) was calculated to summarize the results. ROC is regularly used to summarize the sensitivity and specificity over varying thresholds(Hastie *et al.*, 2009). Sensitivity is defined as the true positive rate, and measures the true predictions that are actually true. Specificity is defined as the true negative rate, and measures the negative predictions that are actually negative. Sensitivity (recall) and specificity are calculated using Equations 3.4.3 and 3.4.5 (Swift *et al.*, 2019). The ROC is plotted as sensitivity in the y-axis and 1 - specificity in the x-axis, where 1 - specificity represents the false positive rate.

$$specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} \tag{3.4.5}$$

Thresholds are values used to determine at what probability a value will be predicted in each class. For example, if the threshold is 70%, then the probability will need to exceed 70% for the point to be classified as that class(Hand and Anagnostopoulos, 2013). The ROCs and AUC are calculated in python using the functions

"roc_curve" and "auc". These functions were inside a "for-loop" to find the ROC and AUC for each predicted class. The code for this process can be found in Appendix A.2.4 and process can be seen in Figure 3.8. ROC and AUC are plotted for each class, and are shown together on one plot. The perfect ROC would have a be a square like curve in the top left corner of the graph, showing a high true positive rate with a low false positive rate. Code to plot the multi-class ROC curves was taken from a tutorial done by Analytics Vidhya(Bhandari, 2020). Conclusions were made about model performance based on ROC curves, classification result summaries, and the confusion matrices together as opposed to one individual metric.

All methods described above were used for all trials of prediction, where different combinations of stage predictions were attempted to find the best model performance. Chapter 4 describes the first attempt, where all RTA and RTS stages were predicted. Next, feature reduction was attempted to determine whether eliminating highly correlated features could improve model performance. This is explained in Chapter 5. Since feature reduction did not greatly improve performance, stage 1 was removed and models were attempted again, which is described in chapter 6. Finally, in a final attempt to find the best models, stage 1 was removed and the last two stages of each protocol were combined. This is described in Chapter 7. Some supplementary steps were taken depending on the attempt, and any additional methods will be described in the corresponding chapter below.

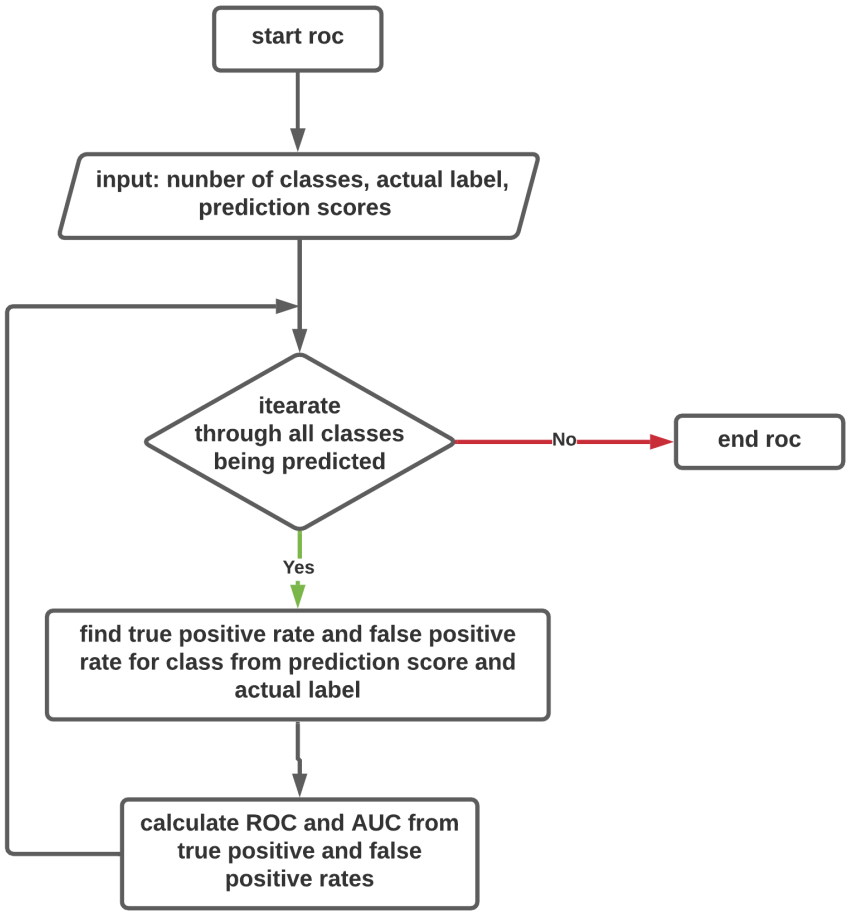


Figure 3.8: Flowchart showing process for calculating ROC and AUC values.

Chapter 4

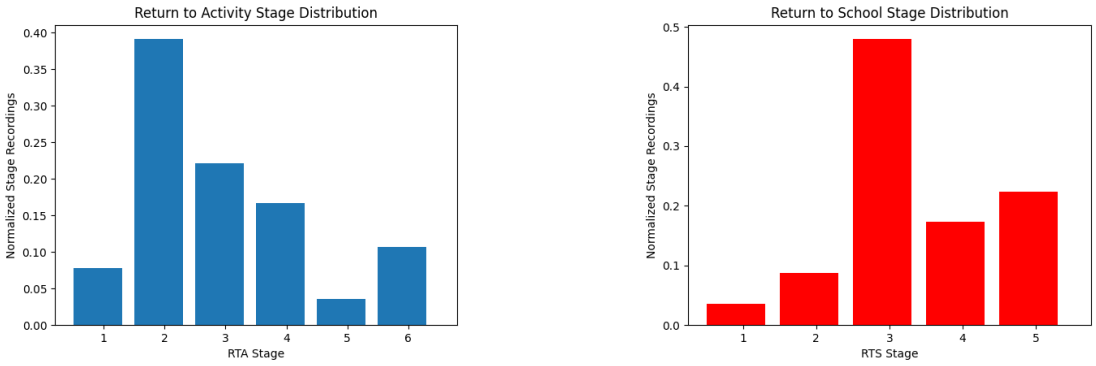
Predicting All Stages

The following chapter describes the results for the RF and SVC models predicting RTA stages 1 through 6, and RTS stages 1 through 5. This was done as the first attempt to evaluate algorithm performance to see if predicting all stages of return was possible with the extracted feature set.

4.1 Pre Processing

When predicting all stages, no additional steps were taken from the methodology explained in Chapter 3. However, one modification in methodology in this section was the hyperparameter tuning for the RF algorithm. In this attempt, the range of number of trees used i the randomized grid search was from 200 to 2000, instead of the range specified in Table 3.4. This range was changed for future iterations to encompass a larger variety of tree sizes that could potentially produce good results. All 6 RTA stages and 5 RTS stages were predicted. The distribution of the data for both RTA and RTS can be found in Figures 4.1a and 4.1b. Distributions have been

plotted as normalized values to show the proportion of data out of 1 in each stage.



(a) (b)

Figure 4.1: Return to activity (a) and return to school (b) normalized distributions for all stages

Classes were not evenly distributed, however it was decided to use the data without any additional processing to analyze model prediction on the initial dataset. The 'class_weight' parameter was set to "balanced" for both RF and SVC models to account for the imbalance in the classes.

4.2 Return to School

The best set of hyper parameters found for the RF and SVC classifiers for this set of data can be seen in Tables 4.1 and 4.2.

Parameter	**Value**
n_estimators	150
max_features	'sqrt'
max_depth	'None'
min_samples_split	2
min_samples_leaf	3
bootstrap	True
class_weight	'balanced'

Table 4.1: Random Forest Hyperparameters Tuned for RTS All Stages

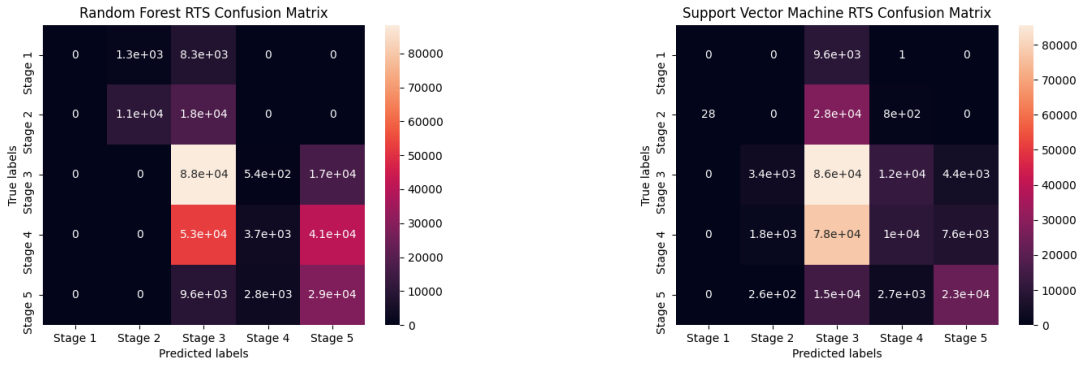
Parameter	**Value**
C	1
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 4.2: Support Vector Machine Hyperparameters Tuned for RTS All Stages

The classification results for both RF and SVC classifiers can be seen in the confusion matrices (Figure 4.2), ROC cruves 4.3, and summary table of results (Table 4.3) The overall accuracy was 0.47 for the RF model and 0.42 for the SVC model.

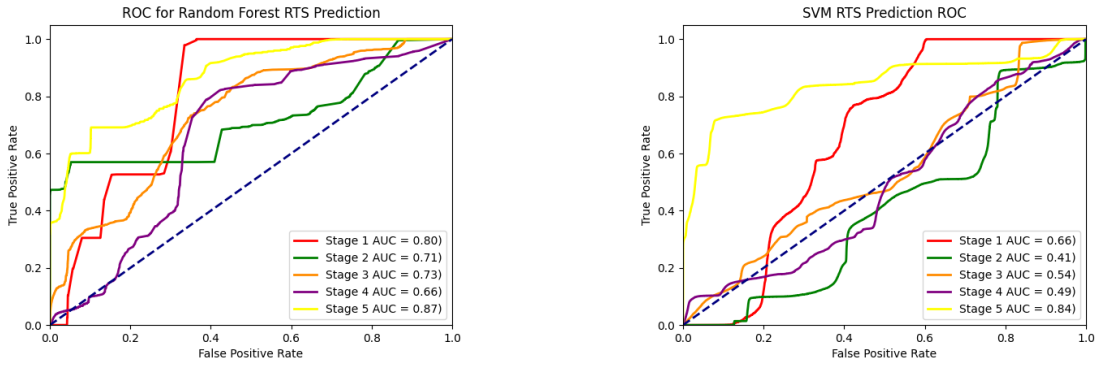
Class	Model	Precision	Recall	f1-Score
Stage 1	RF	0.00	0.00	0.00
	SVC	0.00	0.00	0.00
Stage 2	RF	0.89	0.37	0.53
	SVC	0.00	0.00	0.00
Stage 3	RF	0.50	0.84	0.63
	SVC	0.40	0.81	0.53
Stage 4	RF	0.53	0.04	0.07
	SVC	0.39	0.10	0.16
Stage 5	RF	0.33	0.70	0.45
	SVC	0.66	0.56	0.61

Table 4.3: Summary of results for RTS predictions of all stages



(a) (b)

Figure 4.2: Random forest (a) and support vector machine (b) confusion matrices for RTS classifier prediction for all stages.



(a) (b)

Figure 4.3: Random forest (a) and support vector machine (b) ROC curves for RTS classifier prediction for all stages.

Based on the results shown above, it can be seen that this model did not predict the stages very well since it had a poor overall accuracy, and many misclassified data points between all stages. Stage 1 especially had zero true positives for both models and therefore made no correct classifications. One possible explanation for this could be the lack of data for this stage, reducing the number of entries the model has to learn from making it harder for the model to train for predicting this stage. Stages 2, 3, and 5 predicted better than others for the RF model but the results were still not convincing, showing low predictive ability. The SVC had no true predictions for stage 2, and similar trends to the RF model for all other stages. The ROC curves for both models showed that in order to have high true positive rates there would also be a high false positive rate, and therefore the model would likely misclassify more data points. For example with the random forest model, for all stages to have true positive rates above 0.8, the false positive rate would be between approximately 0.4 and 0.8 for the stages. This shows that although there could be true positive classifications, there would also be a great amount of false positive predictions, which could tell the

user it is safe to move to the next stage when it is not. The ROC curves also showed that the RF model had better predictive ability with higher AUC values and curves closer to the top left corner of the plot.

Overall, when looking at all results together, its clear that these models did not perform well enough to solve the problem of predicting stages of recovery for children in real world applications. The RF classifier performed slightly better overall than the SVC, which was especially evident when looking at the ROC curves and AUC values.

4.3 Return to Activity

The best set of hyper parameters found for this set of data can be seen in Tables 4.4 and 4.5.

Parameter	**Value**
n_estimators	1600
max_features	'sqrt'
max_depth	90
min_samples_split	10
min_samples_leaf	1
bootstrap	True
class_weight	'balanced'

Table 4.4: Random forest hyper-parameters tuned for predicting RTA all stages

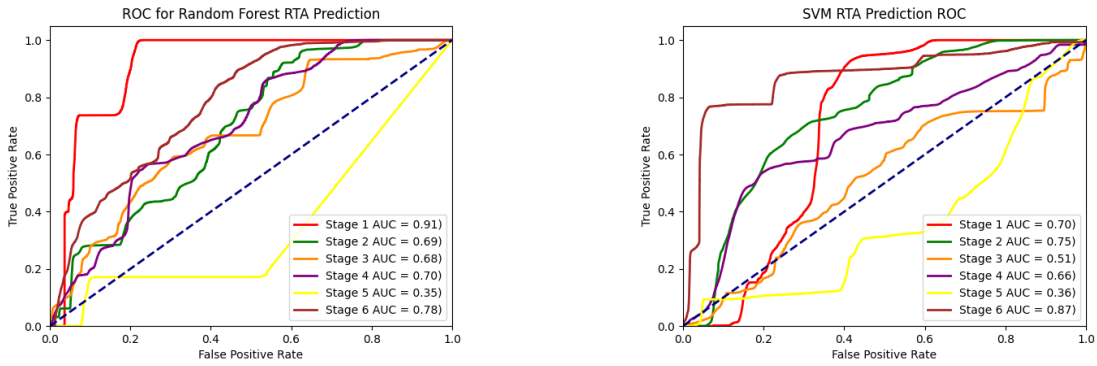
Parameter	**Value**
C	1000
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 4.5: Support vector machine hyper-parameters tuned for predicting RTA all stages

The classification results for both models can be seen in the confusion matrices (Figure 4.5), ROC curves (Figure 4.4) and summary table of results (Table 4.6) The overall accuracy was 0.46 for the RF model and 0.44 for the SVC.

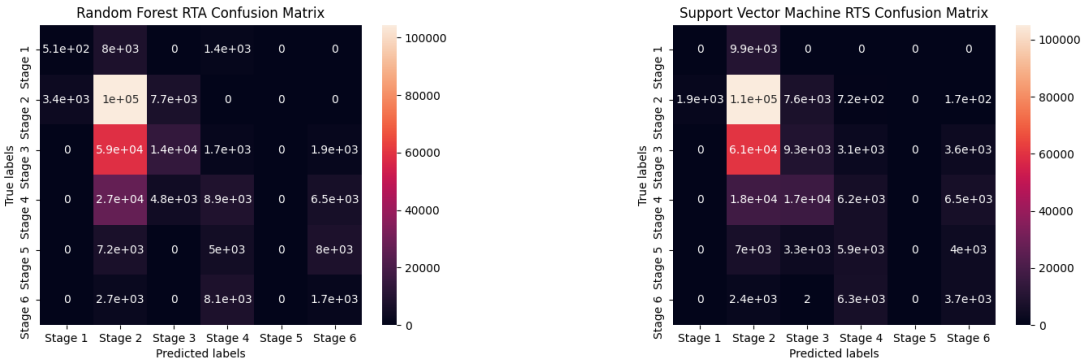
Class	Model	Precision	Recall	f1-Score
Stage 1	RF	0.13	0.05	0.07
	SVC	0.00	0.00	0.00
Stage 2	RF	0.50	0.90	0.64
	SVC	0.52	0.91	0.66
Stage 3	RF	0.53	0.18	0.27
	SVC	0.25	0.12	0.16
Stage 4	RF	0.36	0.19	0.25
	SVC	0.28	0.13	0.18
Stage 5	RF	0.00	0.00	0.00
	SVC	0.00	0.00	0.00
Stage 6	RF	0.09	0.14	0.11
	SVC	0.21	0.30	0.24

Table 4.6: Results summary for RTA prediction for all stages



(a) (b)

Figure 4.4: Random forest (a) and support vector machine (b) ROC curves for RTA classifier prediction for all stages.



(a) (b)

Figure 4.5: Random forest (a) and support vector machine (b) confusion matrices for RTS classifier prediction for all stages.

Results for predicting RTA showed stages 1, 5 and 6 having very low f1-scores with very few true positive predictions. Stage 5 had no true positive predictions. Stages 3 and 4 predicted better than 1, 2 and 5 however their f1-scores were still quite low and therefore did not classify well. Stage 2 had adequate classification with an f1-score of 64% for RF and 66% for SVC. Stage 3 and 4 had better classification for the RF compared to the SVC, but was low for both models. ROC curves for both models showed, to get higher true positive rates for all classes, the false positive rate would also be quite high. Stage 1 in the RF model could have improved prediction with a different classification threshold, however this would maybe not benefit other stages and could lead to more misclassification of those stages. AUC values between RF and SVC were quite different some stages, such as Stage 1, however the trends were similar, and overall the models performed similarly.

When comparing the SVC and RF models, both performed at approximately the same overall accuracy with just 0.02% difference. Both showed promise for Stage 2 but not for any other stage. At this point, no model can be determined better suited

for this application. It was determined that further pre-processing should be done in hopes of improving overall model performance for predicting stages of recovery.

Chapter 5

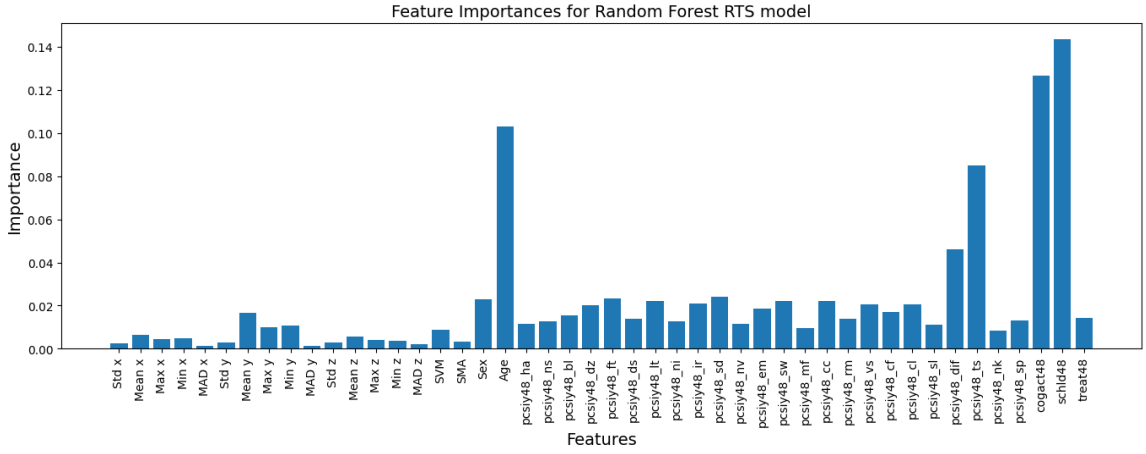
Predicting All Stages with Feature Selection

Feature selection is a pre-processing method in machine learning that selects a subset of features from an initial feature set, removing irrelevant or redundant information (Cai *et al.*, 2018). By reducing the feature set size and removing unnecessary information, model accuracy can be improved, and learning time can be decreased (Cai *et al.*, 2018). Due to results found in Chapter 4, feature reduction was attempted to determine if removing highly correlated features could help improve algorithm performance.

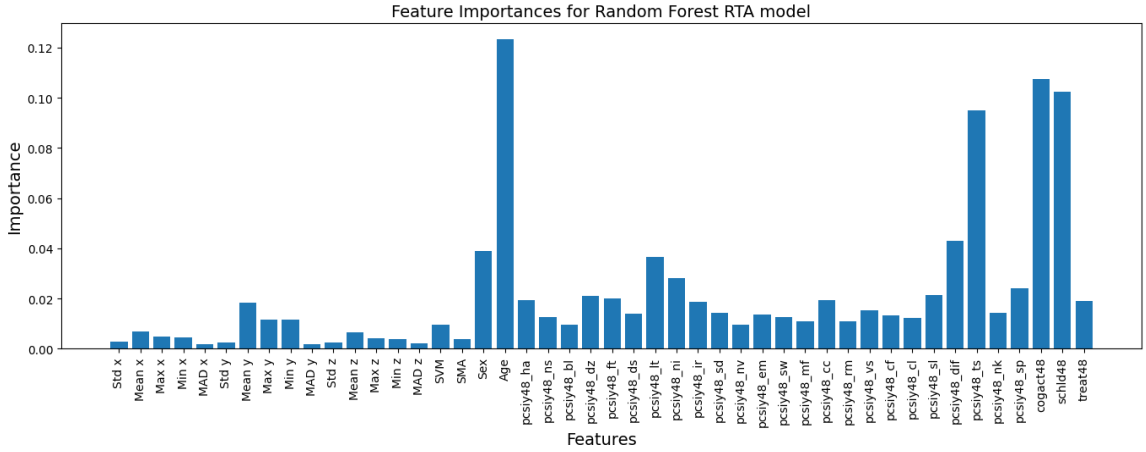
5.1 Feature Selection

The first attempted method at reducing the feature set was using a built in random forest function called "feature_importances_", which calculates the importance of each feature when fitting the model with the training set. Using the matplotlib library in

python, feature importance was plotted as a bar chart for both the RTS and RTA random forest models (Figure 5.1).



(a)



(b)

Figure 5.1: Return to School (a) and Return to Activity (b) feature importances.

Based on the importance plotted above, it is clear that a few features had much higher importances, and a ranking of top 5 features based on importance can be seen in Table .

Ranking	**Feature**
1	Score of how the school days in the past 48 hours were
2	Level of cognitive activity done in the past 48 hours
3	Age
4	Total symptom score
5	Degree of feeling differently in the past 48 hours

Table 5.1: Top 5 features in order of importance for Return to School Classification

Ranking	**Feature**
1	Age
2	Level of cognitive activity done in the past 48 hours
3	Score of how the school days in the past 48 hours were
4	Total symptom score
5	Degree of feeling differently in the past 48 hours

Table 5.2: Top 5 features in order of importance for Return to Activity Classification

All other features for both models had importance below 0.04. Although some were smaller than others, it was decided that there were no obvious features to remove

using this method. Both RTA and RTS models had the same trends for all features, with slight variance in the actual value of importance. The SVC did not have a similar built in function and therefore this was only analyzed on the RF model. It is speculated that importance would be similar for SVC models.

The second method used to reduce the feature set was correlation between features. Features that are highly correlated to eachother would provide the same information to the model, and therefore having both features in the set adds redundant information (Yu and Liu, 2003). The python function "corr()" was used to calculate the Pearson correlation coefficient between each feature. A correlation matrix for the feature set was plotted as a heat map and features with 0.8 or higher were deemed as highly correlated, and one of the features was removed. Correlation of 0.8 and higher has been recorded in medical research as displaying a very strong relationship between values (Chan, 2003)(Akoglu, 2018). The correlation matrix for the feature set can be seen in Figure B.1 in Appendix B. The features that had correlation above 0.8 with at least one other feature were removed. These features were 'Std x', 'MAD y', 'Std z', 'pcsiy48_dif', 'pcsiy48_sd', 'Mean y'. The models were run again with the new feature set not including the features mentioned. The code to find and plot the correlation can be found in Appendix A.2.5.

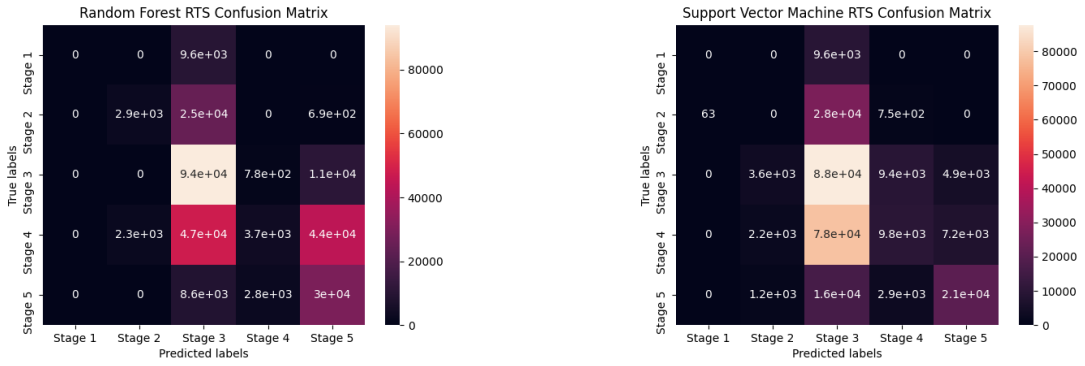
5.2 Return to School

The hyperparameters for both RF and SVC were the same from those used in Chapter 4 and can be found in Tables 4.1 and 4.2 respectively. The results can be found in the confusion matrices (Figure 5.2), ROC curves (Figure 5.3), and summary table of results (Table 5.3). Overall accuracy was 0.46 for the RF model and 0.42 for the

SVC.

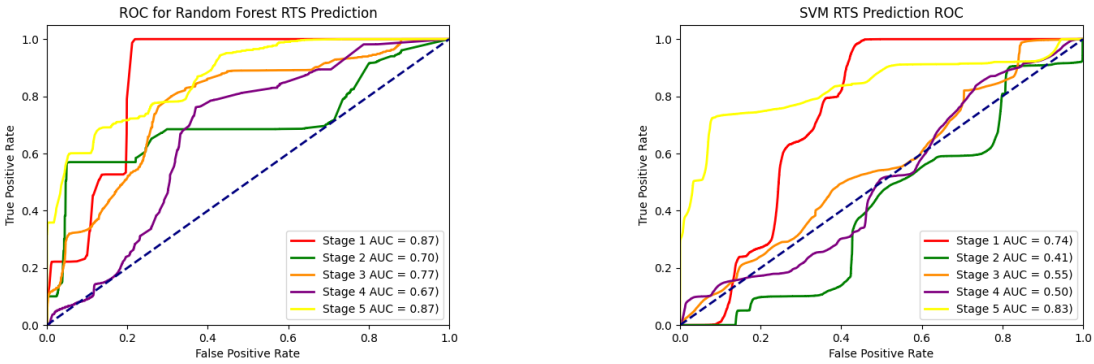
Class	Model	Precision	Recall	f1-Score
Stage 1	RF	0.00	0.00	0.00
	SVC	0.00	0.00	0.00
Stage 2	RF	0.55	0.10	0.17
	SVC	0.00	0.00	0.00
Stage 3	RF	0.51	0.89	0.65
	SVC	0.40	0.83	0.54
Stage 4	RF	0.51	0.04	0.07
	SVC	0.43	0.10	0.16
Stage 5	RF	0.35	0.72	0.47
	SVC	0.63	0.51	0.56

Table 5.3: Summary of results for RTS predictions of all stages with feature reduction



(a) (b)

Figure 5.2: Random forest (a) and support vector machine (b) confusion matrices for RTS classifier prediction with feature reduction.



(a) (b)

Figure 5.3: Random forest (a) and support vector machine (b) ROC curves for RTS classifier prediction with feature reduction.

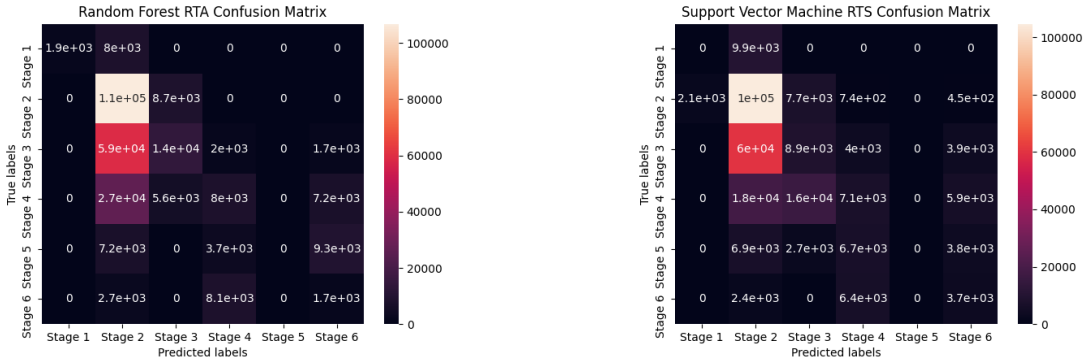
RTS predictions were very similar to the results seen in Chapter 4. Stages 3, 4, and 5 predicted the same or slightly better with feature reduction for the RF mode, however stage 2 had much worse classification. No major differences were seen in the ROC curves for both models. Overall accuracy for RF was comparable, at just 0.01% lower than previously. For the SVC classifier, all stages prediction the same with stage 5 having a slightly worse classification compared to the previous model. Overall accuracy was the same for SVC models with and without feature reduction.

5.3 Return to Activity

The hyperparameters for both RF and SVC were the same from those used in Chapter 4 and can be found in Tables 4.4 and 4.5 respectively. The results can be found in the confusion matrices (Figure 5.4), ROC curves (Figure 5.4), and summary table of results (Table 5.4). Overall accuracy was 0.47 for the RF model and 0.44 for the SVC.

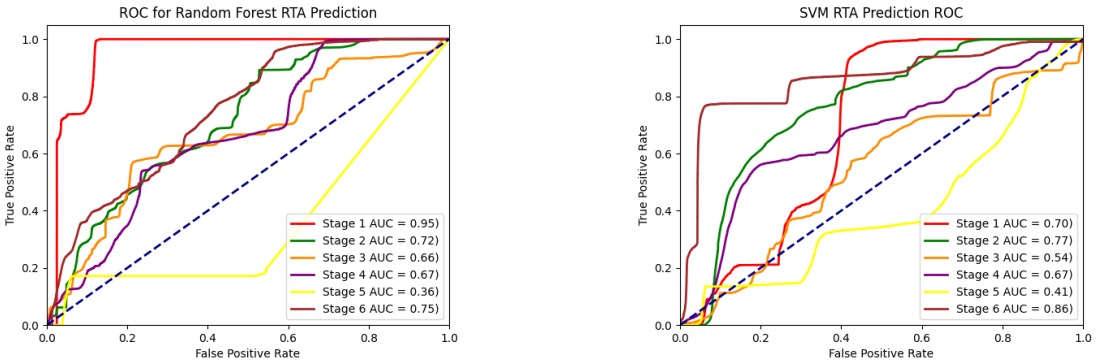
Class	Model	Precision	Recall	f1-Score
Stage 1	RF	1.00	0.19	0.32
	SVC	0.00	0.00	0.00
Stage 2	RF	0.51	0.91	0.66
	SVC	0.52	0.91	0.66
Stage 3	RF	0.49	0.18	0.27
	SVC	0.25	0.12	0.16
Stage 4	RF	0.37	0.17	0.23
	SVC	0.28	0.15	0.20
Stage 5	RF	0.00	0.00	0.00
	SVC	0.00	0.00	0.00
Stage 6	RF	0.08	0.14	0.10
	SVC	0.21	0.29	0.24

Table 5.4: Results summary for RTA prediction for all stages with feature reduction.



(a) (b)

Figure 5.4: Random forest (a) and support vector machine (b) confusion matrices for RTA classifier prediction with feature reduction.



(a) (b)

Figure 5.5: Random forest (a) and support vector machine (b) ROC curves for RTA classifier prediction with feature reduction.

Simialrly to results seen for RTS prediciton, feature reduction did not greatly improve model performance of the RTA classification. The RF model had better prediction for stage 1 and 2, however all other stages were either the same or slightly lower. The SVC model had slightly better performance at stage 4, but all other stages were the exact same. No major differences were seen in the ROC curves for both models. There were slight improvements for the models with feature reduction, but not great enough for feature reduction to have a large effect. Therefore, the full feature set was kept and further pre-processing methods were tested to improve algorithm performance.

Chapter 6

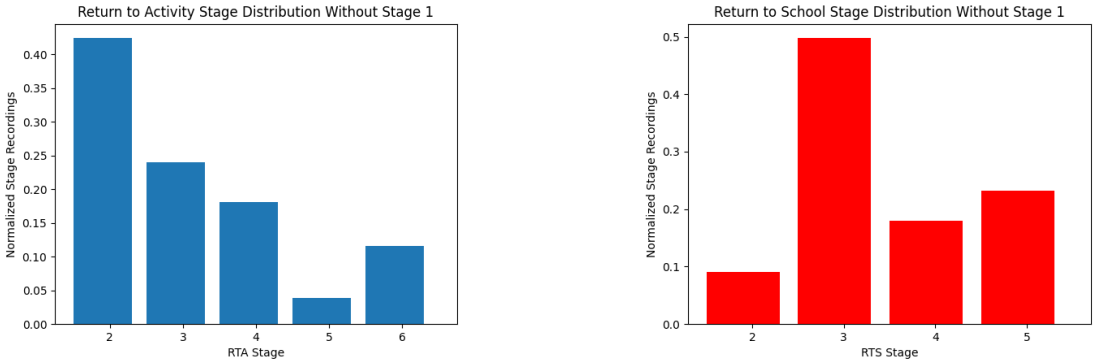
Predicting Stages while Removing Stage 1

The following chapter describes the results for the RF and SVC models predicting RTA stages 2 through 6, and RTS stages 2 through 5. Stage 1 was removed because it had a much smaller amount of data, making it difficult for models to predict. Furthermore, stage 1 of recovery occurs normally within the first 24-48 hours after injury. Therefore data collection, especially for a research study, is more challenging during this period (i.e typically a smaller window of opportunity for recruitment). Thus, removing stage 1 was the next test to find the best possible model for predicting RTA/RTS stages.

6.1 Pre Processing

Methodology from Chapter 3 was used with an additional step to remove the data from stage 1. Two data sets were created, one for RTA with the RTS labels removed,

and one for RTS with the RTA labels removed. For each data set, any row of features that was labelled as stage one was removed. This data set contained features for 62 participants with a total of 778915 rows for RTA and 813761 rows for RTS, where each row represents one labelled entry. The distribution of the data for both RTA and RTS can be found in Figure 6.1.



(a) (b)

Figure 6.1: Return to activity (a) and return to school (b) normalized stage distributions without stage 1.

Similarly to the distributions seen in Chapter 4, there is not an even distribution between stages even when removing stage 1.

6.2 Return to School

The best set of hyper parameters found for this set of data can be seen in Tables 6.1 and 6.2.

Parameter	**Value**
n_estimators	550
max_features	'sqrt'
max_depth	60
min_samples_split	5
min_samples_leaf	2
bootstrap	'False'
class_weight	'balanced'

Table 6.1: Random forest hyperparameters tuned for RTS removing stage 1

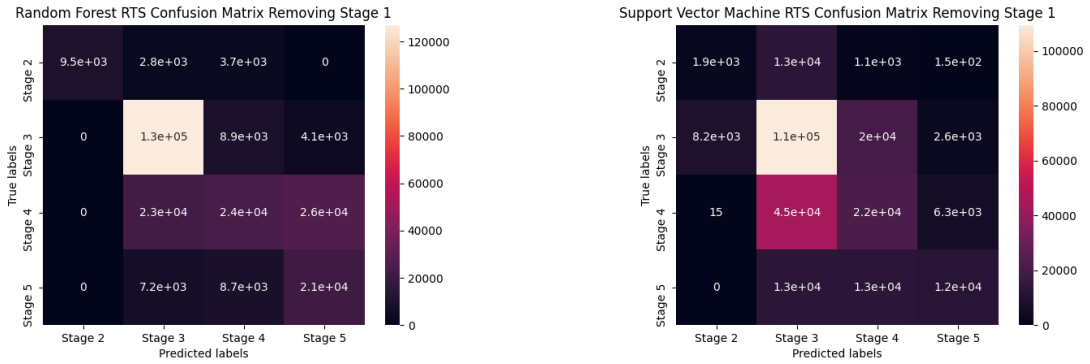
Parameter	**Value**
C	100
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 6.2: Support vector machine hyperparameters tuned for RTS without stage 1

The classification results for both RF and SVC can be seen in the confusion matrics (Figure 6.2), ROC curves (Figure 6.3), and summary table of results (Table 6.3) The overall accuracy was 0.68 for the RF model and 0.54 for the SVC.

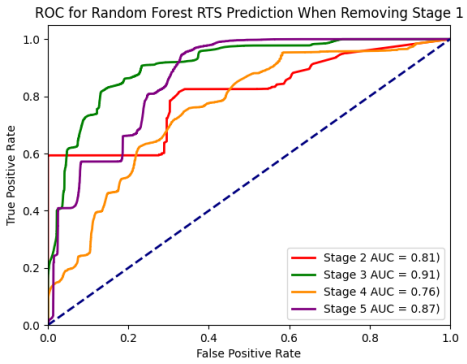
Class	Model	Precision	Recall	f1-Score
Stage 2	RF	1.00	0.59	0.74
	SVC	0.19	0.12	0.15
Stage 3	RF	0.80	0.91	0.85
	SVC	0.61	0.78	0.68
Stage 4	RF	0.53	0.33	0.41
	SVC	0.39	0.30	0.34
Stage 5	RF	0.41	0.57	0.48
	SVC	0.57	0.32	0.41

Table 6.3: Results summary for RTS predictions while removing stage 1

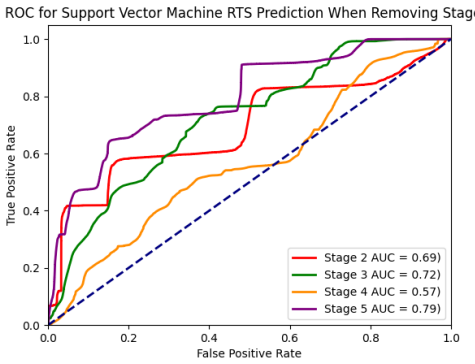


(a) (b)

Figure 6.2: Random forest (a) and support vector machine (b) confusion matrices for RTS predictions without stage 1.



(a)



(b)

Figure 6.3: Random forest (a) and support vector machine (b) ROC predicting RTS stages without stage 1.

Overall, the prediction for RTS while removing stage 1 was much better than the results seen in Chapter 4. Stage 2 predicted well for both the RF and SVC models, and 3 had good prediction for the RF model. Stage 4 and 5 still had relatively low results, with f1-scores under 50% for both models. Over half of the misclassifications for both stages 4 and 5 were with each other, likely due to the similarities between stage 4 and 5 of the RTS protocol. The ROC curve showed high area under the curves for all RF predicted stages, with stage 3 having the best score. The RF values were higher for all stages than the SVC model, showing better predictive ability for the RF model. Overall, to have a high true positive rate for all classes, the false positive rate would also need to be high meaning more mislcassifications could occur.

The RTS predictions were much better when stage 1 was removed from the data set. The RF model worked better than the SVC classifier with the overall accuracy being 14% higher, however there is still a need to improve algorithm performance prior to APP implementation.

6.3 Return to Activity

The best set of hyper parameters found for this set of data can be seen in Tables 6.4 and 6.5.

Parameter	**Value**
n_estimators	1100
max_features	'sqrt'
max_depth	10
min_samples_split	2
min_samples_leaf	1
bootstrap	'True'
class_weight	'balanced'

Table 6.4: Random forest hyperparameters tuned for RTA without stage 1

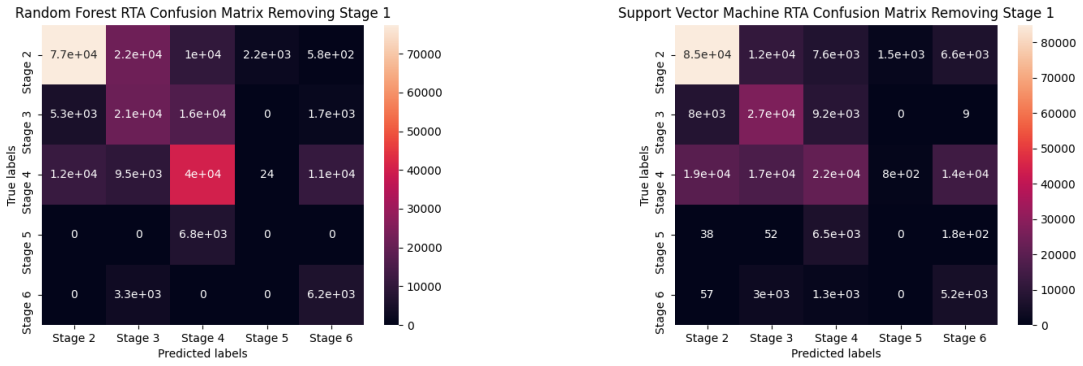
Parameter	**Value**
C	1000
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 6.5: Support vector machine hyperparameters tuned for RTA without stage 1

The classification results can be seen in the confusion matrices (Figure 6.4), ROC curves 6.5), and summary table of results (Table 6.6) The overall accuracy was 0.59 for the RF model and 0.57 for the SVC.

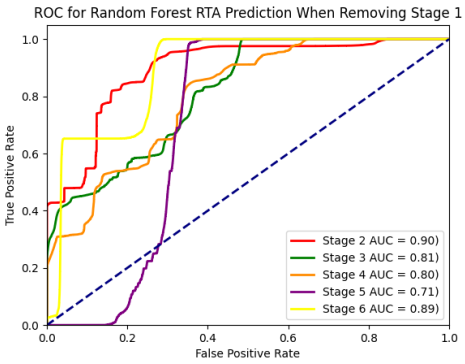
Class	Model	Precision	Recall	f1-Score
Stage 2	RF	0.82	0.69	0.75
	SVC	0.76	0.76	0.76
Stage 3	RF	0.38	0.48	0.42
	SVC	0.47	0.61	0.53
Stage 4	RF	0.55	0.56	0.55
	SVC	0.47	0.30	0.37
Stage 5	RF	0.00	0.00	0.00
	SVC	0.00	0.00	0.00
Stage 6	RF	0.32	0.65	0.43
	SVC	0.20	0.54	0.29

Table 6.6: Results summary for RTA predictions without stage 1

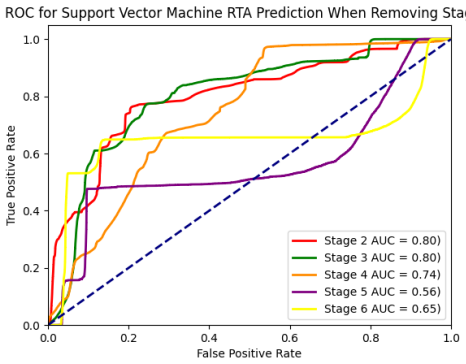


(a) (b)

Figure 6.4: Random forest (a) and support vector machine (b) confusion matrices for RTA predictions without stage 1.



(a)



(b)

Figure 6.5: Random forest (a) and support vector machine (b) ROC predicting RTA stages without stage 1.

RTA predicitons showed worse performance than RTS, but still a higher performance than results shown in Chapter 4. Stage 5 had no correct predictions for obth models, but all other stages had some true predictions with stage 2 showing the best classification results. Aside from stage 5, AUC values were high for the RF model showing promising predictive abilities of the algorithm. Stage 2, 3 and 4 for the SVC classifier also had high AUC values, but stages 5 and 6 had lower values than the RF model, and with values closer to 0.5 showed it had difficulty predicting these stages. For both models, to get high true positive rates for all stages there would be a high false positive rate as well.

Overall, the results outlined in this chapter exceed those in Chapter 4, showing that algorithm performance improved by removing stage 1 from the data sets. Algorithm performance for all models was still not high enough for app implementation, and further modifications were made in attempt to improve performance.

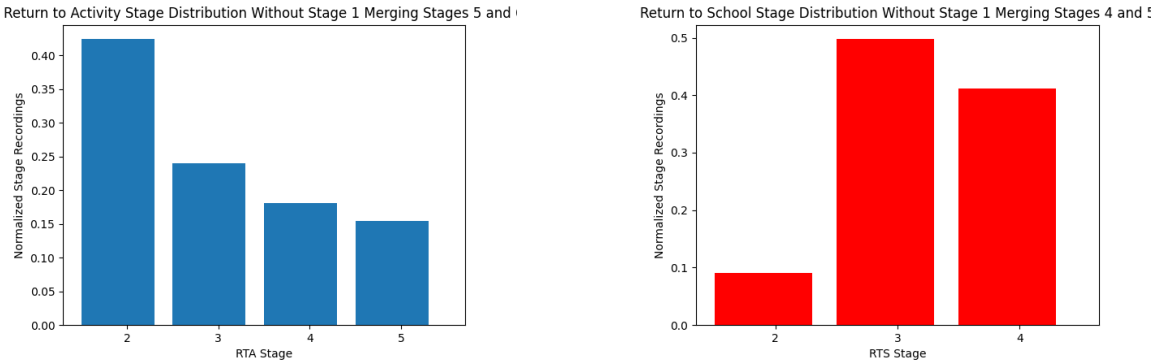
Chapter 7

Predicting Stages with Stage Combination

The following chapter describes the results for the RF and SVC models predicting RTA stages 2 through 5, where 5 is a combination of stages 5 and 6. Additionally, RTS stages 2 through 4 were modified such that 4 was a combination of stages 4 and 5. The last 2 stages from each return protocol were combined because, based on results presented in earlier chapters, these stages had many misclassifications between them. These classification errors were undoubtedly due to stage similarities and the fuzziness between them, especially when relying on self-reported measures. Combining the stages increased the number of data points for the class, improving balance in the data set, and reducing the number of classes that the model needed to predict. This was the final attempt at improving algorithm performance, yielding the best results across all trials.

7.1 Pre Processing

All methods from Chapter 3 were used, and the methodology from Chapter 6 used to remove stage 1 from the data set. Stage combination was done using the python function 'replace' which replaces a specified value from a column with another specified value. This data set contained features for 62 participants with a total of 778915 rows for RTA and 813761 rows for RTS, where each row represents one labelled entry. The distribution for these data sets can be seen in Figure 7.1.



(a) (b)

Figure 7.1: Return to activity (a) and return to school (b) normalized stage distributions without stage 1 and with combining the last 2 stages for each case.

While combining the stages improves the balance between the last two stages and stage 3, there is still a large gap between stage 2 in both cases. In RTA, stage 2 has much more data than the later stages, whereas in RTS stage 2 has much less data. Although stage distribution is better than the previous cases, the'class_balanced' parameter was still used for all models to account for remaining imbalances.

7.2 Return to School

The best set of hyper parameters found for this set of data can be seen in Tables 7.1 and 7.2.

Parameter	**Value**
n_estimators	700
max_features	'sqrt'
max_depth	'None'
min_samples_split	2
min_samples_leaf	1
bootstrap	'True'
class_weight	'balanced'

Table 7.1: Random forest hyperparameters tuned for RTS combining stages 4 and 5

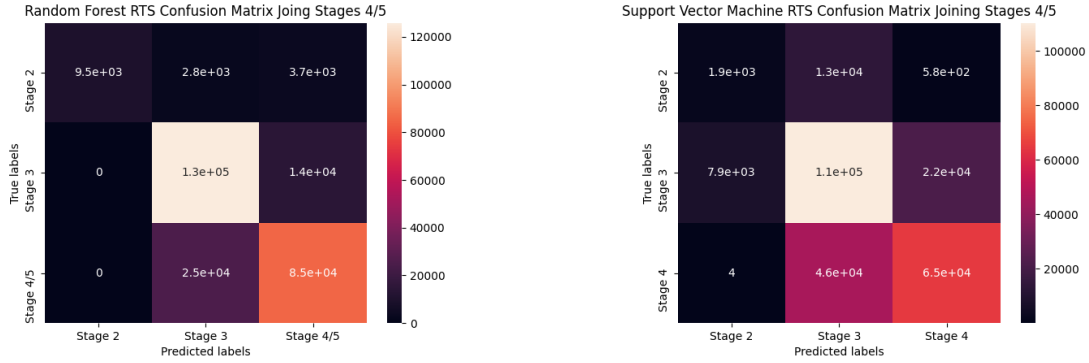
Parameter	**Value**
C	1000
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 7.2: Support vector machine hyperparameters tuned for RTS while combining stages 4 and 5

The classification results for both models can be seen in the confusion matrices (Figure 7.2), ROC curves (Figure 7.3), and summary table of results (Table 7.3). The overall accuracy was 0.83 for the RF model and 0.66 for the SVC.

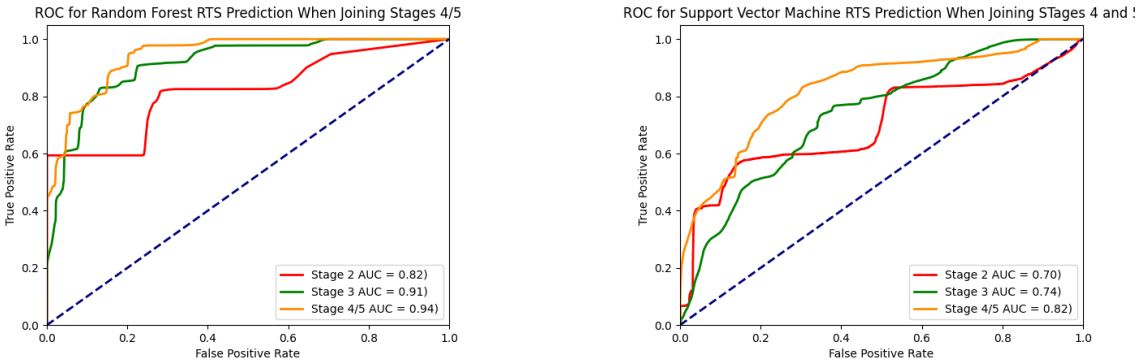
Class	Model	Precision	Recall	f1-Score
Stage 2	RF	1.00	0.59	0.74
	SVC	0.19	0.12	0.15
Stage 3	RF	0.82	0.90	0.86
	SVC	0.65	0.79	0.71
Stage 4/5	RF	0.83	0.77	0.80
	SVC	0.74	0.59	0.66

Table 7.3: Results summary for RTS predictions while combining stages 4 and 5



(a) (b)

Figure 7.2: Random forest (a) and support vector machine (b) confusion matrices for predicting RTS stages while combining stages 4 and 5



(a) (b)

Figure 7.3: Random forest (a) and support vector machine (b) ROC curves for predicting RTS stages while combining stages 4 and 5.

The RF classifier to predict RTS was the best performing model. All stages predicted with an f1-score above 0.70 showing good predictive abilities. Stage 3 had slightly better performance than stages 4/5, which was anticipated since it had the largest amount of data, and is consistent with the results from previous chapters. However, the SVC model had good results for stages 3 and 4/5, but not for stage 2. The RF model predicted each stage better than the SVC. AUC values were all higher than previous models and the ROC curve showed good trade-off between true positive and false positive rates for all classes. RF AUC values were higher than the SVC showing better predictive ability for that model. Although the model had some difficulty predicting stage 2, it still showed good results and promise for predicting concussion recovery stages. Overall, RTS stages were best predicted when stage 1 was removed and stages 4 and 5 were combined.

7.3 Return to Activity

The best set of hyper parameters found for this set of data can be seen in Tables 7.4 and 7.5.

Parameter	**Value**
n_estimators	550
max_features	'sqrt'
max_depth	15
min_samples_split	10
min_samples_leaf	2
bootstrap	'True'
class_weight	'balanced'

Table 7.4: Random forest hyperparameters tuned for RTA while combining stages 5 and 6

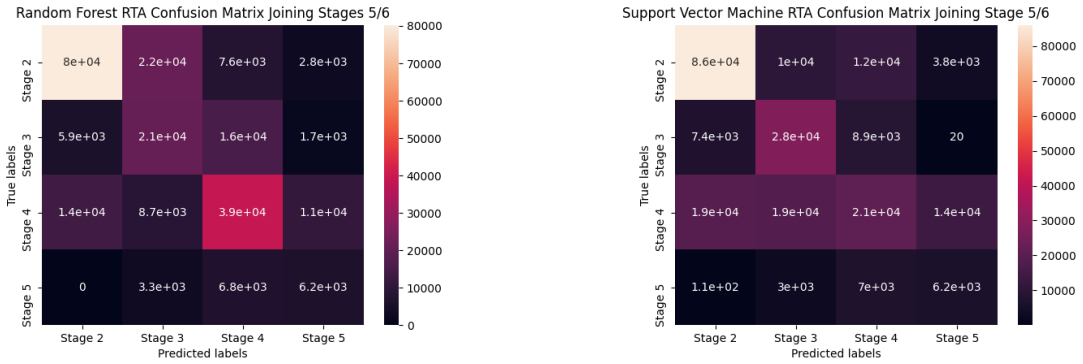
Parameter	**Value**
C	10
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 7.5: Support vector machine hyperparameters tuned for RTA while combining stage 5 and 6

The classification results can be seen in the confusion matrices (Figure 7.4), ROC curves (Figure 7.5), and summary table of results (Table 7.6) The overall accuracy was 0.60 for the RF model and 0.58 for the SVC.

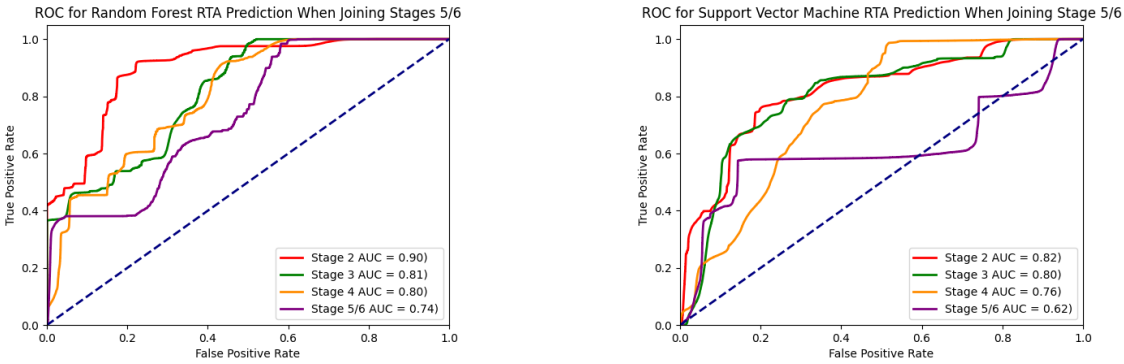
Class	Model	Precision	Recall	f1-Score
Stage 2	RF	0.80	0.72	0.76
	SVC	0.76	0.77	0.77
Stage 3	RF	0.39	0.48	0.43
	SVC	0.47	0.63	0.54
Stage 4	RF	0.57	0.54	0.55
	SVC	0.43	0.29	0.34
Stage 5/6	RF	0.29	0.38	0.33
	SVC	0.26	0.38	0.31

Table 7.6: Results summary for RTA predictions while combining stages 5 and 6



(a) (b)

Figure 7.4: Random forest (a) and support vector machine (b) confusion matrices for predicting RTA stages while combining stages 5 and 6



(a) (b)

Figure 7.5: Random forest (a) and support vector machine (b) ROC curves for predicting RTA stages while combining stages 5 and 6.

The RF and SVC classifiers to predict RTA stages while removing stage 1 and combining stages 5 and 6 showed the best results for RTA predictions. However, only stage 2 had high f1-scores at 0.76 and 0.77. All other stages had f1-scores close to or below 0.50. ROC curves had good AUC values, however for all stages to have high true positive rates they would also have high false positive rates. AUC values were higher for the RF model when compared to the SVC, showing better predictive ability for this model. Overall, this model performed best compared to all previous RTA models, however it still did not predict all stages with a high accuracy. Further work could be done to try and improve model performance before implementation into the APP for use on stage prediction for children with concussion.

Chapter 8

Stage Predictions for New Data

Data was collected using the first iteration of the Back2Play APP developed by Medic, a software development company working with CanChild. Recruited participants wore an Apple Watch which collected accelerometer data at a sampling frequency of 33Hz. Participants answered a symptom survey three times per day, and at the end of each day recorded their RTS and RTA stage. Heart rate was also recorded. However this data was not recorded for some participants. Heart rate was therefore unusable due to a lack of data. A total of 5 participants, 3 female and 2 male, were recruited. Participant ages ranged from 8 to 15 which was disappointingly low due to Covid-19. Lockdowns and cancellation of sports and schools lowered the number of children getting concussed and therefore decreased the pool of potential participants in the study.

8.1 Pre Processing

Symptom processing, accelerometer windowing, data combination, and modelling methods were the same as those described in Chapter 3, however some additional formatting and changes were required. Symptom recordings were collected as a list where one column contained each question asked in the survey, and another column had the response for each question, with all participants mixed throughout the list. Data was separated by participant by encoding each participant with a single letter identifier, and extracting every row with that identifier for each participant. Some questions were removed as they were not desired as features for the models. These questions include participant listing which cognitive and physical activities they did, and a rank from 1-5 of how they were doing that day. These symptoms were removed to keep symptom features consistent with the features used from the previous data with the hopes of having similar model results. The remaining features included all 22 symptoms from the PCSS, listed in Table 2.1. All other recordings from the surveys that weren't necessary as features were removed, such as ID's used in the database to label entries, questions, and other files.

A function was written to format the file into a table where each survey was represented by one row containing the user ID, time of completion, and symptom recordings. Firstly, rows that contain questions that weren't required for the feature set were removed using a function that checks if rows contain the specific value in a certain column using a conditional, and if the response is 'False', then the row is kept. Columns containing unnecessary information for the feature set were removed using the python function 'drop'. The 'pivot_table' function was then used to create a matrix where the values from the 'question_text' column was set as the new columns,

and answers for each question from the 'answer_text' column were set as the row values in each column. Rows were grouped by values that had the same participant ID and completion time, making one row for each survey entry. Similar to processing done on previous data, the symptom values were changed to yes/no responses. In this data collection, participants were asked to answer whether a symptom was new (recorded as yes), better, worse, the same, or gone. Any answer recorded as yes, better, worse, or same was changed to 1, representing that they were experiencing the symptom. All symptoms recorded as no or gone were changed to 0, representing that they were not experiencing the symptom. Time entries were then changed to a 'datetime' so that it could be sorted in chronological order. Entries were grouped by participant and sorted by time. Finally, a label matrix was extracted from the overall matrix, containing RTA and RTS stages, user ID, and time completed. Symptom files could be processed by using this function, where the file was the input and the outputs were the symptom matrix and the label matrix. The function process is described in Figure 8.1.

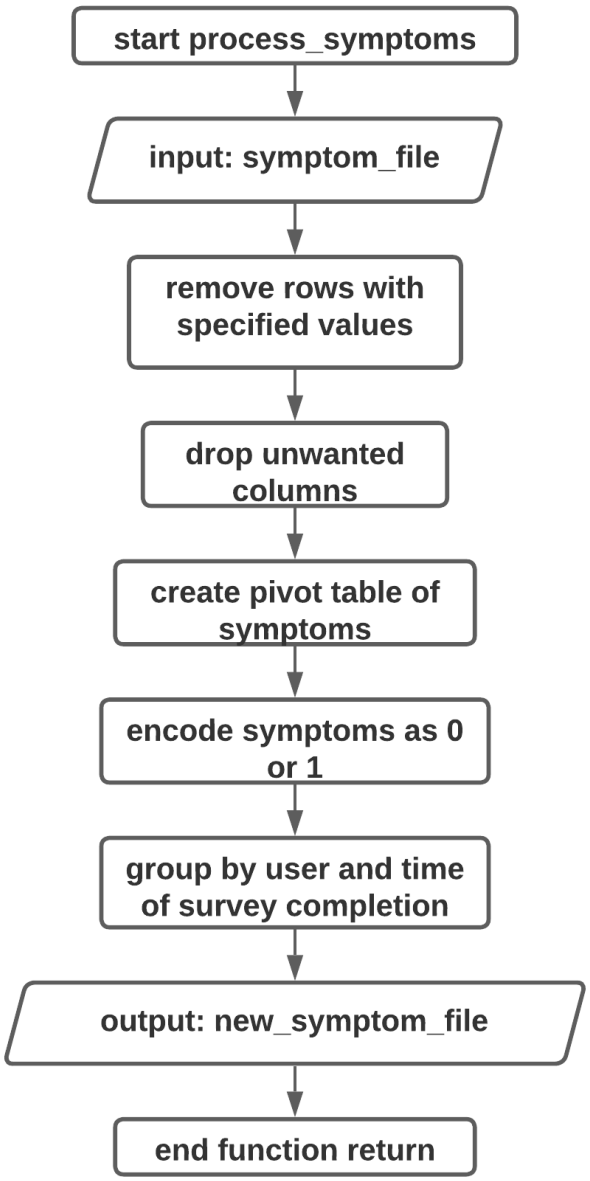
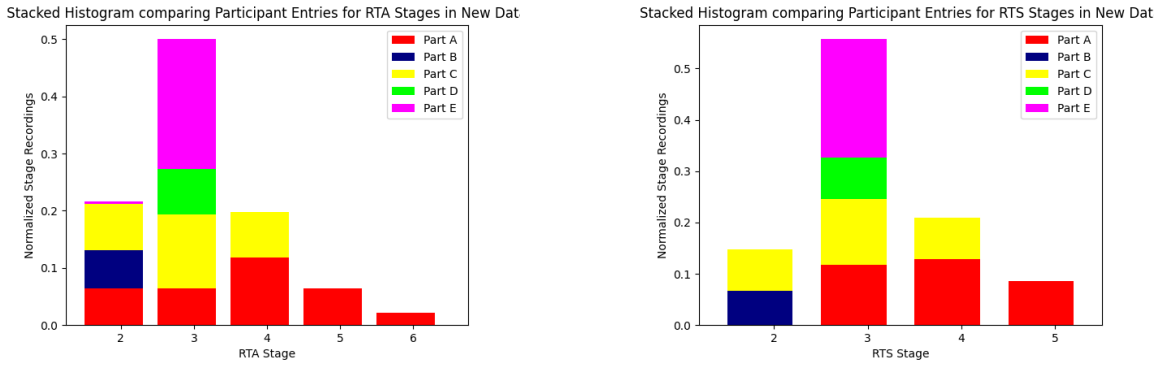


Figure 8.1: Flowchart showing process for processing new symptom data.

Accelerometer data processing was the same as detailed in Chapter 3, however there were no non-wear times to remove. The APP on the Apple Watch was set with

security to lock the watch when it was removed. If the watch was locked, accelerometer data was not collected, therefore data was not collected when the watch was not worn.

Stage 1 was removed from the dataset since there was very little data in this class, to reduce the number of classes, and attempt to get the best algorithm possible. For RTS, stage 4 and 5 were combined, and for RTA stages 4, 5, and 6 were combined. All three stages had to be combined due to lack of data in stages 5 and 6. Only one participant had data past stage 4, and therefore there was not enough data to have those stages represented in both the training and testing set. Each model was therefore predicting 3 stages. Due to a lack of data, the training and testing split was changed to 80/20 for these models to add as much training data as possible. The distribution of the data can be seen in Figure 8.2. All pre-processing code can be found in Appendix A.2.7.



(a) (b)

Figure 8.2: Return to activity (a) and return to school (b) normalized stage distributions for new data, with distinction between participants.

Distributions were split per participant to show the lack of distribution over all stages. Based on the distributions, it is clear that there was a not enough data for this test and the models were expected to have difficulty predicting the classes. Three

participants are only represented in one stage, where the other two have data split over various stages which causes a lack of variability in the training and testing sets.

Hyper-parameter tuning was the same as mentioned in Chapter 3, however the range for SVC gamma was increased. Gamma values of 100, 10, 1, 0.1, 0.01, 0.001, 0.0001, and 0.00001 were tested. The range was increased to include larger values since the dataset was much smaller, and therefore the grid search was much quicker and more values could be tested without too much additional computation time. Increasing the range of gamma was done to see if larger values of gamma could improve algorithm performance. Since gamma controls the distance required between samples for them to be similar, a higher gamma requires less distance between points for them to be similar, making the boundary more defined but in exchange taking longer to compute.

8.2 Return to School

The best set of hyper parameters found for this set of data can be seen in Tables 8.1 and 8.2.

Parameter	**Value**
n_estimators	10
max_features	'sqrt'
max_depth	90
min_samples_split	5
min_samples_leaf	4
bootstrap	'True'
class_weight	'balanced'

Table 8.1: Random forest hyperparameters tuned for RTS prediction of new data

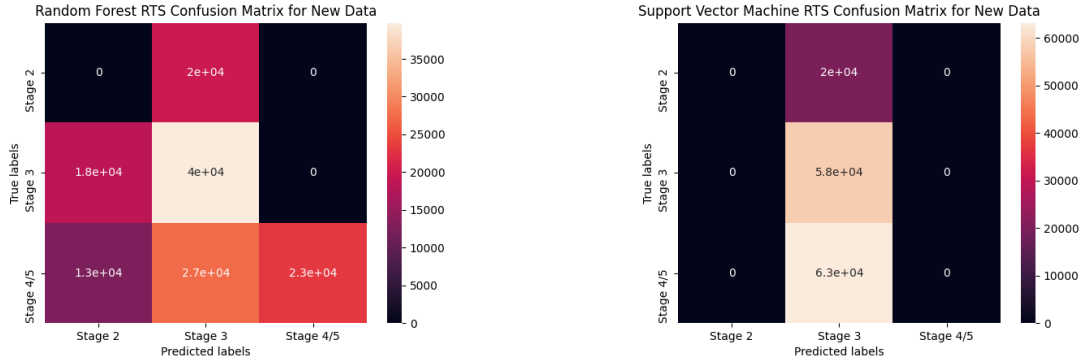
Parameter	**Value**
C	1
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 8.2: Support vector machine hyperparameters tuned for RTS prediction of new data

The classification results for both models can be seen in the confusion matrices (Figure 8.3), ROC curves (Figure 8.4), and summary table of results (Table 8.3). The overall accuracy was 0.45 for the RF model and 0.41 for the SVC.

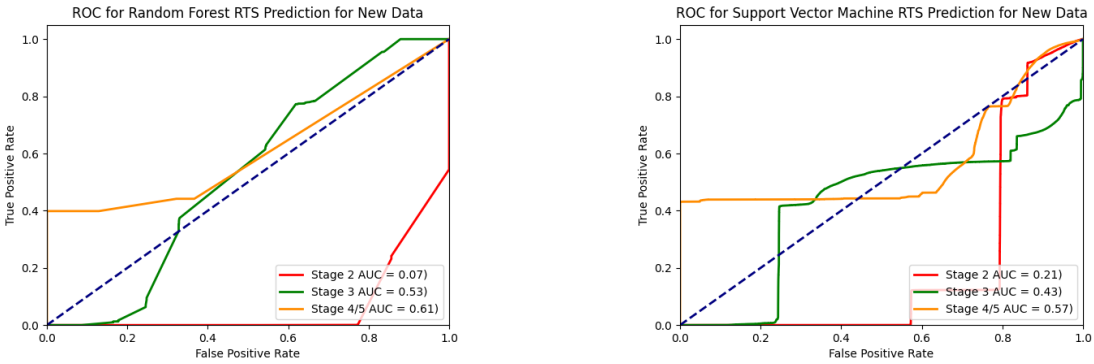
Class	Model	Precision	Recall	f1-Score
Stage 2	RF	0.00	0.00	0.00
	SVC	0.00	0.00	0.00
Stage 3	RF	0.46	0.68	0.55
	SVC	0.41	1.00	0.58
Stage 4/5	RF	1.00	0.37	0.54
	SVC	0.00	0.00	0.00

Table 8.3: Results summary for RTS predictions of the new data



(a) (b)

Figure 8.3: Random forest (a) and support vector machine (b) confusion matrices for predicting RTS stages of new data



(a) (b)

Figure 8.4: Random forest (a) and support vector machine (b) ROC curves for predicting RTS stages of new data

Both the RF and SVC models predicted stage 3 the best, with the SVC having slightly better classification than the RF with an f1-score of 0.58. The RF was able to predict some correct samples for stages 4/5 however the SVC was not. Both models were unable to predict any correct for stage 2. The ROC curves showed that both models had very low predictive ability, where all curves were around or below the 50% line. The curves also show that the RF model had better predictive ability than the SVC, which is also evident when comparing the overall accuracies of the models. Overall, neither model performed particularly well when predicting RTS stages, which was expected given the lack of data.

8.3 Return to Activity

The best set of hyper parameters found for this set of data can be seen in Tables 8.4 and 8.5.

Parameter	**Value**
n_estimators	10
max_features	'sqrt'
max_depth	90
min_samples_split	15
min_samples_leaf	4
bootstrap	'True'
class_weight	'balanced'

Table 8.4: Random forest hyperparameters tuned for RTA prediction of new data

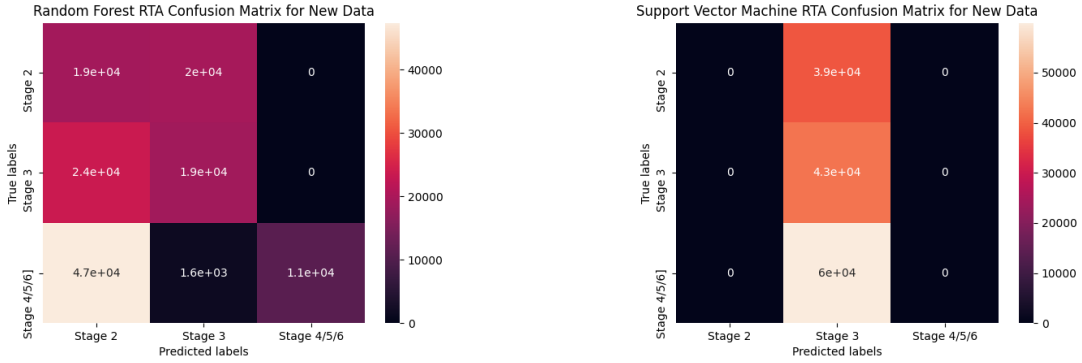
Parameter	**Value**
C	1
Gamma	0.1
Kernel	'rbf'
class_weight	'balanced'

Table 8.5: Support vector machine hyperparameters tuned for RTA prediction of new data

The classification results for both models can be seen in the confusion matrices (Figure 8.5), ROC curves (Figure 8.6), and summary table of results (Table 8.6). The overall accuracy was 0.35 for the RF model and 0.30 for the SVC.

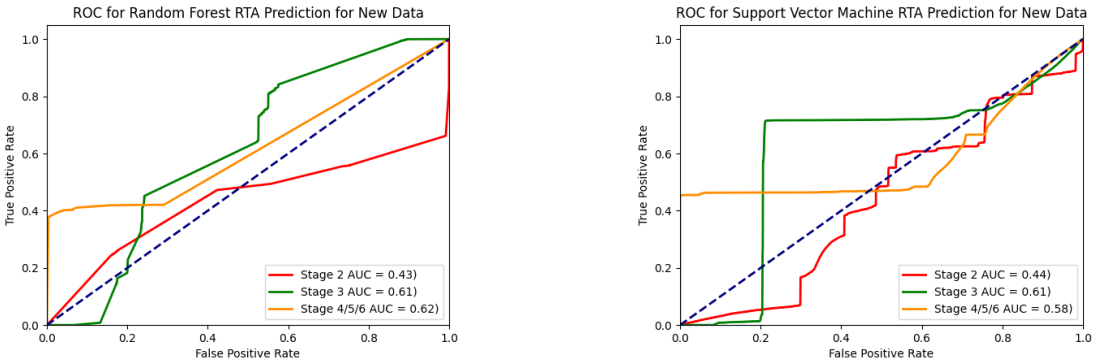
Class	Model	Precision	Recall	f1-Score
Stage 2	RF	0.21	0.49	0.30
	SVC	0.00	0.00	0.00
Stage 3	RF	0.47	0.45	0.46
	SVC	0.30	1.00	0.46
Stage 4/5/6	RF	1.00	0.18	0.31
	SVC	0.00	0.00	0.00

Table 8.6: Results summary for RTA predictions of the new data



(a) (b)

Figure 8.5: Random forest (a) and support vector machine (b) confusion matrices for predicting RTA stages of new data



(a) (b)

Figure 8.6: Random forest (a) and support vector machine (b) ROC curves for predicting RTA stages of new data

Similarly to the results above, stage 3 had the best classification for both models. The SVC was unable to correctly classify any points in the other two stages, whereas the RF model had some correct classifications with f1-score of 0.30 for stage 2 and 0.31 for stage 4/5/6. ROC curves held similar trends to those seen for RTS classification, where most stages had predictive ability around the 50% line. The ROC curve shows the RF forest had better predictive ability than the SVC with higher AUC values. For all ROC curves it is evident that in order to have high true positive rates, the false positive rates would also be very high. This proves that these models did not accurately predict the RTA/RTS stages very well.

Overall, for both RTA and RTS predictions it was found that the RF model predicted better. However, the model performance was not very good, which is likely due to the lack of data. It is hopeful that with more data, the RF models could be used to accurately predict RTS and RTA stages and could then be incorporated into the APP for use by concussed users.

Chapter 9

Discussion

Chapters 4 through 8 present the results found from various models created to predict RTA and RTS stages. From algorithms creating using previously acquired concussion data, it was found that best predictions occurred when stage 1 was removed and the final two stages of each protocol were combined. Stage 1 could be removed since the protocols advise staying in stage 1 for just the first 24 to 48 hours after injury. Given the short period in this stage, and the clear protocols for moving to the next stage, it would be unnecessary to determine for the user. The guidelines for using the APP could clearly state to start only following predictions after the first 24-48 hours. Furthermore, this stage had little data collected in the previous study. This is also likely attributed to the short time period participants would spend in this stage, and the increased difficulty of recruiting participants immediately after injury. The last two stages of each protocol were combined to once again decrease the number of stages that needed to be predicted by each algorithm. It was found that there were misclassifications between these stages likely due to similarity in the data. The protocols show that the last RTA/RTS stage is just a full return to school or activity,

whereas the second last stage is almost a full return with slight limitations. This could mean that symptoms would be very similar between stages, and there would be very little difference in the level of physical activity. By combining them, there was more data in the final stage to be predicted, also balancing out the dataset slightly more. Overall, removing stage 1 and combining the final stages removed two classes that needed to be predicted, decreasing complexity of the algorithms and allowing room for improvement. This case found the best results and shows promise in predicting RTA/RTS stages.

Given there were slight differences between data collected by the earlier study and data that would be collected by the APP, new models were required on a new set of participants to create algorithms that could be used in the APP. The most successful model was matched to create algorithms for new data, since it was assumed that it would similarly have the most success. The algorithms were made on a dataset of 5 participants, which was low due to Covid-19 restrictions. Algorithms were not as successful as those found with previous data, likely due to limitations in data size. A five participant pool was not enough to provide good training and testing data for the algorithms. Furthermore, it was seen that just one participant had data in all stages of return. Each participant only used the APP for approximately two weeks, meaning there was a chance they didn't use it for the full recovery and therefore did not capture all stages of return for RTA and RTS. Although overall accuracies of the models were lower, the results show good indication that they can successfully predict some stages of return, especially in the stage that had more data and higher distributions between participants. Given these results, the models can be used moving forward on larger datasets to attempt to improve results.

In each test, both RF and SVC models were tested. Through each trial, it was clear that the RF algorithms performed better than SVC. RF classifiers have been found in literature to have good accuracy compared to other models. Furthermore, it has been seen that these classifiers have decreased training time compared to SVC, and are better at handling noise/outliers in a dataset (Parmar *et al.*, 2019). It was suspected that the RF algorithms would perform better than the SVC in this study, mostly since outlier detection was not done in preprocessing. This puts the RF model at a advantage since, as mentioned above, SVC's are more sensitive to outliers. Furthermore, a benefit of the RF model was the training time, as it was observed during training that the RF classifiers all trained much quicker than the SVc. This is highly advantageous in this study since the end goal is implementation into an APP, which has device battery limitations. A model that runs more quickly could decrease time of background use and therefore limit the risk of using too much battery. Based on the results found in this study, RF models predicted RTA/RTS stages with higher accuracy, and should therefore be used moving forward.

Chapter 10

Conclusions

In this thesis, machine learning algorithms to predict return to school and return to activity stages were created. Four iterations were done to find the best algorithm, including predicting all stages, predicting all stages with feature reduction, predicting stages without stage 1, and predicting stages without stage 1 and while combining the last two stages. It was found that when reducing the number of stages by removing stage 1 and combining the last two stages, predictive ability of the models were higher than previous iterations. In general, the random forest models performed better than the support vector machine models for all iterations. Furthermore, similar models were created to predict RTA and RTS stages on newly collected data, and preliminary results were found on a small set of participants. Future work is required to complete models that can be implemented into the smartphone APP to be used to predict RTA and RTS stages for children with concussions.

10.1 Future Work

With the conclusions of this study, some future work is still required to complete the expectations set out at the beginning. Firstly, processing and modelling of newly collected data is required. First iterations of the APP have been released and participants have been recruited to collect data throughout their recovery. This recorded data could be used alongside the small data set described in Chapter 8, and models could be retrained to improve algorithm performance. Once modelling of these algorithms is complete, they could then be exported to CoreML, which is the language required for implementing a MLA in an Apple APP. Once implemented into the APP, further recruitment and data collection would be required to analyze model performance. Model performance could then be compared to brain recovery using medical imaging techniques. Stages of recovery determined by the model could be compared to symptom severity and functional brain recovery through the RTA/RTS stages by using "TBIfinder" brain analysis software(TBIFinder, 2021). "TBIfinder" provides personalized brain injury analysis using MRI and Big Data techniques to identify and locate brain injury, and quantify severity(TBIFinder, 2021).

Further work to try and improve model performance could be done by exploring other options, such as data augmentation, and additional pre-processing such as filtering, or changes in the window length or overlap. Finally, different extracted features from the accelerometer data could also be explored to test if a different feature set improves predictive ability of the models. Features derived from heart rate and heart rate variability could also be useful, and therefore collecting this data from future participants and incorporating it into the feature sets could benefit model performance.

10.2 Limitations

The biggest limitation in this study was the lack of distribution in the data, decreasing the predictive ability of the models for stages where data was limited. This was apparent in all iterations using the previously collected data, and was especially apparent when analyzing the newly collected data. With the newly collected data, limitations also included lack of data due to a low number of participants in the study. Participant recruitment was difficult due to the Covid-19 pandemic, since school and sports were cancelled for all kids, decreasing the incidence of concussions and the ability to run research projects. Covid-19 also limited the availability of research MRI scanning, preventing the use of MRI to compare imaging results to stage of return predicted from the algorithms, which was initially a goal of this study.

A second limitation which could become more apparent in future work, was the difference in devices used to collect accelerometer data in the previous study compared to the new study. Previous data was collected by an Actigraph worn on the waist, whereas new data was collected by an Apple watch worn on the wrist. These differences could cause the models created on the previous data to be less transferable to the new data, and could therefore require more pre-processing on the new data to reach the same level of model performance.

Appendix A

Appendix A: Code

A.1 Matlab Code

A.1.1 Symptom Data Cleansing

```
close all
clear all
%everyone, everyone with 2 weeks removed, remove missing data,
%just 10 to 18 yrs


data = readtable('SymptomData_LaurenEdits_8to18yrs.csv');
a = data.Properties.VariableNames;
a = a(1, 3:66);
data5to8 = readtable('SymptomData_LaurenEdits_5to8yrs.csv');
a5to8 = data5to8.Properties.VariableNames;
```

```
a5to8 = a5to8([1, 3, 4, 5, 6, 7, 8, 9, 18:44]);
datamat5to8 = data5to8{:, [1, 3, 4, 5, 6, 7, 8, 9, 18:45]};
symptomlabels = a([1, 2, 3, 7:37]);
labels = string(symptomlabels);
datamat = data{:, [1, 3:37]};
totaldata = [datamat;datamat5to8];
totalagemat = totaldata(:,3);

totagecount = 0;
%Counts how many participants there are in total

for r=1:length(totalagemat)
  if totalagemat(r) >= 0
       totagecount=totagecount+1;
   end
end

%Changes all NaN values that represent having no symptoms to 0
%done for case when participant specified that they had
%no symptoms in the last 48 hours
for d=1:length(totaldata(:,1))
   if totaldata(d, 4) == 0 %checks entry for symptom at 48 hours
        totaldata(d, 5) = 0; %sets 'recurring symptoms' to 0
        for c=9:32
```

```
                totaldata(d, c)=0; %changes NaN to 0
            end
        end
    end

%This next block uses the original data matrix
%to calculate the number of
%datapoints if all 2 week time period
%entries are removed.
no2week = totaldata(:,:);
week2totcount = 0;
%This for loop calculates how many rows of data there are
%for 2 weeks time period entries.

for c=1:length(no2week(:,2))
    if isNaN(no2week(c,4))
        week2totcount = week2totcount+1;
    end
end

%Remove all 2 week rows from the data set
no2week(any(isNaN(no2week(:, 4)),2), :) = [];
%Creates a new symptoms matrix containing
%age, sex, rta, rts and the symptoms.
```

```
symptomstotal = no2week(:, [1, 2, 3, 7:35]);
%Calculate how many participants and rows of data there are
%without 2weeks data
no2weekage= no2week(:,3);
totagecount2week = 0;

for r=1:length(no2weekage)
    if isNaN(no2weekage(r)) == 0
        totagecount2week=totagecount2week+1;
    end
end

%Any row with at least 1 missing value is removed
for c = 4:32
    symptomstotal(any(isNaN(symptomstotal(:, c)),2), :) = [];
end

no2weekage2= symptomstotal(:,3);
totagecountmiss = 0;

for r=1:length(no2weekage2)
    if no2weekage2(r) >= 0
        totagecountmiss=totagecountmiss+1;
    end
```

```
end

%Creates a vector of just the ages of participants
%to calculate how many
%participants and observations there would be if
%all participants under 10 were removed.
agemattotal = symptomstotal(:,3);
totalpart=0;
pover10=0;

for r=1:length(agemattotal)
    if agemattotal(r) >= 0
        totalpart=totalpart+1;
    end
    if agemattotal(r) >= 10
        pover10 = pover10+1;
    end
end

%creates a new vector an fills all NaN values with the
%previous value, in
%order to count the total number of observations for
%participants over the  age of 10.
```

```
newagemattotal = fillmissing(agemattotal,'previous');
agecount = 0;

for r=1:length(newagemattotal)
    if newagemattotal(r) >= 10
        agecount=agecount+1;
    end
end

sym_10to18 = symptomstotal;
  sym_10to18(:,3) = fillmissing(sym_10to18(:,3), 'previous');
rows = zeros(length(sym_10to18(:,1)),1);
n=1;

for i = 1:length(sym_10to18(:,3))
    if sym_10to18(i,3) < 10
        rows(n) = i;
        n=n+1;
    end
end

rows( all(~rows,2), : ) = [];
sym_10to18(rows, :) = [];
symptom_data = array2table(sym_10to18);
```

```
writetable(symptom_data, "symptoms_10to18_cleaned.csv");
```

A.2 Python Code

A.2.1 Remove Non-Wear Code

```
import pandas as pd
import numpy as np

 #load file with header names
non_wear = pd.read_csv('1098_wearinfo.csv',
           names=['Start_time', 'Stop_time', 'Wear',
           'Length', 'Use'])

 #drops columns that aren't useful
non_wear = non_wear.drop(non_wear.index[0:3])

samp_freq = 30

wear = non_wear.Length.astype(float)

 #cumulative sum for each value to get  total time from
 begining for each wear/nonwear time
```

```
wear = np.cumsum(wear, axis=0)


 #load accelerometer data and skip 10 rows
 #which does not contain any data
sig = pd.read_csv('1098_raw.csv', skiprows=10)

 #set to length of sample for each
rem_times = wear*60*samp_freq

rem_times[len(rem_times)+2] = len(sig)
rem_vec = np.append(0, rem_times)

 #create vector from 0 to length of wear-time vectors
 #for the loop iterators
loop_vec = np.arange(0, len(rem_vec)-1, 2)

 #create an index vector for removal
rem_ind = []

 #for loop loops through the loop vector to get every
 # other value as the starting index and in the loop
 #gets the next value
 #in the wear length vector as the end index.
```

```
 #Saves these values in an index vector
for i in loop_vec:
    index = np.arange(rem_vec[i], rem_vec[i+1])
    rem_ind.append(index)

 #changes to one array
ind_del = np.concatenate(rem_ind, axis=0).astype(int)

 #removes all values between start and end index from
 #accelerometer signal
new_sig = sig.drop(sig.index[ind_del])

new_sig.to_csv('1098_clean.csv')  #saves as new file
```

A.2.2 Windowing

```
import pandas as pd
import numpy as np
from datetime import datetime
from scipy import stats
import math

 #load symptom data
dates_keep = pd.read_csv('symptoms_10to18_cleaned_0to1_dated.csv')
```

```
part = "1016"  #value is changed depending on which participant
#data is being windowed

dates_keep = dates_keep.dropna(subset=['Participant'])

dates_keep['Participant']=dates_keep['Participant'].astype(str)

 #Create a date vector with just the date of each symptom recording
df1 = dates_keep[dates_keep['Participant'].str
      .contains('%s' % part)]
df1 = df1.reset_index()
dates = df1.Date

 #Convert the string date to a type datetime
dates.dtype
keepdates = pd.to_datetime(dates)

 #Sets a unique variable for each date that has symptom data.
 #This section is manually
 #changed for each participant depending on how many days
 #they had symptom recordings
 #for. This example is for participant 1016
day1 = keepdates[0]
day2 = keepdates[1]
```

```
day3 = keepdates[2]
 # ... for number of dates


 #Accelerometer file is loaded in. Some participants had
 #multiple files and therefore
 # the option of multiple files is present. The index is
 #reset for each file so they all
 #start at 0 indexing. Files are concatenated together
 #to create one signal file
sig1 = pd.read_csv('%sa_clean.csv' % part,
      names=['Date', 'x', 'y', 'z'])
sig2 = pd.read_csv('%sb_clean.csv' % part ,
      names=['Date', 'x', 'y', 'z'])
 #sig3 = pd.read_csv('%sc_clean.csv' % part,
      names=['Date', 'x', 'y', 'z'])
 #sig4 = pd.read_csv('%sd_clean.csv' % part,
      names=['Date', 'x', 'y', 'z'])
sig1 = sig1.drop(sig1.index[0])
sig2 = sig2.drop(sig2.index[0])
 #sig3 = sig3.drop(sig3.index[0])
 #sig4 = sig4.drop(sig4.index[0])
sig = np.concatenate((sig1, sig2))
sig = pd.DataFrame(sig, columns=['Date', 'x', 'y', 'z'])
```

```
#sets a window size and shift size in seconds, and the
#sampling frequency of the signal  #in Hz
win_size = 30
win_shift = 15
samp_freq = 30

#creates a vector containing the dates in the signal
day = sig.Date

#Converts date to a datetime, and removes the time so just
#the day is left.
date_time=pd.to_datetime(day)
date_day = date_time.dt.date

#Creates a signal for each day that needs to be kept for analysis
day1_sig = date_day[date_day == day1]
day2_sig = date_day[date_day == day2]
day3_sig = date_day[date_day == day3]
# ... continues for number of dates

#Creates an index for each date to know which samples need
#to be kept from the original signal.
day1_ind = day1_sig.index
day2_ind = day2_sig.index
```

```
day3_ind = day3_sig.index
 # ... continues for number of dates

 #Creates one vector containing all indexes to account for all
 #dates with both symptom and accelerometer
 #data (this ex. has 26 days).
keep_ind = np.concatenate((day1_ind, day2_ind, day3_ind,
  day4_ind, day5_ind, day6_ind, day7_ind,   day8_ind, day9_ind,
  day10_ind, day11_ind, day12_ind, day13_ind, day14_ind, day15_ind,
  day16_ind,   day17_ind, day18_ind, day19_ind, day20_ind, day21_ind,
  day22_ind, day23_ind, day24_ind,
  day25_ind, day26_ind))

 #Creates a signal dataframe with only the dates wanted
sig_sym = sig.loc[keep_ind]

 #Resets the indexing to 0 and creates a 1D array for
 #each signal (x, y, z axes and date)

sig_sym = sig_sym.reset_index()

x = sig_sym.x
y = sig_sym.y
z = sig_sym.z
```

```
symday = sig_sym.Date

 #function to window the data - provided by colleague Ama Simons
#requires 1D signal array, window size, window shift,
#and sampling frequency as inputs. Outputs windowed
#sample array and number of windows
def windowing(array, win_size, win_shift, samp_freq):

    j=0

  #how many samples in win_size window
    winsamp = win_size*samp_freq
    winsamp_shift = int(win_shift*samp_freq)

  #takes window samples, then transposes to (1,  n_samples)
    winsig = array[j:j+winsamp].T

    count = 1

    if win_shift:

        j+=winsamp_shift

        while (j+winsamp<array.shape[0]):
```

```
            win = array[j:j+winsamp].T
            count = count + 1
            # concatenate     the windows by the rows
            winsig = np.concatenate((winsig,win),axis=0)
            j+= winsamp_shift

        return winsig, count

    else:  # in the case that window is zero

        j+=winsamp

        while (j+winsamp<array.shape[0]):

            win = array[j:j+winsamp].T
            count = count + 1
            # concatenate the windows by the rows
            winsig = np.concatenate((winsig,win),axis=0)
            j+= winsamp

        return winsig, count

#Passes each 1D signal array into the windowing
```

```
 #function to return the windowed signal.
{[winsigx, count]} = windowing(x, win_size, win_shift, samp_freq)
{[winsigy, count]} = windowing(y, win_size, win_shift, samp_freq)
{[winsigz, count]} = windowing(z, win_size, win_shift, samp_freq)
{[windate, count]} = windowing(symday, win_size, win_shift, samp_freq)

 #Changes dates to a datetime and removes the timestamp.
win_day = pd.to_datetime(windate).normalize()

winsigx = winsigx.astype(float)
winsigy = winsigy.astype(float)
winsigz = winsigz.astype(float)

 #Creates a day vector containing integer values for each day.
 #ie. first day with symptom data is day 1.
day_num = []

for sample in win_day:
    if sample == day1:
        day_num.append(1)
    elif sample == day2:
        day_num.append(2)
    elif sample == day3:
        day_num.append(3)
```

```
    # ... continues for number of days

    #Extract features from the windows.Average day is used to
    #remove overlapping signal between dates. Since
    #labelling for stages is per day then the one window
    #overlapping between days should be removed.
    win_samples = win_size*samp_freq
    size = len(winsigx)
    win_start = np.arange(0, size, win_samples)

    std_vec = []
    day_vec = []
    svm = 0
    sma_x = 0
    sma_y = 0
    sma_z = 0

    for sample in win_start:
        std_x = np.std(winsigx[sample:sample+win_samples])
        mean_x = np.mean(winsigx[sample:sample+win_samples])
        max_x = max(winsigx[sample:sample+win_samples])
        min_x = min(winsigx[sample:sample+win_samples])
        mad_x = stats.median_absolute_deviation
                (winsigx[sample:sample+win_samples], axis=None)
```

```
        std_y = np.std(winsigy[sample:sample+win_samples])
        mean_y = np.mean(winsigy[sample:sample+win_samples])
        max_y = max(winsigy[sample:sample+win_samples])
        min_y = min(winsigy[sample:sample+win_samples])
        mad_y = stats.median_absolute_deviation
                (winsigy[sample:sample+win_samples], axis=None)
        std_z = np.std(winsigz[sample:sample+win_samples])
        mean_z = np.mean(winsigz[sample:sample+win_samples])
        max_z = max(winsigz[sample:sample+win_samples])
        min_z = min(winsigz[sample:sample+win_samples])
        mad_z = stats.median_absolute_deviation
                (winsigz[sample:sample+win_samples], axis=None)
        day_win = np.average(day_num[sample:sample+win_samples])

        for i in range(sample, sample+win_samples):
                svm = svm + (math.sqrt((winsigx[i]**2)+ (winsigy[i]**2) +
(winsigz[i]**2)))
                sma_x = sma_x + abs(winsigx[i])
                sma_y = sma_y + abs(winsigy[i])
                sma_z = sma_z + abs(winsigz[i])

        sma = (1/win_size)*(sma_x + sma_y + sma_z)
        std_vec.append((std_x, mean_x, max_x, min_x, mad_x, std_y,
        mean_y, max_y, min_y, mad_y, std_z, mean_z, max_z, min_z,
```

```
    mad_z, svm, sma, day_win))


    svm = 0
    sma_x = 0
    sma_y = 0
    sma_z = 0


cols = ['Std_x', 'Mean_x', 'Max_x', 'Min_x', 'MAD_x',
'Std_y', 'Mean_y', 'Max_y', 'Min_y', 'MAD_y', 'Std___z',
'Mean_z', 'Max_z', 'Min_z', 'MAD_z', 'SVM', 'SMA', 'Date']
feature_mat = pd.DataFrame(std_vec, columns = cols)

 #Removes overlapping days and creates dataframe with features
 #(this ex. has 26 days)
df1 = feature_mat[(feature_mat['Date'].isin([1.0, 2.0, 3.0, 4.0,
5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0,  12.0, 13.0, 14.0, 15.0,
16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0]))]


 #Saves dataframe as CSV
df1.to_csv('#s_featmat.csv' # part)
```

A.2.3 Data Fusion

```
import pandas as pd
```

```
 # load both symptom and acc feature datasets
sym = pd.read_csv('symptoms_dates.csv')
features = pd.read_csv('features_total.csv')

 #take columns containing desired features
symptoms = sym.iloc[:, 0:33]

 #fill in NaN sex and age spaces for each participant
symptoms['Sex'] = symptoms['Sex'].fillna(method='ffill')
symptoms['Age'] = symptoms['Age'].fillna(method='ffill')

 #create date vectors for each data type for comparison
symptom_date = symptoms['Date']
features_date = features['Date']

 #Loop to compare participant and date of both data types.
 #If both are the same then data is combined and added as new
 #row to total feature matrix
feat_df = []
for i in range(0, len(symptoms)):
    for j in range(0, len(features)):
        if features.Part[j] == symptoms.Participant[i]:
            if features_date[j] == symptom_date[i]:
```

```
                    symp_row = symptoms.iloc[i]
                    feat_row = features.iloc[j]
                    new_row = pd.concat((feat_row, symp_row))
                    feat_df.append(new_row)

 #Convert to array
import numpy as np
data_feat = np.array(feat_df)

 #convert to dataframe
feat_mat = pd.DataFrame(data_feat)

 #Create column labels to add to dataframe
cols = ['Part', 'Std x', 'Mean x', 'Max x',
      'Min x', 'MAD x', 'Std y', 'Mean y', 'Max y', 'Min y',
      'MAD y', 'Std z', 'Mean z', 'Max z', 'Min z', 'MAD z',
      'SVM', 'SMA', 'DateNum', 'Date','Participant', 'Date1',
      'Sex', 'Age', 'RTA', 'RTS', 'pcsiy48_ha', 'pcsiy48_ns',
      'pcsiy48_bl', 'pcsiy48_dz', 'pcsiy48_ft', 'pcsiy48_ds',
      'pcsiy48_lt', 'pcsiy48_ni', 'pcsiy48_ir', 'pcsiy48_sd',
      'pcsiy48_nv', 'pcsiy48_em', 'pcsiy48_sw', 'pcsiy48_mf',
      'pcsiy48_cc',  'pcsiy48_rm', 'pcsiy48_vs', 'pcsiy48_cf',
      'pcsiy48_cl', 'pcsiy48_sl', 'pcsiy48_dif',
      'pcsiy48_ts', 'pcsiy48_nk', 'pcsiy48_sp', 'cogact48',
```

```
        'schld48', 'treat48']

    #Take desired columns
    total_feat = feat_mat.iloc[:,2:55]

    #label columns
    total_feat.columns=cols

    #Drop columns that aren't required
    final_mat = total_feat.drop(['DateNum', 'Participant',
                'Date1'], axis=1)

    #Save as csv for later use
    final_mat.to_csv('feature_mat.csv')
```

A.2.4 Modelling

Data Preparation for Modelling

```
import pandas as pd
import numpy as np
import pylab as pl
import math
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.utils import shuffle
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

features = pd.read_csv('feature_mat.csv')

RTA_label = features.RTA
RTS_label = features.RTS
groups = features.Part

features = features.drop(['Unnamed: 0', 'Part', 'Date','RTA', 'RTS'],
           axis = 1)

  from sklearn.model_selection import GroupKFold

gkf = GroupKFold(n_splits=3)
trainRTA, testRTA = next(gkf.split(features, RTA_label, groups=groups))
X_trainRTA = features.loc[trainRTA]
y_trainRTA = RTA_label.loc[trainRTA]
X_testRTA = features.loc[testRTA]
y_testRTA = RTA_label.loc[testRTA]

trainRTS, testRTS = next(gkf.split(features, RTS_label, groups=groups))
```

```
X_trainRTS = features.loc[trainRTS]
y_trainRTS = RTS_label.loc[trainRTS]
X_testRTS = features.loc[testRTS]
y_testRTS = RTS_label.loc[testRTS]

from sklearn.preprocessing import StandardScaler

scalerRTA = StandardScaler()
trainRTA = scalerRTA.fit_transform(X_trainRTA)
testRTA = scalerRTA.transform(X_testRTA)

scalerRTS = StandardScaler()
trainRTS = scalerRTS.fit_transform(X_trainRTS)
testRTS = scalerRTS.transform(X_testRTS)
```

General RF Model

```
#Grid search to determine best parameters

from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10,
            stop = 1500, num = 10)]
```

```
 # Number of features to consider at every split
max_features = ['sqrt']
 # Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
 # Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
 # Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
 # Method of selecting samples for training each tree
bootstrap = [True, False]
class_weight = ['balanced']

 # Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap, 'class_weight' : class_weight}

print(random_grid)
 #Example - values change depending on results from randomized search
from sklearn.model_selection import GridSearchCV
```

```
#Changes dependant on what the randomized grid search found
# Number of trees in random forest
n_estimators = [450, 500, 550]
# Number of features to consider at every split
max_features = ['sqrt']
# Maximum number of levels in tree
max_depth = [60, 70, 80]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [5]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]
# Method of selecting samples for training each tree
bootstrap = [False]
class_weight = ['balanced']

param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap, 'class_weight' : class_weight}

rf = RandomForestClassifier()
```

```
 # Grid search of parameters, using 5 fold cross validation
modelRTS = GridSearchCV(estimator=rf, param_grid=param_grid,
scoring='neg_mean_absolute_error',
                                cv = 5, verbose=10,
                                return_train_score=True)
modelRTS.fit(trainRTS, y_trainRTS)
print(modelRTS.best_params_, modelRTS.best_score_)

y_score2 = modelRTS.predict_proba(testRTS)
predictionRTS = np.argmax(y_score2, axis=1)

from sklearn.preprocessing import label_binarize

 #Classes change depending on which model and if RTA/RTS
testRTS_label = label_binarize(y_testRTS, classes=[1, 2, 3, 4, 5])
trainRTS_label = label_binarize(y_trainRTS, classes=[1, 2, 3, 4, 5])
n_classesRTS = testRTS_label.shape[1]
```

General SVC Model

```
from sklearn.model_selection import GridSearchCV

param_grid = [{'kernel': ['rbf'], 'gamma': [1e-1, 1e-2, 1e-3, 1e-4],
  'C': [1, 10, 100, 1000, 10000], 'class_weight' : ['balanced']}]
```

```
modelRTS = GridSearchCV(SVC(), param_grid=param_grid, cv=5,
verbose = 2, return_train_score=True)
modelRTS.fit(trainRTS, y_trainRTS)

print("The best parameters are %s with a score of %0.2f"
    # (modelRTS.best_params_, modelRTS.best_score_))


final_modelRTS = modelRTS.best_estimator_
Y_predRTS = final_modelRTS.decision_function(testRTS)

predictionRTS = np.argmax(Y_predRTS, axis=1)
```

Result Visualization

ROC

```
from sklearn.metrics import roc_curve, auc

fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classesRTS):
    fpr[i], tpr[i], _ = roc_curve(testRTS_label[:,i],
```

```
            y_score2[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

 # Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(testRTS_label.ravel(),
y_score2.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

plt.figure()
width = 2
plt.plot(fpr[0], tpr[0], color='red',
         lw=width, label='Stage 1 AUC = %0.2f)' % roc_auc[0])
plt.plot(fpr[1], tpr[1], color='green',
         lw=width, label='Stage 2 AUC = %0.2f)' % roc_auc[1])
plt.plot(fpr[2], tpr[2], color='darkorange',
         lw=width, label='Stage 3 AUC = %0.2f)' % roc_auc[2])
plt.plot(fpr[3], tpr[3], color='purple',
         lw=width, label='Stage 4 AUC = %0.2f)' % roc_auc[3])
plt.plot(fpr[4], tpr[4], color='yellow',
         lw=width, label='Stage 5 AUC = %0.2f)' % roc_auc[4])
plt.plot([0, 1], [0, 1], color='navy', lw=width, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('ROC for Random Forest RTS Prediction')
plt.legend(loc="lower right")
plt.savefig('rfROC_RTS.png')
plt.show()
```

Confusion Matrix

```
#Changes depending on model and RTA or RTS
predictionRTS[predictionRTS == 4] = 5
predictionRTS[predictionRTS == 3] = 4
predictionRTS[predictionRTS == 2] = 3
predictionRTS[predictionRTS == 1] = 2
predictionRTS[predictionRTS == 0] = 1


confRTS = confusion_matrix(y_testRTS, predictionRTS)

ax= plt.subplot()
# labels for x-axis
x_axis_labels = ['Stage 1','Stage 2','Stage 3','Stage 4', 'Stage 5']
# labels for y-axis
y_axis_labels = ['Stage 2','Stage 3','Stage 4','Stage 5', 'Stage 6']

sns.heatmap(confRTA, xticklabels = x_axis_labels,
```

```
    yticklabels = y_axis_labels, annot=True, ax = ax);

 # labels, title and ticks
   ax.set_xlabel('Predicted labels');
ax.set_ylabel('True labels');
ax.set_title('Random Forest RTS Confusion Matrix');

plt.savefig('rfconf_rts.png')
plt.show()
```

Classification Report

```
from sklearn.metrics import plot_confusion_matrix

print(confusion_matrix(y_testRTS, predictionRTS))

print(classification_report(y_testRTS, predictionRTS))

from sklearn.metrics import accuracy_score, f1_score

print('Accuracy:', accuracy_score(y_testRTS, predictionRTS))
print('F1:', f1_score(y_testRTS, predictionRTS, average='micro'))
```

A.2.5 Feature Selection

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('feature_mat.csv')

features = data.drop(['Unnamed: 0', 'Part', 'Date','RTA', 'RTS'],
        axis = 1)

corrMat = features.corr()
plt.figure(figsize=(25,25))
sns.heatmap(corrMat, annot=True)
plt.title("Correlation Matrix for Feature Selection")
plt.savefig("corr_mat.png")
plt.show()
```

A.2.6 Removing Stages

Removing Stage 1

```
#Sets separate file for RTA and RTS, and removes the
#other label from each.
#Drops any row that has stage 1.
featuresRTS = features.drop(['RTA'], axis=1)
```

```
featuresRTA = features.drop(['RTS'], axis=1)
dataRTA = featuresRTA[(featuresRTA.RTA != 1)]
dataRTS = featuresRTS[(featuresRTS.RTS != 1)]
```

Joining Last Two Stages

```
#Replaces last stage with label for second last stage
dataRTA['RTA'] = dataRTA['RTA'].replace(6, 5)
dataRTS['RTS'] = dataRTS['RTS'].replace(5, 4)
```

A.2.7 New Data

Symptom Processing

```
import pandas as pd
import numpy as np


sym1 = pd.read_csv("symptom_data_dates.csv")
sym2 = pd.read_csv("survey_results_summary2.csv")
sym = pd.concat([sym1, sym2], axis=0)

#Coded as since value for each
#participant to make referring
#to participants and calling them
```

```
#later in code easier
 sym["user_id"].replace({'c5027e89-e61e-40a9-929f-dc2d6e61d90d': "A",
 '7f371bb2-d7fe-4e0b-9faf-16d67cf0f6a5': "B",
 '894286ff-2619-4e65-9af7-ccee28f969c9' : "C",
 'f1c31998-56f6-4bf5-9b3e-05ebd0f66a8e' : "D" ,
 '3a3d29b6-5f59-412b-8910-4ea03ad51208' : "E"}, inplace=True)

 def process_sym(file):
     def rem_values(df, col, values):
     #function to keep rows that don't
     #include a certain value in a specified
     #column
    return df[df[col].isin(values) == False]

      #creates a new file that removes rows containing questions
      #that aren't wanted for the feature set.
     new_sym = rem_values(file,"question_text",
     ["Please rate 1-5 what you've been doing  today",
        "How are you feeling ? Take a minute to log your symptoms!",
        'Cognitive Activities', 'Physical Activities'])

      #removes columns that aren't wanted for the feature set
     new_sym = new_sym.drop(columns = ['survey_results_id',
          'survey_stage_id', 'light_activity_minutes_completed',
```

```
        'activity_minutes_goal',
        'moderate_activity_minutes_completed',
        'vigorous_activity_minutes_completed', 'question_id','id'])

    #file had two feel foggy responses (one had quotations)
    #so these responses were combined  to one response.
new_sym = new_sym.replace('Feeling "foggy"', 'Feeling foggy')

    #pivot table function takes the responses in question_text
    #column and uses those as new  column headings.
    #Values from answer_text
    #are used as the row values for each column,
    #and it is sorted using the participant id and
    #time they completed the survey.
    #table is then transposed and reset, adding the column
    #names for each symptom.

syms = pd.pivot_table(new_sym, values=['answer_text'],
index=['time_completed',  'user_id'],
columns='question_text', aggfunc='first')
syms = syms.T.reset_index(drop=True).T.rename(columns=
{0:'Balance Problems',
    1: 'Being  more emotional', 2:'Clumsiness',
    3: 'Confusion with tasks', 4: 'Difficulty concentrating',
```

```
    5:'Difficulty remembering', 6: 'Dizziness', 7: 'Drowsiness',
    8: 'Fatigue', 9: 'Feeling foggy', 10: 'Feeling slowed down',
    11: 'Headaches', 12: 'Irritability', 13: 'Nausea',
    14: 'Neck Pain',
    15: 'Nervousness', 16: 'Sadness',17: 'RTA', 18: 'RTS',
    19: 'Sensitivity to light', 20: 'Sensitivity to noise',
    21: 'Sleep Problems', 22: 'Slow to answer questions',
    23: 'Vision problems'})

 #resets the index for time_completed and user_id
newfile = syms.reset_index(level = ['time_completed',
'user_id'])

 #replaces any entry that meant they had the
 #symptom to a 1, and any response meaning they didnt to a 0
newfile = newfile.replace(['yes','better', 'worse', 'same'], 1)
newfile = newfile.replace(['no','gone'], 0)

 #converts the time completed to a datetime
newfile['time_completed'] =
pd.to_datetime(newfile['time_completed'])

 #sorts the file to group the users together,
 #and sort the time in chronological order for  each user.
```

```
    newfile = newfile.groupby('user_id').apply(lambda x:
    x.sort_values('time_completed'))

     #creates a label vector to separate data and labels
    labels = newfile[['RTA', 'RTS', 'user_id',
    'time_completed']].copy()

     #removes labels from feature file
    symptoms = newfile.drop(columns = ['RTA', 'RTS'])

    return(symptoms, labels)

feat, labels = process_sym(sym)
```

Accelerometer Pre-Processing

Data Combination

```
import pandas as pd
import numpy as np

 #allows participant input so do each part file.
part = input("Enter part id: ")
print("Part ", part)

 #Load in acc features, date column
```

```
#(dates were formatted wrong when data was first
#sent so good formating of dates in this file)
# symptom features and heart rate column.

file = pd.read_csv("features_part%s.csv" % part)
dates = pd.read_csv("dates%s.csv" % part)
sym = pd.read_csv("newdata2_symptoms.csv")
hr1 = pd.read_csv("hr_new.csv")

print(sym.head())
print(file.head())

#function that keeps rows that contain a specified value
def keep_rows_by_value(df, col, values):
        return df[df[col].isin(values) == True]

#use function to keep symptom rows that pertain to
#specific participant
part_sym = keep_rows_by_value(sym, "user_id", [part])
print(part_sym.head())
del part_sym["Unnamed: 0"]
part_sym = part_sym.reset_index()
del part_sym["index"]
```

```
#deletes dates column from acc file since dates need to be replaced
del file["Date"]
del dates["Unnamed: 0"]

#creates new acc file with proper date formating
acc = pd.concat((file, dates), axis = 1)

#keep hr rows that pertain to correct participant
hr = keep_rows_by_value(hr1, "user_id", [part])

#deletes unnecessary rows
del hr["Unnamed: 0"]
del hr["id"]
del hr["user_id"]

#sorts based on dates
hr2 = hr.sort_values("date_recorded")
hr2 = hr2.reset_index()
print(hr2.head())

del hr2["index"]
# User B has no hr data so comment out this section
#and just use symptom data
#Use this line if they don't have hr data and
```

```
#skip first for loop
  sym_hr = part_sym
print(part_sym.head())

'''
#This section combines symptom and heart rate data -
#only to be used if part has both
sym_hr = []
#nested for loop with conditional to find heart rate
#values from the last survey time
#If heart rate is recorded after the survey then it
#corresponds to that survey recording, up until the
#next survey time.
for i in range(0, len(part_sym)-1):
        for j in range(0, len(hr2)):
                a = part_sym.time_completed[i] <=
                  hr2.date_recorded[j] <
                  part_sym.time_completed[i+1]
                if a == True:
                        sym_row = part_sym.iloc[i]
                        feat_row = hr2.iloc[j]
                        new_row = pd.concat((feat_row, sym_row))
                        sym_hr.append(new_row)
sym_hr = pd.DataFrame(sym_hr)
```

```
print(sym_hr.head())

del sym_hr["time_completed"]
'''

 #resets the index of the combined array
sym_hr = sym_hr.reset_index()
print(sym_hr.head())
 #converts date recorded to a datetime value
sym_hr["date_recorded"] =pd.to_datetime(sym_hr["date_recorded"])
 #print(sym_hr.head())
del sym_hr["index"]

 #converts date to a datetime, and removes time zone
 #so values can be compared to dates in sym_hr
acc["Date"] = pd.to_datetime(acc["Date"])
acc["Date"] = pd.DatetimeIndex([i.replace(tzinfo=None)
      for i in acc["Date"]])
 #print(acc.head())
print(sym_hr.head())

   #nested for loop to combine sym_hr data with acc features
acc = acc.dropna()
```

```
#same idea as above, if acc features are from after the last
#survey/hr but before next then it is  combined with that data/
feat_df = []
for k in range(0, len(sym_hr)-1):
        for l in range(0, len(acc)):
                b = sym_hr.time_completed[k] <=
                        acc.Date[l] <
                        sym_hr.time_completed[k+1]  #use if no hr
                #b = sym_hr.date_recorded[k] <=
                        acc.Date[l] <
                        sym_hr.date_recorded[k+1]
                if b == True:
                        sym_row = sym_hr.iloc[k]
                        feat_row = acc.iloc[l]
                        new_row = pd.concat((feat_row, sym_row))
                        feat_df.append(new_row)

feat = pd.DataFrame(feat_df)
print(feat.head())

#saved as dataframe
feat.to_csv("features_total%s.csv" % part)
print("Done.")
```

```
# Once all participants are done, use this to combine
#all parts together into final feature set
import pandas as pd
import numpy as np

a = pd.read_csv("features_totalA.csv")
b = pd.read_csv("features_totalB.csv")
c = pd.read_csv("features_totalC.csv")
d = pd.read_csv("features_totalD.csv")
e = pd.read_csv("features_totalE.csv")

print(a.head())
print(a.date_recorded)
print(b.head())
print(b.time_completed)
print(c.head())
print(c.date_recorded)
print(d.head())
print(d.date_recorded)
print(e.head())
print(e.date_recorded)

a["RTA"] = a["RTA"].fillna(method='ffill')
a["RTS"] = a["RTS"].fillna(method='ffill')
```

```
    b["RTA"] = b["RTA"].fillna(method='ffill')
    b["RTS"] = b["RTS"].fillna(method='ffill')

    c["RTA"] = c["RTA"].fillna(method='ffill')
    c["RTS"] = c["RTS"].fillna(method='ffill')

    d["RTS"] = d["RTS"].fillna(method='ffill')
    d["RTA"] = d["RTA"].fillna(method='ffill')

e["RTS"] = e["RTS"].fillna(method='ffill')
e["RTA"] = e["RTA"].fillna(method='ffill')

    print(e.head())
    for col in a.columns:
            print(col)

    features_hr = pd.concat((a,c,d,e), axis = 0)

    print(features_hr.head())
    print(features_hr['date_recorded'])
    del features_hr["date_recorded"]

    features_nohr = pd.concat((a, b, c, d, e), axis = 0)
```

```
print(features_nohr.head())
print(features_nohr['date_recorded'])
print(features_nohr['time_completed'])

  del features_nohr["time_completed"]
del features_nohr["date_recorded"]

features_nohr.to_csv("features_nohr.csv")
features_hr.to_csv("features_hr.csv")
```

Appendix B

Feature Correlation Matrix

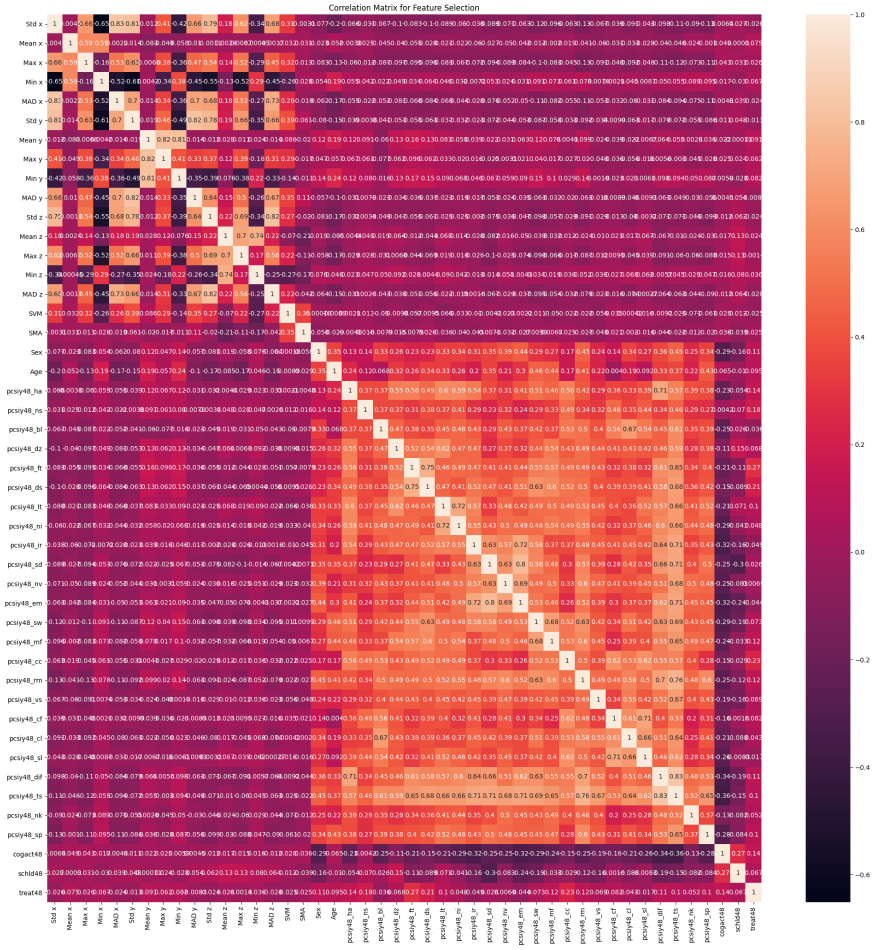


Figure B.1: Correlation matrix for full feature set. Heatmap shows highly correlated features with a lighter colour versus low correlated features with darker colours. Diagonal from top left to bottom right shows a value of 1 as it is comparing to itself.

Bibliography

Abaji, J. P., Curnier, D., Moore, R. D., and Ellemberg, D. (2016). Persisting effects of concussion on heart rate variability during physical exertion. *Journal of Neurotrauma*.

Abdull Sukor, A. S., Zakaria, A., and Abdul Rahim, N. (2018). Activity recognition using accelerometer sensor and machine learning classifiers. *Proceedings - 2018 IEEE 14th International Colloquium on Signal Processing and its Application, CSPA 2018*, pages 233–238.

Akoglu, H. (2018). Users guide to correlation coefficients. *Turkish Journal of Emergency Medicine*, **18**(3), 91–93.

Bergeron, M. F., Landset, S., Maugans, T. A., Williams, V. B., Collins, C. L., Wasserman, E. B., and Khoshgoftaar, T. M. (2019). Machine Learning in Modeling High School Sport Concussion Symptom Resolve. *Medicine and Science in Sports and Exercise*, **51**(7), 1362–1371.

Bhandari, A. (2020). Auc-roc curve in machine learning clearly explained.

Bilchick, K. C. and Berger, R. D. (2006). Heart rate variability.

Bishop, S. A., Dech, R. T., Guzik, P., and Neary, J. P. (2018). Heart rate variability and implication for sport concussion.

Cai, J., Luo, J., Wang, S., and Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, **300**, 70–79.

Canada, P. H. A. o. (2020). Government of canada.

CanChild (2017). Concussion/Mild Traumatic Brain Injury Guideline Brochures.

Chamard, E. and Lichtenstein, J. D. (2018a). A systematic review of neuroimaging findings in children and adolescents with sports-related concussion. *Brain Injury*.

Chamard, E. and Lichtenstein, J. D. (2018b). A systematic review of neuroimaging findings in children and adolescents with sports-related concussion. *Brain Injury*.

Chan, Y. H. (2003). Biostatistics 104. Correlational analysis. *Singapore Medical Journal*, **44**(12), 614–619.

Chen, J. K., Johnston, K. M., Frey, S., Petrides, M., Worsley, K., and Ptito, A. (2004). Functional abnormalities in symptomatic concussed athletes: An fMRI study. *NeuroImage*, **22**(1), 68–82.

Churchill, N. W., Hutchison, M. G., Graham, S. J., and Schweizer, T. A. (2018). Connectomic markers of symptom severity in sport-related concussion: Whole-brain analysis of resting-state fMRI. *NeuroImage: Clinical*.

Complete Concussion Management (2020). Concussion Tracker App.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*.

Davide Anguita, Alessandro Ghio, L. O. X. P. and Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. *Computational Intelligence and Machine Learning*, pages 24–26.

Davis, G. A., Purcell, L., Schneider, K. J., Yeates, K. O., Gioia, G. A., Anderson, V., Ellenbogen, R. G., Echemendia, R. J., Makdissi, M., Sills, A., Iverson, G. L., Dvořák, J., McCrory, P., Meeuwisse, W., Patricios, J., Giza, C. C., and Kutcher, J. S. (2017). The Child Sport Concussion Assessment Tool 5th Edition (Child SCAT5): Background and rationale. *British journal of sports medicine*, **51**(11), 859–861.

de Almeida Mendes, M., da Silva, I. C., Ramires, V. V., Reichert, F. F., Martins, R. C., and Tomasi, E. (2018). Calibration of raw accelerometer data to measure physical activity: A systematic review. *Gait and Posture*, **61**, 98–110.

de Mello, R. F. and Ponti, M. A. (2018). *Machine Learning:A Practical Approach on the Statistical Learning Theory*. Springer.

Dematteo, C., Dip, P., Stazyk, K., Ont, O. T. R., Giglia, L., Frcp, C., Mahoney, W., Frcp, C., Singh, S. K., Frcs, C., Hollenberg, R., Frcs, C., Harper, J. A., Missiuna, C., Ont, O. T. R., Law, M., Ont, O. T. R., Mccauley, D., and Randall, S. (2015a). A Balanced Protocol for Return to School for Children and Youth Following Concussive Injury.

Dematteo, C., Dip, P., Stazyk, K., Ont, O. T. R., Singh, S. K., Frcs, C., Giglia, L., Frcp, C., Hollenberg, R., Frcs, C., Malcolmson, C. H., Frcp, C., Mahoney, W., Frcp, C., Harper, J. A., Missiuna, C., Ont, O. T. R., Law, M., Ont, O. T. R., and

Mccauley, D. (2015b). Development of a Conservative Protocol to Return Children and Youth to Activity Following Concussive Injury.

Dematteo, C., McCauley, D., Stazyk, K., Harper, J., Adamich, J., Randall, S., and Missiuna, C. (2015c). Post-concussion return to play and return to school guidelines for children and youth: A scoping methodology. *Disability and Rehabilitation*, **37**(12), 1107–1112.

DeMatteo, C. A., Randall, S., Lin, C.-Y. A., and Claridge, E. A. (2019). What Comes First: Return to School or Return to Activity for Youth After Concussion? Maybe We Don't Have to Choose. *Frontiers in Neurology*, **10**, 1–9.

Developers, G. (2020). Classification: Precision and recall nbsp;—nbsp; machine learning crash course.

Duarte, E. and Wainer, J. (2017). Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters. *Pattern Recognition Letters*, **88**, 6–11.

Ebner, J., Emmanuel, Sight, S., and Ak (2021). Numpy standard deviation explained.

Echemendia, R. J., Meeuwisse, W., McCrory, P., Davis, G. A., Putukian, M., Leddy, J., Makdissi, M., Sullivan, S. J., Broglio, S. P., Raftery, M., Schneider, K., Kissick, J., McCrea, M., Dvořák, J., Sills, A. K., Aubry, M., Engebretsen, L., Loosemore, M., Fuller, G., Kutcher, J., Ellenbogen, R., Guskiewicz, K., Patricios, J., and Herring, S. (2017a). CONCUSSION RECOGNITION TOOL 5.

Echemendia, R. J., Meeuwisse, W., McCrory, P., Davis, G. A., Putukian, M., Leddy, J., Makdissi, M., Sullivan, S. J., Broglio, S. P., Raftery, M., Schneider, K., Kissick,

J., McCrea, M., Dvořák, J., Sills, A. K., Aubry, M., Engebretsen, L., Loosemore, M., Fuller, G., Kutcher, J., Ellenbogen, R., Guskiewicz, K., Patricios, J., and Herring, S. (2017b). Sport concussion assessment tool - 5th edition.

Echemendia, R. J., Meeuwisse, W., McCrory, P., Davis, G. A., Putukian, M., Leddy, J., Makdissi, M., Sullivan, S. J., Broglio, S. P., Raftery, M., Schneider, K., Kissick, J., McCrea, M., Dvořák, J., Sills, A. K., Aubry, M., Engebretsen, L., Loosemore, M., Fuller, G., Kutcher, J., Ellenbogen, R., Guskiewicz, K., Patricios, J., and Herring, S. (2017c). The Sport Concussion Assessment Tool 5th Edition (SCAT5): Background and rationale. *British journal of sports medicine*, **51**(11), 848–850.

Figo, D., Diniz, P. C., Ferreira, D. R., and Cardoso, J. M. (2010). Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, **14**(7), 645–662.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., and Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**(10), 906–914.

Galetta, K. M., Liu, M., Leong, D. F., Ventura, R. E., Galetta, S. L., and Balcer, L. J. (2016). The King-Devick test of rapid number naming for concussion detection: meta-analysis and systematic review of the literature. *Concussion*, **1**(2).

Giza, C. C. and Hovda, D. A. (2014). The new neurometabolic cascade of concussion. *Neurosurgery*, **75**(3), S24–S33.

Glover, G. H. (2011). Overview of functional magnetic resonance imaging. *Neurosurgery Clinics of North America*.

Guskiewicz, K. M. (2011). Balance Assessment in the Management of Sport-Related Concussion.

Halstead, M. E., Walter, K. D., McCambridge, T. M., Benjamin, H. J., Brenner, J. S., Cappetta, C. T., Demorest, R. A., Gregory, A. J., Koutures, C. G., LaBella, C. R., Martin, S. S., and Weiss-Kelly, A. K. (2010). Clinical report - Sport-related concussion in children and adolescents. *Pediatrics*, **126**(3), 597–615.

Halstead, M. E., Walter, K. D., and Moffatt, K. (2018). Sport-related concussion in children and adolescents. *Pediatrics*, **142**(6).

Hand, D. and Anagnostopoulos, C. (2013). When is the area under the receiver operating characteristic curve an appropriate measure of classifier performance? *Pattern Recognition Letters*, **34**(5), 492–495.

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.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Elements of Statistical Learning 2nd ed.*

Hawaii Concussion (2017). Post Concussion Symptom Scale.

Highmark Interactive (2020). OUR PRODUCTS.

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

Iyer, K. K., Zalesky, A., Barlow, K. M., and Cocchi, L. (2019a). Default mode network anatomy and function is linked to pediatric concussion recovery. *Annals of Clinical and Translational Neurology*.

Iyer, K. K., Barlow, K. M., Brooks, B., Ofoghi, Z., Zalesky, A., and Cocchi, L. (2019b). Relating brain connectivity with persistent symptoms in pediatric concussion. *Annals of Clinical and Translational Neurology*.

Jantzen, K. J., Anderson, B., Steinberg, F. L., and Kelso, J. A. (2004). A prospective functional MR imaging study of mild traumatic brain injury in college football players. *American Journal of Neuroradiology*, **25**(5), 738–745.

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

Jupyter, P. (2021).

Keightley, M. L., Singh Saluja, R., Chen, J. K., Gagnon, I., Leonard, G., Petrides, M., and Ptito, A. (2014). A functional magnetic resonance imaging study of working memory in youth after sports-related concussion: Is it still working? *Journal of Neurotrauma*, **31**(5), 437–451.

Kurowski, B. G., Wade, S. L., Dexheimer, J. W., Dyas, J., Zhang, N., and Babcock, L. (2016). Feasibility and potential benefits of a web-based intervention delivered

acutely after mild traumatic brain injury in adolescents: A pilot study. *Journal of Head Trauma Rehabilitation*.

Lavanya, P. G. and Mallappa, S. (2019). Activity recognition from accelerometer data using symbolic data approach. *Lecture Notes in Networks and Systems*, **43**, 317–329.

Lin, H. and Lin, C. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *Neural Computation*, (2), 1–32.

Lv, H., Wang, Z., Tong, E., Williams, L. M., Zaharchuk, G., Zeineh, M., Goldstein-Piekarski, A. N., Ball, T. M., Liao, C., and Wintermark, M. (2018). Resting-state functional MRI: Everything that nonexperts have always wanted to know. *American Journal of Neuroradiology*.

Mccrory, P., Meeuwisse, W. H., Aubry, M., Engebretsen, L., Johnston, K., Kutcher, J. S., Raftery, M., Sills, A., Benson, B. W., Davis, G. A., Ellenbogen, R., Guskiewicz, K. M., Herring, S. A., Iverson, G. L., Jordan, B. D., Kissick, J., Mccrea, M., Mcintosh, A. S., Maddocks, D., Schneider, K., Tator, C. H., and Turner, M. (2013). Consensus Statement on Concussion in Sport: The 4th International Conference on Concussion in Sport, Zurich, November 2012. **48**, 554–575.

McCrory, P., Meeuwisse, W., Dvořák, J., Aubry, M., Bailes, J., Broglio, S., Cantu, R. C., Cassidy, D., Echemendia, R. J., Castellani, R. J., Davis, G. A., Ellenbogen, R., Emery, C., Engebretsen, L., Feddermann-Demont, N., Giza, C. C., Guskiewicz, K. M., Herring, S., Iverson, G. L., Johnston, K. M., Kissick, J., Kutcher, J., Leddy, J. J., Maddocks, D., Makdissi, M., Manley, G. T., McCrea, M., Meehan, W. P., Nagahiro, S., Patricios, J., Putukian, M., Schneider, K. J., Sills, A., Tator, C. H.,

Turner, M., and Vos, P. E. (2017). Consensus statement on concussion in sport—the 5th international conference on concussion in sport held in Berlin, October 2016. *British Journal of Sports Medicine*, **51**(11), 838–847.

Murdaugh, D. L., King, T. Z., Sun, B., Jones, R. A., Ono, K. E., Reisner, A., and Burns, T. G. (2018). Longitudinal Changes in Resting State Connectivity and White Matter Integrity in Adolescents with Sports-Related Concussion. *Journal of the International Neuropsychological Society*.

NumPy (2021). numpy.mean.

pandas development team, T. (2020). pandas-dev/pandas: Pandas.

Paniccia, M., Verweel, L., Thomas, S. G., Taha, T., Keightley, M., Wilson, K. E., and Reed, N. (2018). Heart rate variability following youth concussion: How do autonomic regulation and concussion symptoms differ over time postinjury? *BMJ Open Sport and Exercise Medicine*.

Parmar, A., Katariya, R., and Patel, V. (2019). A review on random forest: An ensemble classifier. In J. Hemanth, X. Fernando, P. Lafata, and Z. Baig, editors, *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, pages 758–763, Cham. Springer International Publishing.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Prajapati, G. L. and Patle, A. (2010). On performing classification using SVM with radial basis and polynomial kernel functions. *Proceedings - 3rd International Conference on Emerging Trends in Engineering and Technology, ICETET 2010*, pages 512–515.

Purkayastha, S., Williams, B., Murphy, M., Lyng, S., Sabo, T., and Bell, K. R. (2019). Reduced heart rate variability and lower cerebral blood flow associated with poor cognition during recovery following concussion. *Autonomic Neuroscience: Basic and Clinical*.

Satchell, E. K., Friedman, S. D., Bompadre, V., Poliakov, A., Oron, A., and Jinguji, T. M. (2019). Use of diffusion tension imaging in the evaluation of pediatric concussions. *Musculoskeletal Science and Practice*.

scikit learn (2021a). Sklearn.ensemble.randomforestclassifier.

scikit learn (2021b). Sklearn.svm.svc.

Senthinathan, A., Mainwaring, L. M., and Hutchison, M. (2017). Heart rate variability of athletes across concussion recovery milestones: A preliminary study. *Clinical Journal of Sport Medicine*.

Shaffer, F. and Ginsberg, J. P. (2017). An Overview of Heart Rate Variability Metrics and Norms. *Frontiers in Public Health*.

Shrey, D. W., Griesbach, G. S., and Giza, C. C. (2011). The Pathophysiology of Concussions in Youth. *Physical Medicine and Rehabilitation Clinics of North America*, **22**(4), 577–602.

Starr, R. (2020). HIGHMARK INTERACTIVE'S EQ ACTIVE BRAIN TRACKING.

Swift, A., Heale, R., and Twycross, A. (2019). What are sensitivity and specificity? *Evidence Based Nursing*, **23**(1), 2–4.

TBIFinder (2021).

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.

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, **6**(60), 3021.

Weiss, G. M., Yoneda, K., and Hayajneh, T. (2019). Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, **7**, 133190–133202.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Wu, T., Merkley, T. L., Wilde, E. A., Barnes, A., Li, X., Chu, Z. D., McCauley, S. R., Hunter, J. V., and Levin, H. S. (2018). A preliminary report of cerebral white matter microstructural changes associated with adolescent sports concussion acutely and subacutely using diffusion tensor imaging. *Brain Imaging and Behavior*.

Yu, L. and Liu, H. (2003). Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. *Proceedings, Twentieth International Conference on Machine Learning*, **2**, 856–863.

Zemper, E. D. (2003). Two-year prospective study of relative risk of a second cerebral concussion. *American Journal of Physical Medicine and Rehabilitation*.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman Hall.

Zuckerman, S., Lee, Y., Odom, M., Solomon, G., Sills, A., and Forbes, J. (2012). Recovery from sports-related concussion: Days to return to neurocognitive baseline in adolescents versus young adults. *Surgical Neurology International*, **3**(1), 130.