12-2023

Experimental and Computational Platforms for Studying Systems Mechanobiology

Brendyn Miller
brendym@clemson.edu

EXPERIMENTAL AND COMPUTATIONAL PLATFORMS FOR STUDYING SYSTEMS MECHANOBIOLOGY

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Bioengineering

---

by
Brendyn James Miller
December 2023

Accepted by:
Dr. Jeremy Mercuri, Committee Chair
Dr. William Richardson
Dr. Delphine Dean
Dr. Jiro Nagatomi

ABSTRACT

Mechanical stimulation through physical activity has been shown to play an important role in treating and preventing several non-communicable diseases such as hypertension, lower back pain (LBP), type-2 diabetes mellitus, and several cancers. This is accomplished through the regulation of cellular behavior and tissue remodeling within the body at both the micro- and macro-scale levels. The goal of mechanobiology research is to gain in-depth knowledge and understanding of how cells sense physical forces in conjunction with other biochemical cues and translate those factors into important biological functions that either maintain tissue homeostasis or lead to pathological states. Understanding these processes can lead to the development of medical interventions that influence cells towards a desired outcome. While some of these processes can be observed naturally through thoughtful experimental design, most mechanobiological processes require the development of tools that can enable scientists and researchers to investigate a specific aspect of cell mechanobiology. In this PhD dissertation, we investigated the role that mechanical stimulation plays in regulating cell behavior in different chemical environments by subjecting cells cultured in monolayer to levels of uniaxial tension similar to those experienced in the annulus fibrosus (AF) in the intervertebral disc (IVD). We then developed both an experimental and computational platform that could be used to study the effects of mechanical stimulation on physiological function at both the microscopic and macroscopic levels. The experimental platform was designed to enable long term application of mechanical forces onto cell-seeded tissue scaffolds with the intention of developing a bioreactor to enable in vitro studies focused on investigating how mechanical forces influence cell behavior in healthy and diseased IVDs. The computational platform employed machine learning

algorithms and data science strategies to examine how clinical measurements such as blood pressure and blood serum analytes influence patient risk for masked hypertension in a young, apparently healthy population.

DEDICATION

To my Lord and Savior, Jesus Christ, whose example as a Healer inspired me to go into the medical field in the hope that my work could bless the lives of others through the healing of their bodies.

ACKNOWLEDGMENTS

The work presented in this dissertation is the result of concerted efforts made by multiple individuals who willingly sacrificed their time and energy to aid me in my pursuit of this degree. At this time, I would like to acknowledge and thank them for their efforts and contributions. Without them this work would not have been possible.

First and foremost, I would like to acknowledge Dr. William Richardson for welcoming me into his lab, for securing the funding for me to conduct the research presented here, and for mentoring and advising me as I grew into the scientific researcher I am today. I would not be where I am today without his involvement. He encouraged me to see God's hand at work as I studied the world around me.

Next, I would like to acknowledge Dr. Jeremy Mercuri for always encouraging me and supporting me while simultaneously challenging me and spurring me on to pursue excellence and to be the best I can be. His hands-on and personal approach to mentorship and his tireless work ethic both in and out of the lab serve as an example of the kind of tenacity and drive that I hope to exhibit in my own life.

Furthermore, I would also like to acknowledge Dr. Jiro Nagatomi and Dr. Delphine Dean for the support and instruction they have given me during my time at Clemson. Because of their willingness to teach me and give me opportunities to learn and grow through roles that enabled me to learn new skills and gain new experience, I can confidently step out of my role as a student and into the realm the professional world.

I would like to acknowledge my fellow student candidates for their comradery and assistance during my time here at Clemson. Specifically, I would like to acknowledge Kyle

Cannon for all the work and effort he put into to assisting me with the work presented in Chapter 2. His assistance was crucial to getting that work carried out in a timely and efficient manner. I would also like to acknowledge everyone in the Systems Mechanobiology Lab for their friendship and camaraderie. The bonds we share from our experiences together will last for a lifetime.

Finally, I would like to acknowledge my wife, Mary-Peyton, and my family, for their support and love throughout this process. Their words of encouragement helped to spur me on and overcome times of doubt and insecurity and their presence in my life assured me that I was never alone and inspired me to overcome any challenge that I faced.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Motivation

Several diseases and medical disabilities such as hypertension, lower back pain (LBP), type-2 diabetes mellitus, and certain cancers are regulated in part by mechanical forces [1-4]. Many of these conditions are leading causes of morbidity and disability globally. To illustrate the impact that these mechanically regulated diseases have on society, consider the prevalence and economic burden of just 2 cases: hypertension and LBP. It has been estimated that nearly 116 million adults in the U.S. alone suffer from hypertension [5] while other reports suggest that between 70 and 85% of adults worldwide will experience LBP during their lifetime [6].

Furthermore, a study from 2019 found that hypertension was a primary or contributing cause of death for 516,955 people in the United States [7] and another report found that between 13.1% and 20.3% of people who experience LBP have chronic symptoms [8]. A significant number of economic resources have been allocated to treating these pathological conditions, with hypertension accounting for nearly $131 billion in healthcare costs annually [9] and nearly an average of $8,900 per patient being spent on LBP treatments each year [10].

Given these high societal and economic costs, it is imperative to develop new therapies and interventions that will prevent, mitigate, and reverse these diseases and disabilities. The development of these therapies and interventions comes with increased scientific understanding and technological advancement. Experimental and computational platforms which can be used to test pathological hypotheses and fill in gaps in knowledge are vital to achieving this goal and

therefore there is a significant need to develop novel experimental and computational platforms dedicated to studying specific diseases and disabilities.

Intervertebral disc (IVD) degeneration is thought to be the predominant cause of chronic LBP [11]. IVD degeneration commonly leads to defects in the annulus fibrosus (AF) that forms the outer shell of the IVD and can result in IVD herniation [12]. If the IVD herniation is severe enough, it is typically treated by performing a discectomy to remove portions of the prolapsed IVD [13]. When this happens, the probability of re-herniation is high due to ruptures in the IVD AF. Several companies have investigated different possible devices for sealing the IVD following a tear in the AF; however, these devices are still lacking as they only reduce patient re-herniation by 52% [14,15]. Furthermore, these devices are limited in their ability to repair the IVD because of their inability to support tissue regeneration. One strategy to improve the success rate of these devices that is currently being investigated is the incorporation of stromal cells into the device to promote tissue regeneration at the site of the rupture [16]. However, the optimal cell source for this application is currently unknown. We propose to utilize a commercially available experimental platform to evaluate the effector responses of different stromal cells in a monolayer culture and develop a novel experimental platform capable of evaluating these effector responses from different stromal cells in 3-D biological scaffolds to collect data that could help answer this question.

Hypertension is a disease that can be fatal by increasing the risk of catastrophic heart failure (HF) [17]. Fortunately, hypertension can be managed through preventative measures such as lifestyle changes and medications that lower blood pressure (BP) [18]. Identifying patients who are at-risk for hypertension prior to becoming hypertensive would be beneficial as measures could be taken to prevent or slow the occurrence of the disease [19]. Several other studies have sought to

develop computational models that could be used to predict the onset of hypertension to aid physicians in clinical decision-making [20-23]. Machine learning (ML) computational models have become popular for this application due to their ability to create complex and highly accurate models from large clinical datasets without the need for a complete mechanistic understanding of the underlying biology. However, there are currently several drawbacks to the ML models that have currently been developed [24,25]. First, many of the models are limited in the number of biologically relevant features that are being examined and second, ML models lack expandability, which often is one of the main issues that prevents them from being trusted and adopted by physicians in clinical practice. A ML model that could indicate which factors were most relevant in making a prediction while utilizing a more detailed array of biologically relevant features would give clinicians greater insight into how to treat their patients more effectively and could be used to expedite the adoption of computational models into clinical practice.

1.2 Organization

This dissertation is divided into five chapters. Chapter 1 describes the motivation behind this work, presents an outline of how this dissertation is organized, outlines the overall aims for this work, and provides a general review of each of fundamental topics presented herein. These topics include mesenchymal stromal cell (MSC) mechanobiology, bioreactors, and machine learning models. More extensive background information will be provided within the following chapters as it relates to the topic of the study discussed in that particular chapter. Chapter 2 presents a study entitled *"Mechanical Stimulation Alters Mesenchymal Stromal Cell Survivability and Cytokine Production in Low pH Culture Conditions: Implications for*

Intervertebral Disc Repair". This study examines how the biological behavior of MSCs sourced from 3 different tissues changes when exposed to low pH or a combination of pH and cyclic tensile strain. Chapter 3 presents a study entitled *"Design and Testing of a Multi-Well Tissue Stretching Bioreactor"*. This study reports the design details and validation procedures that were implemented in the development and construction of a novel 3-D printed bioreactor capable of applying tensile strain to multiple cell-seeded biological scaffolds concurrently. Chapter 4 presents a study entitled *"Development of a Machine Learning Model for Detecting Masked Hypertension in Young, Apparently Healthy Adults from Low-and-Middle Income Countries"*. This study describes the construction of a machine learning (ML) model that is capable of assessing several easy-to-obtain clinical measurements and calculating a patient's risk for masked hypertension (MHT). Finally, chapter 5 outlines the overall conclusions from the studies presented here and offers recommendations for future work.

1.3 Study Aims

The overarching objective of this dissertation was to further the scientific knowledge of how mechanical forces influence biological behavior, maintain tissue homeostasis, and regulate the progression of disease. This objective was achieved by expanding the field of systems mechanobiology and by developing novel experimental and computational platforms for studying the complex interactions between mechanical forces and biological behavior. The findings of this research can be broadly categorized into 3 distinct aims.

<u>Aim 1: Investigate the Impact of Mechanical Stimulation on MSC Behavior in Normal and Low pH Cell Culture Conditions.</u> The first aim of this dissertation was to expand the field of systems

mechanobiology by conducting a study that resulted in new knowledge and insights regarding how mechanical forces could alter MSC behavior in the context of repairing and regenerating the degenerate IVD and how MSC behavior would vary when sourced from different tissues. The work presented in Chapter 2 demonstrated that mechanical stimulation does alter MSC behavior in a low pH microenvironment. This work also demonstrated that the MSC response to biochemical and mechanical conditions varies depending on the MSC type.

Aim 2. Develop and Validate a Long-Term Bioreactor Capable of Mechanically Stimulating Cells Seeded on Collagenous Tissue. The second aim of this study was to design a novel experimental platform for studying systems mechanobiology. This was accomplished through the development of a novel multi-well 3-D printed bioreactor system that could apply mechanical stimulation to a cell-seeded biological scaffold. Furthermore, a methodology of imaging the cells and tissues in the bioreactor to assess the level of strain experienced by the tissue and the resident cells during an experiment was also developed. The work in Chapter 3 demonstrated that DIC strain analysis could be implemented during an experiment to evaluate tissue strains without having to disrupt or terminate the experiment. Fluorescent image analysis was used to evaluate cell strains during testing. Cytotoxicity assays and long-term viability tests were used to demonstrate that the bioreactor could be utilized for long-term cell culture experiments.

Aim 3. Develop a Machine Learning Model Capable of Predicting Cardiac Morphology and Health as a Function of Physical Activity and Biochemical Profiles. The third aim of this study was to develop a novel computational platform for analyzing patient data and identifying

potential relationships that link clinical measures of *in vitro* mechanical forces and biochemicals to MHT with the purpose of predicting a patient's risk for that disease. Machine learning techniques were employed to develop a computational model that was capable of predicting a patient's risk for MHT in a young, apparently healthy population from a LMIC using several clinical measures that could be easily obtained during a single outpatient visit. The work presented in Chapter 4 demonstrate that a functional ML model can be developed using only a few clinical measurements that can be easily obtained from a single outpatient visit and that the model presented herein would function better than the current "rule-of-thumb" method for assessing a patient's risk for MHT without having to immediate resort to evaluating the patient with ABPM.

1.4 Background

1.4.1 MSC Mechanobiology

Stromal cell therapy and tissue engineering strategies have the potential to revolutionize treatments for injury, disease, and age-related conditions [26]. However, this potential has yet to be realized as our understanding of stromal cell biology is limited. While many studies have been published detailing the role of biochemical cues and signaling events in directing stromal cell behavior, recent work in the field has shown that mechanical stimulation also plays a significant role in regulating stromal cell function [27]. Previous reports have been published detailing how mechanical loading can affect cell behavior through the assembly and disassembly of focal adhesions, the rearrangement of cytoskeletal proteins, the activation of stretch ion channels and G protein-coupled receptors, and conformational changes in the nucleus [28]. For MSCs in particular, mechanical signaling has been shown to play a significant role in influencing MSC

fate decisions. For example, compressive forces have been shown to induce chondrogenic differentiation while tensile forces have been shown to induce osteogenic differentiation [29,30]. Other studies have shown that mechanical forces can impact MSC apoptosis and proliferation [31,32].

1.4.2 Bioreactors

Bioreactors can be broadly defined as "devices in which biological and/or biochemical processes develop under closely monitored and tightly controlled environmental and operating conditions (e.g., pH, temperature, pressure, nutrient supply, and waste removal)" [33]. These devices are invaluable tools for developing tissue engineering strategies as they enable the application of biochemical and mechanical stimulation to cells and tissues *in vitro* within a tightly regulated environment. The ability to apply these stimuli in a controlled manner allows for the fundamental mechanisms of cell function under normal and pathological conditions to be assessed.

There are several factors that should be taken into consideration when designing a bioreactor. First, a bioreactor must be able to establish environmental conditions that are favorable for cell survival. This includes maintaining the cells at a temperature of 37 °C with a high relative humidity, and a 5% CO_2 gas mixture. Another requirement of a bioreactor system is the ability to provide the cell culture with nutrients, often in the form of a specially formulated media that can provide the necessary metabolites, oxygen, growth factors, and other components necessary for maintaining cell viability. The bioreactor should also be made of materials that are not cytotoxic and will not negatively impact cell viability.

The second design factor that must be considered when creating a bioreactor is the type of desired stimulation that the bioreactor should be applying to the cells. This can include a variety of mechanical and physical stimuli such as tension, compression, torsion, and shear. It can also involve electromagnetic stimulation through the application of magnetic fields or alterations in biochemical signaling in the form of different cytokines or alterations in environmental conditions. Occasionally, bioreactors will combine multiple types of stimuli in order to fulfill the requirements of an experimental protocol.

Another important consideration when designing a bioreactor is the parameters that need to be monitored. Again, the parameters which should be evaluated depend on the requirements of the experimental protocol, but in general, a system should have sensors to monitor the environmental conditions of the system and/or the physical properties of the cells and tissue in the system.

Other considerations related to bioreactor design include the ease of assembling the bioreactor and loading the cells and/or tissue, the ability to test multiple samples at once, and the ease of scaling-up the system.

1.4.3 Machine Learning Models

In 1959, Arthur Samuel defined machine learning (ML) as a subfield of artificial intelligence (AI) that focuses on developing algorithms that "enable computers to learn without explicitly being programmed" [34]. Using this approach, computers can develop highly accurate predictive models using only historical data without any a priori knowledge regarding the causal relationships between the different variables within the dataset. This is accomplished through the

use of mathematical, computational, and statistical methodologies that enable underlying patterns in the data to be identified, extracted, and employed in a model framework.

The ML modeling approach has several advantages over more mechanistic modeling approaches which require that the rules of the model be explicitly defined. First, models created using a ML approach are dynamic in nature, operating off of trends observed in the data using statistical analysis rather than having a static model which operates on a single set of hardcoded rules. This allows the model to update itself over time and correct for mistakes in the model without requiring any input from the model user. Second, ML models can detect and incorporate relationships between variables in the data that were previously unknown and therefore may not be accounted for in a mechanistic model. This can be useful when little is known about the variables included in the dataset. Finally, ML models can handle a large number of variables because of this method's ability to learn rules for how each variable behaves from the data directly; whereas more mechanistic model tend to be more simplistic and operate using fewer variables because the relationships for each variable in the model need to be comprehensively defined in order to successfully incorporate them into the model.

One drawback to using ML methods to develop predictive algorithms is that they are completely reliant on the quality and quantity of the dataset that they learn on, making them highly susceptible to noise or false signals. Furthermore, it is typically difficult to determine how a ML-based model makes a prediction, giving ML-based modeling a reputation for being a "black-box" approach to modeling. Thus, while ML-based modeling is not necessarily the best universal approach to computational modeling, it does have several advantageous that make it an appropriate choice for particular applications.

1.5 References:

1. Diaz KM, Shimbo D. Physical activity and the prevention of hypertension. *Curr Hypertens Rep*. 2013;15(6):659-668.

2. Henchoz Y, So AK. Exercise and nonspecific low back pain: A literature review. *Joint Bone Spine*. 2008;75(5):533-539.

3. Burr JF, Rowan CP, Jamnik VK, Riddell MC. The role of physical activity in type 2 diabetes prevention: Physiological and practical perspectives. *The Physician and sportsmedicine*. 2010;38(1):72-82.

4. Wang Q, Zhou W. Roles and molecular mechanisms of physical exercise in cancer prevention and treatment. *Journal of Sport and Health Science*. 2021;10(2):201-210.

5. Centers for Disease Control and Prevention, (CDC). Hypertension cascade: Hypertension prevalence, treatment and control estimates among US adults aged 18 years and older applying the criteria from the american college of cardiology and american heart association's 2017 hypertension Guideline—NHANES 2013–2016. *Atlanta, GA: US Department of Health and Human Services*. 2019.

6. Andersson GB. Epidemiological features of chronic low-back pain. *The lancet*. 1999;354(9178):581-585.

7. Centers for Disease Control and Prevention, National Center for Health Statistics. About multiple cause of death, 1999-2019. . Updated 2019. Accessed February 1, 2021.

8. Meucci RD, Fassa AG, Faria NMX. Prevalence of chronic low back pain: Systematic review. *Rev Saude Publica*. 2015;49:73.

9. Kirkland EB, Heincelman M, Bishu KG, et al. Trends in healthcare expenditures among US adults with hypertension: National estimates, 2003–2014. *Journal of the American Heart Association*. 2018;7(11):e008731.

10. Geurts JW, Willems PC, Kallewaard J, van Kleef M, Dirksen C. The impact of chronic discogenic low back pain: Costs and patients' burden. *Pain Research and Management*. 2018;2018.

11. Zheng C, Chen J. Disc degeneration implies low back pain. *Theoretical Biology and Medical Modelling*. 2015;12(1):1-10.

12. Martin MD, Boxell CM, Malone DG. Pathophysiology of lumbar disc degeneration: A review of the literature. *Neurosurgical focus*. 2002;13(2):1-6.

13. Malter AD, Larson EB, Urban N, Deyo RA. Cost-effectiveness of lumbar discectomy for the treatment of herniated intervertebral disc. *Spine*. 1996;21(9):1048-1054.

14. Bailey A, Araghi A, Blumenthal S, Huffmon GV, Anular Repair Clinical Study Group. Prospective, multicenter, randomized, controlled study of anular repair in lumbar discectomy: Two-year follow-up. *Spine*. 2013;38(14):1161-1169.

15. Lequin MB, Barth M, Thomė C, Bouma GJ. Primary limited lumbar discectomy with an annulus closure device: One-year clinical and radiographic results from a prospective, multi-center study. *Korean Journal of Spine*. 2012;9(4):340.

16. Borem R, Madeline A, Theos C, et al. Angle-ply scaffold supports annulus fibrosus matrix expression and remodeling by mesenchymal stromal and annulus fibrosus cells. *Journal of Biomedical Materials Research Part B: Applied Biomaterials*. 2021.

17. Diamond JA, Phillips RA. Hypertensive heart disease. *Hypertension research*. 2005;28(3):191-202.

18. Gabb GM, Mangoni AA, Anderson CS, et al. Guideline for the diagnosis and management of hypertension in adults—2016. *Med J Aust*. 2016;205(2):85-89.

19. Krittanawong C, Bomback AS, Baber U, Bangalore S, Messerli FH, Wilson Tang WH. Future direction for using artificial intelligence to predict and manage hypertension. *Curr Hypertens Rep*. 2018;20(9):1-16.

20. Nour M, Polat K. Automatic classification of hypertension types based on personal features by machine learning algorithms. *Mathematical Problems in Engineering*. 2020;2020.

21. Golino HF, Amaral, Liliany Souza de Brito, Duarte SFP, et al. Predicting increased blood pressure using machine learning. *Journal of obesity*. 2014;2014.

22. Lopez-Martinez F, Schwarcz A, Núñez-Valdez ER, Garcia-Diaz V. Machine learning classification analysis for a hypertensive population as a function of several risk factors. *Expert Syst Appl*. 2018;110:206-215.

23. AlKaabi LA, Ahmed LS, Al Attiyah MF, Abdel-Rahman ME. Predicting hypertension using machine learning: Findings from qatar biobank study. *Plos one*. 2020;15(10):e0240370.

24. Price WN. Big data and black-box medical algorithms. *Science translational medicine*. 2018;10(471):eaao5333.

25. Kelly CJ, Karthikesalingam A, Suleyman M, Corrado G, King D. Key challenges for delivering clinical impact with artificial intelligence. *BMC medicine*. 2019;17(1):1-9.

26. Castillo, Alesha B., and Christopher R. Jacobs. "Mesenchymal stem cell mechanobiology." *Current osteoporosis reports* 8 (2010): 98-104.

27. McKayed, Katey K., et al. "Mesenchymal stem cells: role of mechanical strain in promoting apoptosis and differentiation." *Stem Cells and Cancer Stem Cells, Volume 3: Stem Cells and Cancer Stem Cells, Therapeutic Applications in Disease and Injury: Volume 3* (2012): 199-206.

28. Castillo, Alesha B., and Christopher R. Jacobs. "Skeletal mechanobiology." *Mechanobiology Handbook, Second Edition*. CRC Press, 2018. 281-308.

29. Haudenschild, Anne K., et al. "Pressure and distortion regulate human mesenchymal stem cell gene expression." *Annals of biomedical engineering* 37 (2009): 492-502.

30. Sen, Buer, et al. "Mechanical strain inhibits adipogenesis in mesenchymal stem cells by stimulating a durable β-catenin signal." *Endocrinology* 149.12 (2008): 6065-6075.

31. Kearney, E. M., P. J. Prendergast, and V. A. Campbell. "Mechanisms of strain-mediated mesenchymal stem cell apoptosis." (2008): 061004.

32. Ghazanfari, Samane, Mohammad Tafazzoli-Shadpour, and Mohammad Ali Shokrgozar. "Effects of cyclic stretch on proliferation of mesenchymal stem cells and their differentiation to smooth muscle cells." *Biochemical and biophysical research communications* 388.3 (2009): 601-605.

33. Martin, Ivan, David Wendt, and Michael Heberer. "The role of bioreactors in tissue engineering." *TRENDS in Biotechnology* 22.2 (2004): 80-86.

34. Mahesh, Batta. "Machine learning algorithms-a review." *International Journal of Science and Research (IJSR).[Internet]* 9.1 (2020): 381-386.

CHAPTER 2

Mechanical Stimulation Alters Mesenchymal Stromal Cell Survivability and Cytokine Production in Low pH Culture Conditions: Implications for Intervertebral Disc Repair

2.1 Introduction

Intervertebral disc (IVD) degeneration is an increasingly prevalent condition that has been linked to low back pain and IVD herniation [1,2]. IVD herniation occurs when the nucleus pulposus (NP) pushes through the annulus fibrosus (AF), resulting in the protrusion of IVD material. Despite having a relatively high incidence rate of 1-3% in the global population, treatment options for IVD herniations are mostly limited to symptom management or focus primarily on the late stage of the disease [3,4]. Microdiscectomies are surgical interventions performed to alleviate pain by removing part of the damaged IVD that is impinging on nerve roots. However, this procedure leaves the AF to heal on its own and can result in further degeneration and fibrosis [5,6]. Furthermore, it has been estimated that up to 25% of patients who undergo microdiscectomies require additional care due to reherniation [7]. Annular repair products, including Barricade™ have been shown to reduce the incidence of reherniation and reoperation rates by sealing tears in the AF. However, these products do not promote integration with or regeneration of AF tissue [8].

These current limitations have prompted researchers to investigate novel regenerative strategies for repairing and regenerating the AF of a herniated IVD. These include utilizing the delivery of cells, growth factors, scaffolds, and gene therapy [2,9-12]. Cell therapy utilizing mesenchymal stromal cells (MSCs) is one promising option for this application, as clinical trials have shown that intradiscal injection of MSCs can reduce pain 2 years post operation in patients

undergoing discectomies [13,14], and animal models demonstrate the possibility that MSC injection could help alleviate pain by preventing neovascularization and innervation of the IVD [15]. Furthermore, a recent study by Privu et al., demonstrated the potential of using scaffolds seeded with BM-MSCs to repair ruptured AF tissue [16]. However, use of MSCs for IVD repair is still in its infancy and there remain gaps in our understanding of how to properly employ MSCs in strategies to repair and regenerate the AF following discectomy. Some of these gaps in knowledge include improving our understanding of how to increase the low survival rates of MSCs post implantation and elucidation of how MSCs respond to the harsh microenvironment of the degenerate IVD [9].

Briefly, the microenvironment of the degenerate IVD can be characterized by low pH (6.2- 6.8), low nutrition (8mM glucose), low oxygen (1- 5%), high osmolarity (250mOsm-370mOsm), and complex mechanical loading (-5 to -8% axial strain and 8 to 10% max 3D shear strain in the posterior region of the AF; -4 to -6% axial strain and 7 to 9% max 3-D shear strain in the NP) [9,12,17-20]. This combination of factors results in a microenvironment that can be challenging for both endogenous IVD cells and implanted MSCs. This has been confirmed by previous studies which have shown that MSCs survival is impacted negatively by this microenvironment [11,21,22]. Several research groups have conducted studies to demonstrate how the different microenvironmental stimuli impact MSCs [9]. For example, research published by Liang et al., and Weurtz et al., examined the effects that low pH, low osmolarity, low glucose, and different combinations of each factor had on human adipose derived MSCs (AD-MSCs) [11,12] while other groups such as Li et al., examined the impact of hypoxia on rat AD-MSCs and NP derived MSCs (NP-MSCs) [13]. These studies suggest that low pH, low osmolarity, and hypoxia both had a negative impact on matrix biosynthesis and cell proliferation while low glucose had a

small positive impact on these same metrics. The combination of low pH, low osmolarity, and low glucose resulted in an overall detrimental effect as well. Due to the observation that acidity increases during IVD degeneration while osmolarity decreases, it was suggested that pH in particular, might be the major limiting factor for MSC-based IVD repair strategies[12]. It is important to consider the pH of the IVD when developing a AF repair strategy as the lactic acid in the NP could contaminate the AF tissue during herniation, resulting in an acidic microenvironment that could negatively impact any cells involved in the therapy.

While the majority of these studies have investigated the impact of different biochemical factors on MSC behavior; previous research has shown that it is important to also consider the MSC response to mechanical loading in the context of the AF [23]. In a recent review paper by Vadalà et al., it was reported that mechanical loading played a crucial role in regulating IVD cell activity and metabolism. Thus, in the context of repairing and regenerating the AF with an MSC-based therapy, it is important to not only consider the biochemical factors, but also the relevant mechanical conditions. To our knowledge, the effect of tensile strain in conjunction with low pH on MSC behavior has not yet been explored, particularly in the context of repairing and regenerating the AF. Wuertz et al., did investigate the relationship between osmolarity and cyclic tensile strain on IVD cells, but did not extend their research to investigate the relationship between low pH and tensile strain on MSCs [14]. Furthermore, recent research has demonstrated that native AF cells can react both positively and negatively to mechanical stimulation, suggesting that mechanical stimulation plays an important role in AF repair and regeneration [14,24] Thus, it is important to understand the combined effects of low pH and mechanical stimuli on MSC's.

It is also important to note that researchers have demonstrated that MSC response can vary based on the tissue origins of these cells [15,25]. For example, in a study by Kozlowska et al., researchers found that while MSCs from different tissues may exhibit some common characteristics, their biological behavior differed depending on their tissue of origin [25]. Borem et al., observed different effector responses between AD-MSCs and amnion derived MSCs (AM-MSCs) when cultured conditions designed to simulate the inflammatory microenvironment of the degenerate IVD [15]. Therefore, MSC type must also be considered when developing an MSC-based cellular therapy of IVD regeneration.

Taken together, the aim of this study was to investigate how the combined effects of low pH and simplified uniaxial tensile strain similar to that observed in the degenerate AF, impacted the biological behavior of human MSCs sourced from different tissues. It was hypothesized that the introduction of tensile strain would reduce the negative impact of low pH on MSC survivability, and that the degree of this response would vary depending on the MSC tissue source. To test this, human MSCs derived from bone marrow- (BM), adipose- (AD) and amnion- (AM) tissues were cultured in low pH in a tensile bioreactor and measures of cell survival and cytokine release profiles were completed. Results from these studies suggest that mechanical stimulation impacts MSC biological behavior in low pH conditions and that the response is dependent on the MSC type.

2.2 Materials and Methods

2.2.1 MSC Expansion

Human AD-MSCs and BM-MSCs were purchased from ATCC. Human Amnion derived-MSCs (AMMSC) were isolated from human placentas in accordance with previously published methods [26]. Samples were obtained from consenting patients immediately after delivery via elective cesarean sections of full-term babies under an Institutional Review Board-approved protocol (Prisma Health Upstate: Pro00031185). All MSCs were expanded in standard culture conditions (37 C, 5% CO2) until passage 3 (P3) using cell culture media consisting of Dulbecco's modified Eagle's medium (DMEM), 10% fetal bovine serum (FBS), and 1% antibiotic/antimitotic (Ab/Am).

2.2.2 Experimental Setup

Deformable 16-well silicone culture plates (CellScale) were coated with. 100 uL of a 0.05% fibronectin solution (Advanced BioMatrix) for 2 hours at 37 C and 5% CO2. After coating, 200 uL of cell culture media with MSCs (~5,000 cells/cm2) were dispensed into each well and incubated overnight to allow for MSCs attachment. Live cell nuclear stain (NucBlue, Thermofisher) was added to each well and incubated at 37 C and 5% CO2. Fluorescent images of cell nuclei were obtained for each well to get an initial cell count prior to assigning each culture plate to its experimental condition. Plates were then assigned to 1 of 3 experimental groups: (1) a mechanically static group with normal pH (7.4) cell culture media (Static-Normal), (2) a mechanically static group with low pH (6.5) cell culture media (Static-Low), and (3) a

mechanically dynamic group with low pH cell culture media (Dynamic-Low). These groups were chosen specifically to evaluate the relative impact of both pH and mechanical stimulation on MSC behavior. Silicone well plates assigned to the mechanically static groups were placed into 60 mm cell culture plates (Sigma-Aldrich) while silicon well plates assigned to mechanically dynamic groups were loaded into a device to apply uniaxial tension to the silicone culture plates (MCFX-2, CellScale). Once each plate was assigned to either the dynamic or static grouping, normal pH cell culture media was added to 8 wells of each plate and low pH cell culture media was added to the remaining 8 wells. This resulted in 8 wells per group per experiment for each MSC type. The experimental setup is summarized below in Figure 2.1a. Normal pH media consisted of standard Dulbecco's modified Eagle's medium (DMEM) with 2.13 g/L of sodium bicarbonate, 10% fetal bovine serum (FBS), and 1% antibiotic/antimitotic (Ab/Am). Low pH media consisted of standard Dulbecco's modified Eagle's medium (DMEM) with 0.2g/L of sodium bicarbonate, 10% fetal bovine serum (FBS), and 1% antibiotic/antimitotic (Ab/Am).

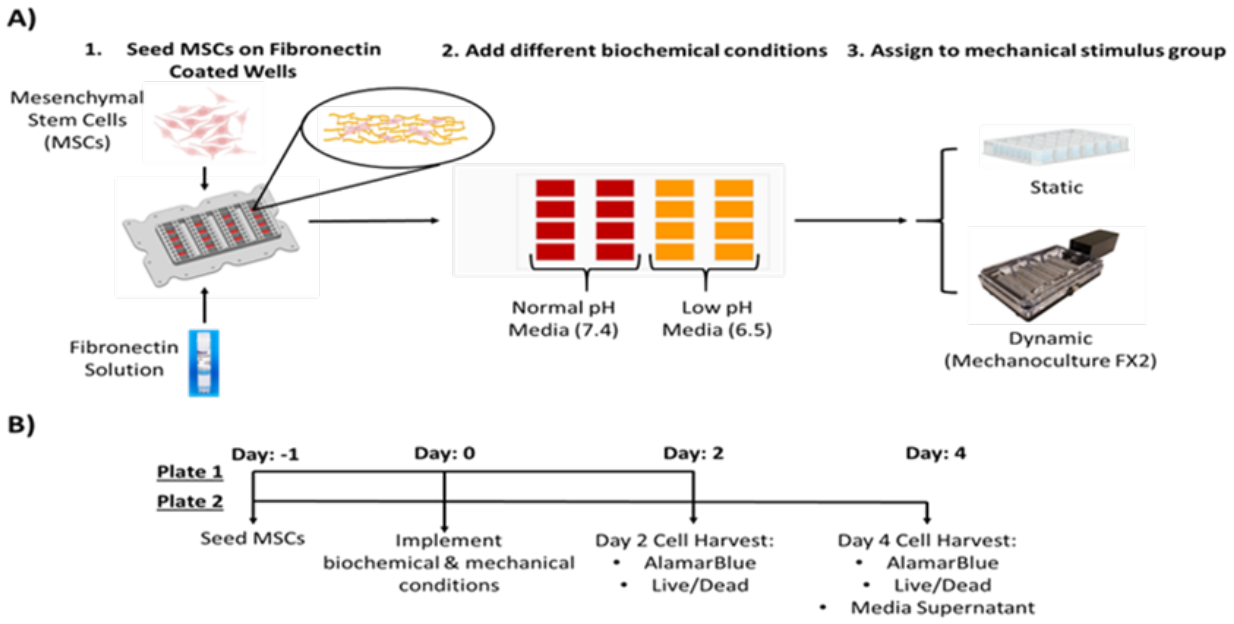

Figure 2.1. Experimental setup and protocol. A) For each experimental group, MSCs were seeded into individual wells of a fibronectin coated 16-well silicone plate. The wells of each plate contained either low or normal pH media (8 wells each) prior to placing them into a Mechanoculture FX2 system for dynamic stimulation or a 60 mm culture dish for static culture. **B)** For both static and dynamic conditions, there were 2 identical plates. The first plate was cultured for 2 days, and the second plate was cultured for 4 days. At each timepoint, data was collected to assess cell metabolic activity, viability, and cytokine production.

2.2.3 Cell Culture and Mechanical Stimulation Protocol

Cells in the dynamic group with low pH were stretched to ~6% uniaxial strain at a frequency of 0.1 Hz for 16 hours and then held statically at ~1% uniaxial strain for 8 hours to simulate the mechanical loading that cells in the AF may experience *in vivo* over the course of a normal 24-hour day. The strain magnitudes and frequencies were chosen to replicate physiologically relevant conditions in the AF [24,27]. Two sets of experiments were done: 1) MSCs cultured for 2 days and 2) MSCs being cultured for 4 days. For the 2-day experiments, cells were cultured in their respective experimental conditions for 48 hours and then 100 uL of cell supernatant was collected from each well and stored at -80 °C. For each experimental group, five wells were used to assess cell metabolic activity (CyQUANT MTT assay, Thermofisher

Scientific). The remaining three wells were used to assess cell number, viability, and morphology via fluorescent imaging using live/dead staining (2 uL/mL calcein, 1uL/mL ethidium homodimer, one drop NucBlue, Thermofisher Scientific). For the 4-day experiments, cells were cultured in their respective experimental conditions for 96 hours with a media change at 48 hours. Cell metabolic activity, count, viability, and morphology was then assessed using the same protocol from the 2-day experiments. The experimental protocol is summarized in Figure 2.1b.

2.2.4 Cell Metabolic Activity

An MTT assay was performed at 2- and 4-days to assess the metabolic activity of each MSC type under each experimental condition and served as a general indicator of cell growth and survival [28]. The assay was performed (n=5/condition/time-point/MSC type) in accordance with manufacturer's instructions. Absorbance was measured at 540 nm and blanked to control samples. Results are expressed in relative absorbance units (RAU). Data is presented as mean RAU ± standard deviation.

2.2.5 Cell Viability and Number

A live/dead assay (n=3//condition/time-point/MSC type) was performed in accordance with manufacturer instructions to assess the viability of each MSC type under each experimental condition and served as a direct indicator of cell survival. Wells were imaged at a 10X total magnification using a (insert microscope name / model) fluorescent microscope. Subsequently an open-source automated image processing and cell segmentation/counting software (CellProfiler,

Broad Institute) [29] was used to identify and count the number of live and dead cells to estimate percent cell viability (# Live cells/# Total cells*100). A machine learning segmentation plugin (Cellpose 2.0) [30] was used to generate segmentations of the MSC membranes using a neural network model trained specifically on fluorescent images of cell membranes (Cyto model). Results are presented as mean percent cell viability ± standard deviation and total cell count ± standard deviation.

2.2.6 Cytokine Analysis

A cytokine array (Human Cytokine Array C5, Raybiotech) was used to semi-quantitatively compare the secretion profile of 80 different growth factors and inflammatory cytokines for each MSC type under the different experimental conditions. Five individual supernatants from each group were pooled together to get a single collection of cell culture supernatants per group. Subsequently, the pooled samples were diluted with 0.5 mL of blocking buffer and allowed to incubate overnight at 4 C and samples were prepared in accordance with the manufacturer's instructions and were imaged using a chemiluminescence reader. For each membrane array, chemiluminescence measurements were quantified using a custom written MATLAB script prior to subtracting background signal from control media samples. Data is presented as relative densitometric units (RDU) normalized to the control (fresh media) and presented as 1) the percent change of cytokine concentrations of culture conditions compared to the control and 2) the precent change of cytokine concentrations from static low to dynamic low culture conditions.

2.2.7 Statistical Analysis

Statistical analysis and graph preparation was completed using GraphPad Prism 9 software. All data is presented as mean ± standard deviation with the exception of the cytokine data. A 3-way ANOVA followed by Tukey's multiple comparisons test was used to evaluate the statistical significance of applicable data. A p-value ≤ 0.05 was considered significant.

2.3 Results:

2.3.1 Effects of pH and Mechanical Stimulation on MSC Metabolic Activity

The metabolic activity of MSCs cultured in low pH conditions were significantly lower than the metabolic activity of MSCs cultured in normal pH conditions at both the day 2 and day 4 time point (Figure 2.2). On day 2, AD-MSC and AM-MSC metabolic activity was increased in low pH with mechanical stimulation conditions compared to the metabolic activity of AD-MSCs (day 2, $p = 0.0114$) and AM-MSCs (day 2, $p = <0.001$) in the low pH conditions while BM-MSC (day 2, $p = 0.0348$) metabolic activity was further reduced (Figure 2.2). This trend continued on day 4 but was not significantly different.

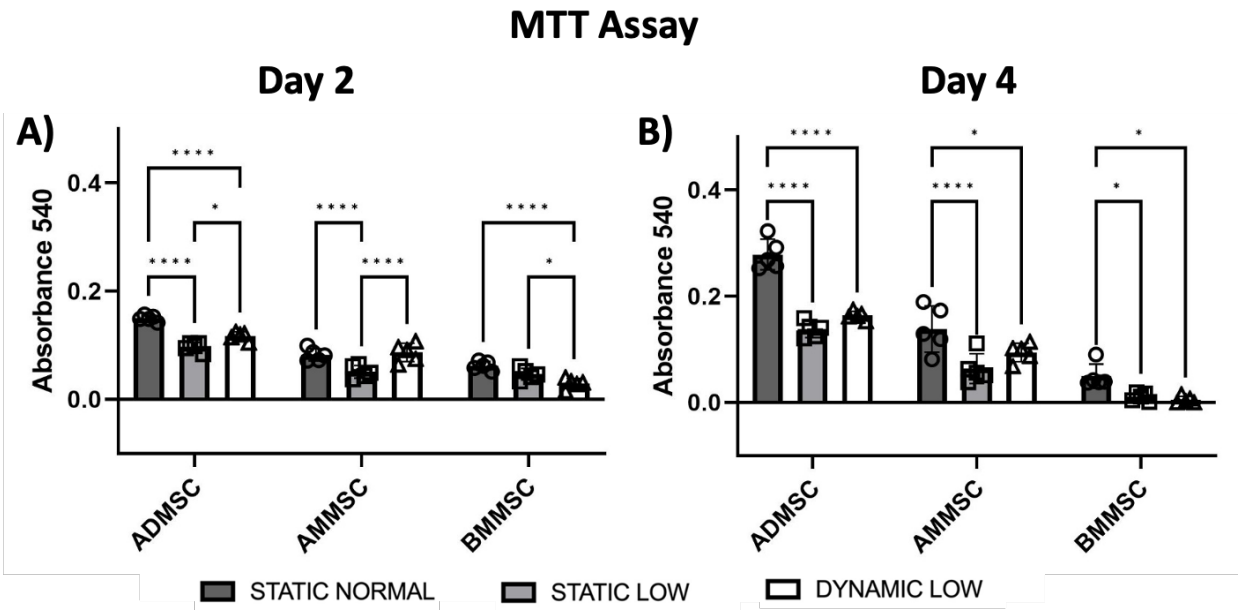

Figure 2.2. Assessment of MSC metabolism. A-B) MTT assay results for day 2 and day 4 time points indicating significant differences between experimental conditions (static-normal: pH 7.4, static-low: pH 6.5, dynamic-low: 6% strain for 16 hrs at 0.1 Hz then 1% strain for 8 hrs, pH 6.5) for different MSC types (AD-MSC, AM-MSC, BM-MSC). * indicates $p<0.05$, ** indicates $p<0.01$, *** indicates $p<0.005$, and **** indicates $p<0.001$.

2.3.2 Effects of pH and Mechanical Stimulation on MSC Viability

The viability of AD-MSCs and AM-MSCs cultured in low pH conditions were not statistically significant to the viability of AD-MSCs and AM-MSCs cultured in normal pH conditions at both the day 2 and day 4 time point (Figure 2.3), however BM-MSC viability was significantly decreased in low pH culture compared to normal pH culture conditions (day 2, $p = <0.0001$; day 4, $p = <0.0001$). AD-MSC and AM-MSC viability was also similar in low pH with mechanical stimulation conditions compared to the viability of AD-MSCs and AM-MSCs in the low pH conditions while BM-MSC viability was significantly reduced on day 2 (day 2, $p = 0.0055$) but then was significantly increased on day 4 (day 4, $p = <0.0001$) (Figure 2.3).

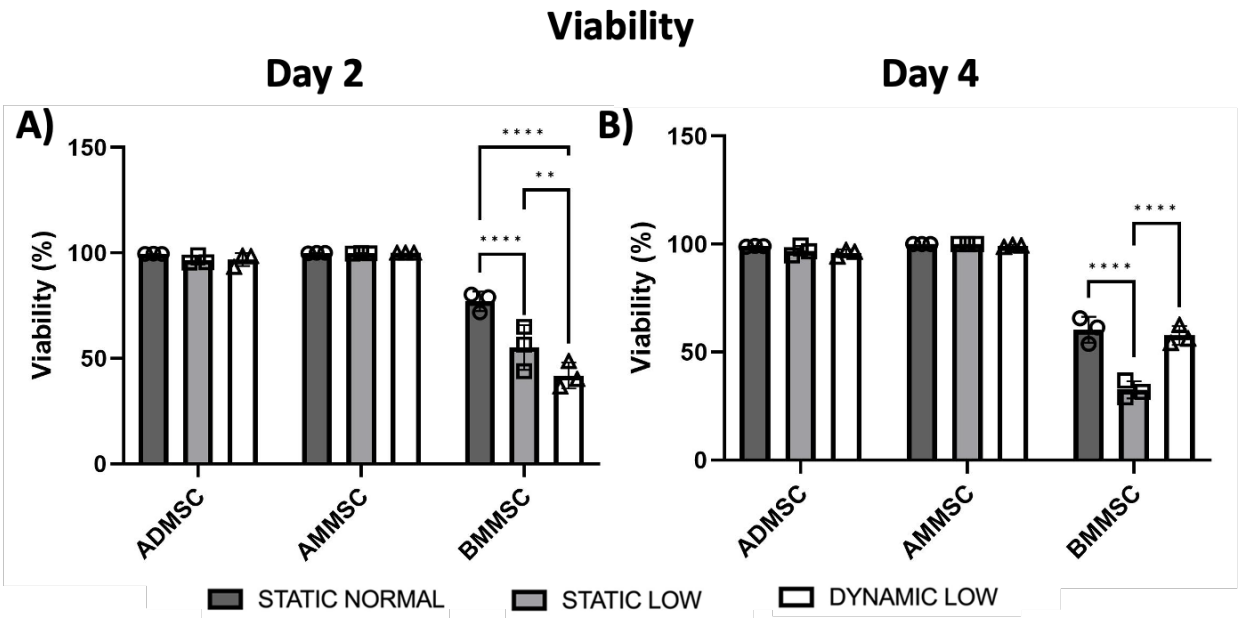

Figure 2.3. Assessment of cell survivability. A-B) Cell viability results for days 2 and 4 time points clustered by MSC type and arranged to highlight statistical changes in the number of cells counted between normal static, low static, and low dynamic culture conditions. * indicates $p<0.05$, ** indicates $p<0.01$, *** indicates $p<0.005$, and **** indicates $p<0.001$.

2.3.3 Effects of pH and Mechanical Stimulation on MSC Number

The number of MSCs cultured in low pH conditions was not significantly changed compared to normal pH conditions at both the day 2 and day 4 time point (Figure 2.4). AD-MSCs demonstrated a non-significant increase in cell numbers in low pH conditions on both day 2 and day 4 while AM-MSCs demonstrated a non-significant increase in cell numbers in low pH conditions on day 2 but this decreased in low pH conditions on day 4. BM-MSCs demonstrated a non-significant decrease in cell number in low pH conditions on both day 2 and day 4. AD-MSCs demonstrated a non-significant decrease in cell number on day 2 and day 4 in low pH with mechanical stimulation conditions compared to low pH conditions (Figure 2.4). AM-MSC cell number was significantly increased in low pH conditions with mechanical stimulation on both day 2 and day 4 (day 2, $p = 0.0039$; day 4, $p = 0.0087$) compared to static low pH culture

conditions while BM-MSC cell number experienced a non-significant increase in cell number in low pH conditions with mechanical stimulation on day 4 when compared to cell number in low pH culture conditions (Figure 2.4).

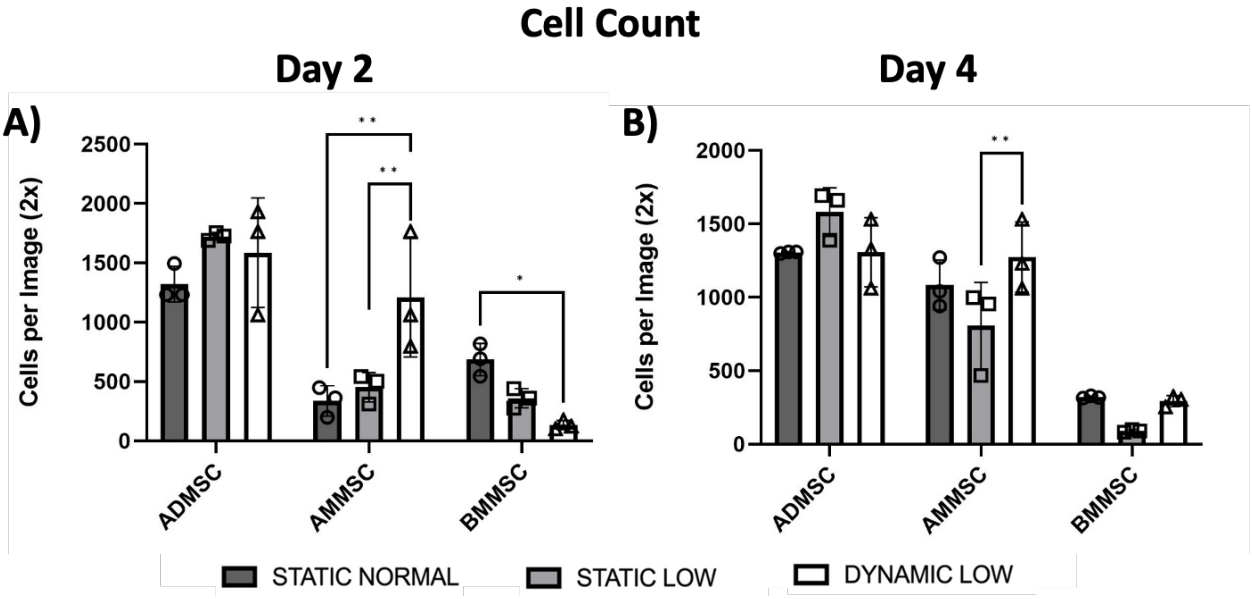

Figure 2.4. Assessment of cell number. A-B) Cell number for days 2 and 4 depicting the average number of individual live cells identified. Results are clustered by experimental condition and arranged to highlight statistical changes in the number of cells counted between AD-MSC, AM-MSC, and BM-MSC cell types. * indicates $p<0.05$, ** indicates $p<0.01$, *** indicates $p<0.005$, and **** indicates $p<0.001$.

2.3.4 Effects of pH and Mechanical Stimulation on MSC Cytokine Production

Relative cytokine concentrations were reduced in AD-MSCs and AM-MSCs in low pH culture conditions compared to normal pH culture conditions while BM-MSC relative cytokine concentrations remained largely unaffected in low pH culture conditions. However, slight increases in the relative concentration of TGF-B2, LIGHT, IGFBP-2, IL-1α, IL-2, IL-12, and IL-16 were observed in the BM-MSCs (Figure 2.5). The addition of mechanical stimulation with the low pH culture condition further altered cytokine production. AD-MSCs cultured in low pH conditions with mechanical stimulation produced more TIMP-1, TIMP-2, GRO a/b/c, BDNF,

Angiogenin, and IGFBP-2 compared to low pH conditions (Figure 2.5). AM-MSCs cultured in low pH conditions with mechanical stimulation produced more MCP-1, TIMP-1, OPG, TGF-B3, IFN-γ, and IL-6 compared to low pH conditions (Figure 2.5). BM-MSCs cultured in low pH conditions with mechanical stimulation produced more MCP1, IFN-γ, IL-2, and IL-16 while producing less IL-8, TIMP-1, TIMP-2, angiogenin, VEGF-A, and LIGHT compared to low pH conditions (Figure 2.5).

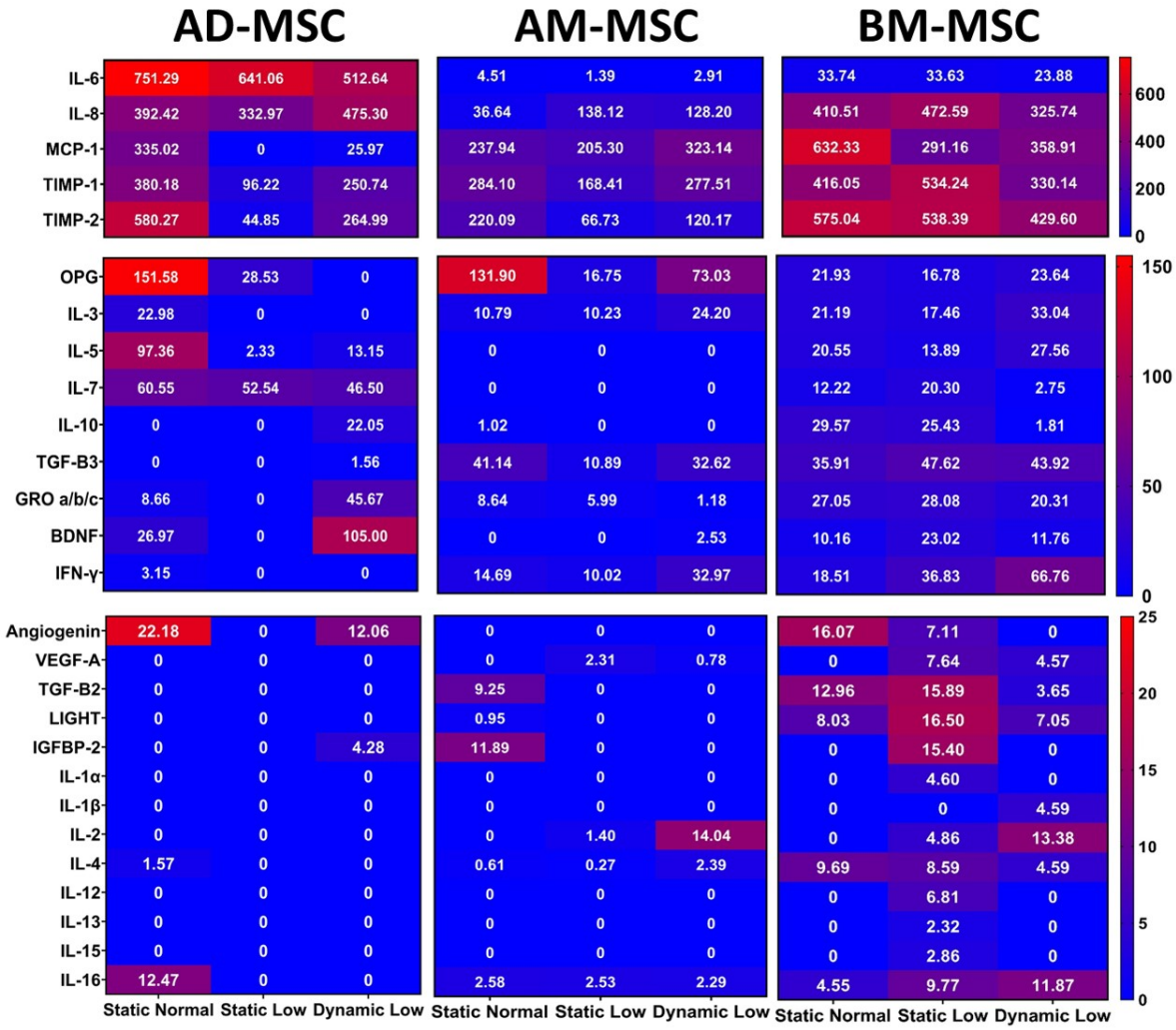

Figure 2.5. Cytokine profiles for each MSC type. Heatmaps depicting chemiluminescent intensity measurements of cytokines present in day 4 cell supernatant. Measurements are clustered by MSC type and arranged to highlight differences between normal static, low static, and low dynamic culture conditions. Each measurement is expressed as a percent change relative to the chemiluminescent intensity measurements of the cell culture media alone to filter out signals not directly produced by MSCs. n = 5 pooled samples per condition.

The change in cytokine concentrations between the low-static and low-dynamic experimental conditions was calculated to highlight the effect that mechanical stimulation had on each MSC type in conjunction with low pH (Figure 2.6). Furthermore, the cytokines that were examined were segregated into three different groupings (anti-inflammatory, pro-inflammatory, and tissue remodeling) based on their reported roles in IVD biology to provide further insights into the effects of tensile strain on MSC biological response in low pH conditions. When tensile strain was applied to AD-MSCs in the presence of low pH a general increase in pro-inflammatory (IGFBP-2, IL-10), anti-inflammatory (GRO a/b/c, IL-5, IL-8, MCP-1), and tissue remodeling cytokines (Angiogenin, BDNF, TGF-b3, TIMP-1, TIMP-2) was observed. Notably, IL-6 was decreased in the dynamic low condition compared to the static low condition. When AM-MSCs were subjected to tensile strain in low pH, there was a slight increase in the concentration of the anti-inflammatory cytokine, IL-4. Like AD-MSCs, AM-MSCs also experienced increases in several other pro-inflammatory (IFN-gamma, IL-2, IL-3, IL-6, MCP-1) and tissue remodeling cytokines (BDNF, TGF-b3, TIMP-1, TIMP-2, OPG); however, the cytokines that changed varied between to two groups. Finally, BM-MSCs in dynamic low pH conditions experienced decreases in anti-inflammatory cytokine concentration levels (IGFBP-2, IL-4, IL-10, and IL-13). For pro-inflammatory cytokines, BM-MSCS saw a mix of cytokines with increased concentrations (IFN-gamma, IL-1beta, IL-2, IL-3, IL-5, and IL-16) and decreased concentrations (GRO a/b/c, IL-1alpha, IL-6, IL-7, IL-8, IL-12, IL-15, and LIGHT). Interestingly, in BM-MSCs, nearly all of the concentrations of cytokines related to tissue remodeling were decreased when subjected to tensile strain in the presence of low pH (Angiogenin, BDNF, TGF-b2, TGF-b3, TIMP-1, TIMP-2, VEGF-A).

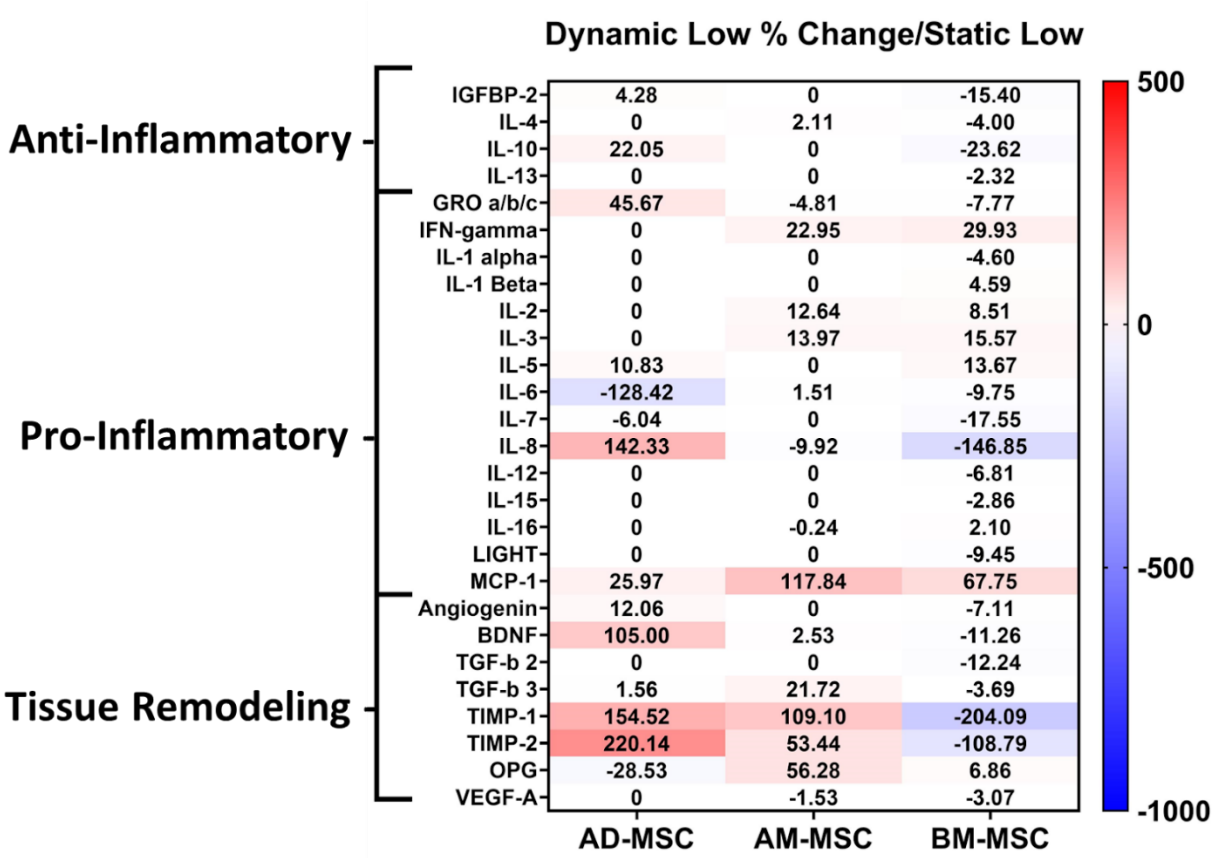

Figure 2.6. Change in the cytokine concentration measurements of the dynamic low condition relative to the static low condition. Dynamic low condition cytokine intensity readings normalized to static low condition cytokine intensity readings to highlight the impact of mechanical stimulation on MSC cytokine expression profiles. Results are grouped by cell type to highlight differences in MSC sources. Cytokines are grouped by known functional role within the IVD to determine the biological response of each MSC type to mechanical (tensile) loading in low pH conditions. Results are reported as percent change compared to static low condition intensity readings. n = 5 pooled samples per condition.

2.4 Discussion

The aim of this study was to evaluate the impact that simplified tensile strain has on the biological behavior of different MSC phenotypes in the context of a low pH microenvironment similar to that of the degenerate IVD and AF repair. Here, it was found that uniaxial tensile strain partially reversed some of the negative impacts that low pH had on cell metabolic activity

and survival in AD-MSCs and AM-MSCs. Interestingly, it was also found that the same level of tensile strain appeared to have a time-dependent effect on BM-MSCs, with strain having a negative impact on metabolic activity, viability, and cell number on Day 2 and positive impact on Day 4. Furthermore, the results presented here suggest that low pH may have a more prominent influence on MSC metabolic activity while tensile strain may stimulate cell proliferation. Overall, this research adds to the understanding of how degenerate IVD-like pH and tensile strain influence MSC biological behavior, and this knowledge can help provide insight into novel strategies or techniques that could be employed to optimize the therapeutic potential of MSC-based therapies targeting the repair and regeneration of the herniated AF.

The motivation for this study results from the current lack of suitable long-term treatments for repairing damaged AF tissue following IVD herniation and subsequent discectomy surgery. Briefly, IVD herniation occurs when degeneration of the AF tissue combined with abnormal or excessive loading of the IVD causes it to prolapse. The current standard of care aims to alleviate patient pain through surgical intervention via discectomy. While studies have shown that these interventions can alleviate short-term discomfort, they can fail to address the long-term complications related to IVD herniation which can include further degeneration of the IVD [31]. One potential solution to this complication is to restore the structural integrity of the AF through the application of an MSC-based therapy in combination with a biomaterial strategy aimed at restoring competency of the damaged outer AF. MSCs are a promising option due to their capacity to proliferate, to differentiate into other cell types, and to modulate the behavior of other cells in the surrounding microenvironment [32]. Several studies have reported that MSC based cell therapies targeting the AF could promote its regeneration [16,33,34]. Although promising, more research needs to be completed to better understand how

MSCs will respond to the harsh microenvironment of the degenerate IVD when used to repair the AF. Furthermore, research on MSCs has shown that the biological behavior of the MSCs can depend on the MSC tissue source [35]. Lastly, prior studies focused on cellular mechanobiology have demonstrated that mechanical stimulation can alter biological behavior of MSCs, suggesting that the application of mechanical forces could potentially improve their function within the harsh microenvironment of the degenerate IVD [36,37]. Thus, this study aimed to evaluate the biological responses of several MSC types from different tissue sources in the presence of both mechanical and biochemical (pH) stimulation representing the harsh chemical microenvironment of the AF.

This study demonstrated that low pH significantly decreased the metabolic activity of AD-, AM-, and BM-MSCs. Low pH also reduced the viability of BM-MSCs. Additionally, in static low pH conditions, AD-MSC cell numbers on both day 2 and day 4 were increased compared to normal pH controls. Furthermore, AM-MSC cell number in the static low pH condition was decreased on day 2 while increased on day 4 when compared to normal pH controls. Finally, BM-MSC cell number in the static low pH condition decreased on both day 2 and day 4 when compared to the normal pH controls. These findings are consistent with the results found in previous studies. Research conducted by Liang et al., found that human AD-MSC viability and metabolic activity were significantly decreased when cultured in low pH (6.5) conditions [22]. This paper provided further evidence that MSC response is dependent on tissue source.

The first major finding of this study is that mechanical stimulation alters MSC metabolic activity, viability, and cell number in low pH conditions. More specifically, the addition of cyclic tensile strain was found to improve AD-MSC and AM-MSC metabolic activity, partially

offsetting the reduction caused by low pH. Cyclic tensile strain also increased AM-MSC cell number. Conversely, cyclic tensile strain further decreased metabolic activity in BM-MSCs. Interestingly, cyclic tensile strain decreased BM-MSC cell number on day 2 compared to the low pH static condition, but then increased cell count on day 4. These findings are similar to those found by Chen et al., where the application of mechanical stimulation via rotational displacement to BM-MSCs on silk fibronectin scaffolds inhibited BM-MSC metabolic activity and proliferation in the short term (1-3 days), while longer time points (9-15 days) demonstrated that this same stimulus increased metabolic activity, proliferation, and collagen production [38]. These findings suggest that mechanical stimulation influences MSC function in a low pH microenvironment. Although increased in low pH when tensile strain was applied, metabolic activity of all MSCs was significantly decreased when compared to cell metabolic activity when cultured in neutral pH (7.4), suggesting that pH is still the dominant characteristic of this microenvironment, which results in an overall negative response [11]. However, this data suggests that mechanical stimulation of MSCs in environments like that in the AF of a degenerated / herniated IVD could result in increased survivability, or that preconditioning in this type of environment prior to implant could potentially improve MSC efficacy. This finding is supported by a study which found that cyclic tensile strain increased expression of genes related to wound healing in MSCs that were implanted in rat AF fissures [33]. The present study and the aforementioned study suggest that mechanical stimulation of MSCs could increase their regenerative capacity within the AF.

The next major finding was that the MSC biological response to tensile strain in the context of low pH is dependent on the tissue source of MSCs. The results of the MTT assay demonstrated that mechanically stimulating AD-MSCs or AM-MSCs in low pH conditions

increased metabolic activity. This suggests that tensile strain could help mitigate the negative effects of low pH on these specific MSCs and could be important to consider when developing an MSC-based repair strategy for the IVD, particularly in the context of disc herniation where the pH of AF tissue could be lowered due to direct contact with the lactic acid from the NP. This finding could also indicate that mechanically conditioning MSCs on a biological scaffold or relevant biomaterial by applying tensile strain prior to application to the AF could potentially improve viability and overall efficacy of the treatment. However, this finding was not universal as tensile strain had a negative effect on BM-MSC metabolic activity. Similarly, ADMSC and AMMSC viability was not altered by application of tensile strain, but BMMSC viability was negatively affected. Thus, BM-MSCs had a different overall response to mechanical loading compared to AD-MSCs and AM-MSCs. Of note, there were no distinct relationships between mechanical stimulation and cell number across MSC types. Previous studies on the effect of cyclic tensile strain on MSC survival and metabolic activity have reported varying results [39,40]. In another study, ADMSC were cultured in monolayer and stretched at 10% cyclic tensile strain at 0.5 Hz for 24 to 72 hours. Their results indicated that ADMSC metabolic activity was enhanced by mechanical stimulation, similar to the results presented herein, however, it was found that ADMSC proliferation was also increased by tensile strain which contradicts the findings in the present study [39]. It is important to note, their study did not incorporate low pH culture conditions, which could explain the differences in our findings. Furthermore, Kang et al, subjected umbilical cord-derived mesenchymal stem cells (UC-MSCs) to 0%, 5%, or 10% strain for 5 seconds followed by 15 seconds of relaxation for 10 days and found that UC-MSC metabolic activity was significantly decreased by cyclic tensile strain. However, ECM (sulfated GAG and elastin) production was significantly increased by the same stimulant [40]. Taken together, it is plausible

that mechanical stimulation of MSCs may increase their therapeutic potential. However, MSC source and the magnitude and intensity of mechanical stimulation appears to dictate the biological response of MSCs, and thus more research must be conducted to further investigate the relationship between mechanical stimulation, MSC source, and biological response.

The third major finding is that mechanical stimulation alters the cytokine release profile of each MSC source. These cytokine release profiles were also MSC source dependent. Overall, it was found that mechanical stimulation up regulated anti-inflammatory cytokine production in AD-MSCs, had little effect on AM-MSC anti-inflammatory cytokine production, and decreased anti-inflammatory cytokine production in BM-MSCs. Furthermore, it altered pro-inflammatory cytokine production across all MSC types. AD-MSCs demonstrated a significant decrease in the concentration of IL-6 while simultaneously expressing a significant increase in IL-8. AM-MSCs experienced mostly increases in pro-inflammatory cytokines, while BM-MSCS mostly experienced a mix of increased and decreased pro-inflammatory cytokine concentrations. In both AD-MSCs and AM-MSCs, mechanical stimulation promoted the production of cytokines related to tissue remodeling (TIMP-1, TIMP-2, OPG, TGF-b2, TGF-b3), blood vessel formation (Angiogenin), and innervation (BDNF). In the context of the IVD, pro-inflammatory cytokines have been linked to IVDD and its progression [41], as well as pain relating to post-discectomy and IVDD [42]. Angiogenic and neurotrophic factors are also critical in the development of IVDD as complications that arise as a part of IVDD, are in part due to nerve and blood vessel ingrowth [43]. This suggests that implanted cells which express these cytokines could be a suboptimal source for use in a cell-based therapy aimed at repairing and regenerating the IVD. In the context of the herniated IVD, blood vessels allow for immune cell migration into the IVD, which further promotes inflammation in and degeneration of the IVD (blood vessels allow for immune cell

migration into the IVD)[44]. Furthermore, studies have shown that BMMSC cultured in low pH increases senescence which has been linked to a proinflammatory phenotype in MSCs [45,46]. Taken together, this data suggests that the low pH conditions of the degenerate IVD could limit the therapeutic potential of certain MSC types in AF repair and regeneration. Additionally, if the implanted MSCs promoted inflammation, angiogenesis, or innervation, this could negatively impact the healing of the IVD in the context of IVDD, herniation repair, and AF regeneration [41,43,46].

Several key observations were also made by examining each cytokine individually. For example, in the context of IVD regeneration and disease, IL-8 has been theorized to be a key pro-inflammatory cytokine in the onset and progression of IVDD related lower back pain (LBP), and a possible therapeutic target for chronic LBP [47]. It is worth noting that the results herein indicated that IL-8 was upregulated by mechanical stimulation in AD-MSCs and downregulated in AM-MSCs and BM-MSCs. This suggests that AM-MSCs and BM-MSCs could potentially induce less inflammation if utilized for an IVD-specific cellular therapy.

Two other key cytokines that we examined in each MSC type were angiogenin and BDNP. Angiogenin is a potent angiogenic factor that promotes blood vessel invasion and development while BDNP is a neurotrophic factor that modulates inflammatory pain hypersensitivity and promotes the formation of vascular structures [48]. Within the IVD, angiogenesis is theorized to be a large contributor to lower back pain and the progression of IVDD. Results suggested that BM-MSCs down-regulated angiogenin and BDNP when subjected to mechanical stimulation, while AD-MSCs up-regulated angiogenin and BDNP. Angiogenin and BDNP production by AM-MSCs was largely unaffected by mechanical stimulation. These results could suggest that the application of mechanical stimulation to MSCs through

interventions such physical therapy or exercise could be helpful in reducing the degree of pain and angiogenesis involved in therapies that utilize MSCs, particularly those therapies that involve AD-MSCs.

Finally, studies have investigated how the MMP/TIMP ratio affects matrix degradation. Prior studies found that as this ratio decreases, matrix degradation is inhibited within the IVD. Results herein indicate that mechanical stimulation up regulated AD-MSC and AM-MSC TIMP production, and down regulated BM-MSC TIMP production. It is important to note that the concentrations of TIMP for BM-MSC were higher than that of AD-MSC and AM-MSCs regardless of the experimental condition examined. The high concentrations of TIMP-1 and TIMP-2 suggest that MSCs may work to reduce matrix degradation and slow degeneration by further inhibiting MMPs [49,50].

Taken together, the cytokine analysis shows that MSCs may work to quell inflammation, suppress angiogenesis, and reduce matrix degradation in the context of AF regeneration. However, the source of the MSC must also be considered, as these cells exhibit remarkably different release profiles. These findings are generally in agreement with other related research which found that MSC cytokine production varies by tissue source [51,52]. Therefore, we can conclude that it is important to consider the differences in MSC types when developing a MSC based therapy for AF repair and regeneration.

A key limitation of this study was the substrate stiffness of the plates utilized for culturing the cells. Previous work by others demonstrated that low (1 - 30 kpa) substrate stiffness significantly decreased BMMSC adhesion and proliferation. The plates that were used in the current study had a low surface stiffness (~30kPa), which may have significantly contributed to

the viability, cell count, and metabolic activity of the BMMSCs that was noted throughout the study [53]. Another limitation of the study was that the effect of low pH and mechanical stimulation on the production of active MMPs was not directly investigated. To address these limitations, further studies should be completed.

2.5 Conclusion

In conclusion, this study demonstrated that MSC survival and response is altered by the combination of low-pH and tensile strain, and that the tissue source of the MSC plays a significant role in the survival and response that is realized by this combined microenvironment. The understanding of how both the chemical and mechanical microenvironment affects MSC behavior in the context of herniation repair and IVDD will allow further research to increase the therapeutic potential of MSC based cellular therapy in the hopes of repairing and regenerating the damaged AF.

2.6 References

1. Lyu F, Cui H, Pan H, et al. Painful intervertebral disc degeneration and inflammation: From laboratory evidence to clinical interventions. *Bone Research*. 2021;9(1):7.

2. Molinos M, Almeida CR, Caldeira J, Cunha C, Gonçalves RM, Barbosa MA. Inflammation in intervertebral disc degeneration and regeneration. *Journal of the Royal Society Interface*. 2015;12(104):20141191.

3. Long DM, BenDebba M, Torgerson WS, et al. Persistent back pain and sciatica in the united states: Patient characteristics. *Clinical Spine Surgery*. 1996;9(1):40-58.

4. Vialle LR, Vialle EN, Henao JES, Giraldo G. Lumbar disc herniation. *Revista Brasileira de Ortopedia (English Edition)*. 2010;45(1):17-22.

5. Strenge KB, DiPaola CP, Miller LE, Hill CP, Whitmore RG. Multicenter study of lumbar discectomy with barricaid annular closure device for prevention of lumbar disc reherniation in US patients: A historically controlled post-market study protocol. *Medicine*. 2019;98(35).

6. Miller LE, McGirt MJ, Garfin SR, Bono CM. Association of annular defect width after lumbar discectomy with risk of symptom recurrence and reoperation: Systematic review and meta-analysis of comparative studies. *Spine*. 2018;43(5):E308.

7. Martens F, Vajkoczy P, Jadik S, Hegewald A, Stieber J, Hes R. Patients at the highest risk for reherniation following lumbar discectomy in a multicenter randomized controlled trial. *JBJS Open Access*. 2018;3(2).

8. Choy WJ, Phan K, Diwan AD, Ong CS, Mobbs RJ. Annular closure device for disc herniation: Meta-analysis of clinical outcome and complications. *BMC musculoskeletal disorders*. 2018;19:1-9.

9. Chu G, Zhang W, Han F, et al. The role of microenvironment in stem cell-based regeneration of intervertebral disc. *Frontiers in Bioengineering and Biotechnology*. 2022;10:968862.

10. Lyu F. Impact of microenvironmental changes during degeneration on intervertebral disc progenitor cells: A comparison with mesenchymal stem cells. *Bioengineering*. 2022;9(4):148.

11. Liang C, Li H, Tao Y, et al. Responses of human adipose-derived mesenchymal stem cells to chemical microenvironment of the intervertebral disc. *Journal of Translational Medicine*. 2012;10:1-10.

12. Wuertz K, Godburn K, Neidlinger-Wilke C, Urban J, Iatridis JC. Behavior of mesenchymal stem cells in the chemical microenvironment of the intervertebral disc. *Spine*. 2008;33(17):1843.

13. Li H, Tao Y, Liang C, et al. Influence of hypoxia in the intervertebral disc on the biological behaviors of rat adipose-and nucleus pulposus-derived mesenchymal stem cells. *Cells Tissues Organs (Print)*. 2014;198(4):266-277.

14. Wuertz K, Urban J, Klasen J, et al. Influence of extracellular osmolarity and mechanical stimulation on gene expression of intervertebral disc cells. *Journal of Orthopaedic Research*. 2007;25(11):1513-1522.

15. Borem R, Madeline A, Bowman M, Gill S, Tokish J, Mercuri J. Differential effector response of amnion-and adipose-derived mesenchymal stem cells to inflammation; implications for intradiscal therapy. *Journal of Orthopaedic Research®*. 2019;37(11):2445-2456.

16. Pirvu T, Blanquer SB, Benneker LM, et al. A combined biomaterial and cellular approach for annulus fibrosus rupture repair. *Biomaterials*. 2015;42:11-19.

17. Liu C, Choi H, Johnson ZI, Tian J, Shapiro IM, Risbud MV. Lack of evidence for involvement of TonEBP and hyperosmotic stimulus in induction of autophagy in the nucleus pulposus. *Scientific reports*. 2017;7(1):4543.

18. Diamant B, Karlsson J, Nachemson A. Correlation between lactate levels and pH in discs of patients with lumbar rhizopathies. *Experientia*. 1968;24(12):1195-1196.

19. Tetlow LC, Adlam DJ, Woolley DE. Matrix metalloproteinase and proinflammatory cytokine production by chondrocytes of human osteoarthritic cartilage: Associations with degenerative changes. *Arthritis & Rheumatism*. 2001;44(3):585-594.

20. Tavana S, Masouros SD, Baxan N, Freedman BA, Hansen UN, Newell N. The effect of degeneration on internal strains and the mechanism of failure in human intervertebral discs analyzed using digital volume correlation (DVC) and ultra-high field MRI. *Frontiers in Bioengineering and Biotechnology*. 2021;8:610907.

21. Wuertz K, Godburn K, Iatridis JC. MSC response to pH levels found in degenerating intervertebral discs. *Biochem Biophys Res Commun*. 2009;379(4):824-829.

22. Li H, Liang C, Tao Y, et al. Acidic pH conditions mimicking degenerative intervertebral discs impair the survival and biological behavior of human adipose-derived mesenchymal stem cells. *Exp Biol Med*. 2012;237(7):845-852.

23. Vadalà G, Ambrosio L, Russo F, Papalia R, Denaro V. Interaction between mesenchymal stem cells and intervertebral disc microenvironment: From cell therapy to tissue engineering. *Stem Cells International*. 2019;2019.

24. Pratsinis H, Papadopoulou A, Neidlinger-Wilke C, Brayda-Bruno M, Wilke H, Kletsas D. Cyclic tensile stress of human annulus fibrosus cells induces MAPK activation: Involvement in proinflammatory gene expression. *Osteoarthritis and cartilage*. 2016;24(4):679-687.

25. Kozlowska U, Krawczenko A, Futoma K, et al. Similarities and differences between mesenchymal stem/progenitor cells derived from various human tissues. *World journal of stem cells*. 2019;11(6):347.

26. Topoluk N, Hawkins R, Tokish J, Mercuri J. Amniotic mesenchymal stromal cells exhibit preferential osteogenic and chondrogenic differentiation and enhanced matrix production compared with adipose mesenchymal stromal cells. *Am J Sports Med*. 2017;45(11):2637-2646.

27. O'Connell GD, Johannessen W, Vresilovic EJ, Elliott DM. Human internal disc strains in axial compression measured noninvasively using magnetic resonance imaging. *Spine*. 2007;32(25):2860-2868.

28. Mosmann T. Rapid colorimetric assay for cellular growth and survival: Application to proliferation and cytotoxicity assays. *J Immunol Methods*. 1983;65(1-2):55-63.

29. Carpenter AE, Jones TR, Lamprecht MR, et al. CellProfiler: Image analysis software for identifying and quantifying cell phenotypes. *Genome Biol*. 2006;7:1-11.

30. Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: A generalist algorithm for cellular segmentation. *Nature methods*. 2021;18(1):100-106.

31. Illien-Jünger S, Lu Y, Purmessur D, et al. Detrimental effects of discectomy on intervertebral disc biology can be decelerated by growth factor treatment during surgery: A large animal organ culture model. *The Spine Journal*. 2014;14(11):2724-2732.

32. Leung VY, Chan D, Cheung KM. Regeneration of intervertebral disc by mesenchymal stem cells: Potentials, limitations, and future direction. *European Spine Journal*. 2006;15:406-413.

33. Deng R, Kang R, Jin X, et al. Mechanical stimulation promotes MSCs healing the lesion of intervertebral disc annulus fibrosus. *Frontiers in Bioengineering and Biotechnology*. 2023;11:1137199.

34. Wei A, Shen B, Williams L, Diwan A. Mesenchymal stem cells: Potential application in intervertebral disc regeneration. *Translational pediatrics*. 2014;3(2):71.

35. Berebichez-Fridman R, Montero-Olvera PR. Sources and clinical applications of mesenchymal stem cells: State-of-the-art review. *Sultan Qaboos University Medical Journal*. 2018;18(3):e264.

36. Sun Y, Wan B, Wang R, et al. Mechanical stimulation on mesenchymal stem cells and surrounding microenvironments in bone regeneration: Regulations and applications. *Frontiers in cell and developmental biology*. 2022;10:808303.

37. Maul TM, Chew DW, Nieponice A, Vorp DA. Mechanical stimuli differentially control stem cell behavior: Morphology, proliferation, and differentiation. *Biomechanics and modeling in mechanobiology*. 2011;10:939-953.

38. Chen J, Horan RL, Bramono D, et al. Monitoring mesenchymal stromal cell developmental stage to apply on-time mechanical stimulation for ligament tissue engineering. *Tissue Eng*. 2006;12(11):3085-3095.

39. Fang B, Liu Y, Zheng D, et al. The effects of mechanical stretch on the biological characteristics of human adipose-derived stem cells. *J Cell Mol Med*. 2019;23(6):4244-4255.

40. Kang M, Yoon H, Seo Y, Park J. Effect of mechanical stimulation on the differentiation of cord stem cells. *Connect Tissue Res*. 2012;53(2):149-159.

41. Li S, Huang C, Xiao J, et al. The potential role of cytokines in diabetic intervertebral disc degeneration. *Aging and Disease*. 2022;13(5):1323.

42. Zu B, Pan H, Zhang X, Yin Z. Serum levels of the inflammatory cytokines in patients with lumbar radicular pain due to disc herniation. *Asian Spine Journal*. 2016;10(5):843.

43. Wiet MG, Piscioneri A, Khan SN, Ballinger MN, Hoyland JA, Purmessur D. Mast cell-intervertebral disc cell interactions regulate inflammation, catabolism and angiogenesis in discogenic back pain. *Scientific reports*. 2017;7(1):12492.

44. Lee S, Millecamps M, Foster DZ, Stone LS. Long-term histological analysis of innervation and macrophage infiltration in a mouse model of intervertebral disc injury–induced low back pain. *Journal of Orthopaedic Research®*. 2020;38(6):1238-1247.

45. Fliefel R, Popov C, Tröltzsch M, Kühnisch J, Ehrenfeld M, Otto S. Mesenchymal stem cell proliferation and mineralization but not osteogenic differentiation are strongly affected by extracellular pH. *Journal of Cranio-Maxillofacial Surgery*. 2016;44(6):715-724.

46. Liu Y, Chen Q. Senescent mesenchymal stem cells: Disease mechanism and treatment strategy. *Current molecular biology reports*. 2020;6:173-182.

47. Guo Y, Li C, Shen B, et al. Is there any relationship between plasma IL-6 and TNF-α levels and lumbar disc degeneration? A retrospective single-center study. *Dis Markers*. 2022;2022.

48. Gruber HE, Ingram JA, Hoelscher G, Zinchenko N, Norton HJ, Hanley EN. Brain-derived neurotrophic factor and its receptor in the human and the sand rat intervertebral disc. *Arthritis research & therapy*. 2008;10(4):1-8.

49. Kozacı LD, Guner A, Oktay G, Guner G. Alterations in biochemical components of extracellular matrix in intervertebral disc herniation: Role of MMP-2 and TIMP-2 in type II collagen loss. *Cell Biochemistry and Function: Cellular biochemistry and its modulation by active agents or disease*. 2006;24(5):431-436.

50. Deng B, Ren JZ, Meng XQ, et al. Expression profiles of MMP-1 and TIMP-1 in lumbar intervertebral disc degeneration. *Genet Mol Res*. 2015;14(4):19080-19086.

51. Ivanova-Todorova E, Bochev I, Mourdjeva M, et al. Adipose tissue-derived mesenchymal stem cells are more potent suppressors of dendritic cells differentiation compared to bone marrow-derived mesenchymal stem cells. *Immunol Lett*. 2009;126(1-2):37-42.

52. Chan C, Wu K, Lee Y, et al. The comparison of interleukin 6–associated immunosuppressive effects of human ESCs, fetal-type MSCs, and adult-type MSCs. *Transplantation*. 2012;94(2):132-138.

53. Sun M, Chi G, Li P, et al. Effects of matrix stiffness on the morphology, adhesion, proliferation and osteogenic differentiation of mesenchymal stem cells. *International journal of medical sciences*. 2018;15(3):257.

CHAPTER 3

Design and Testing of a Multi-Well Tissue Stretching Bioreactor

3.1 Introduction

Mechanotransduction is the process by which mechanical stimuli are converted into biochemical signals that direct cellular processes. Tensile, compressive, or shear stresses cause conformational changes in the cell at the molecular level and activate different signaling pathways that alter gene transcription [1]. This in turn influences many cellular functions such as proliferation, differentiation, protein synthesis, migration, and apoptosis. As a result, mechanotransduction plays a crucial role in organ development and tissue homeostasis [2] and deviation from the physiologic range of mechanical stimulation can lead to dysfunction and disease. For example, several studies have related diseases such as arteriosclerosis [3], fibrosis [4], muscular dystrophy [5], osteoporosis [6], and certain types of cancers [7] to abnormal mechanical stimulation. Understanding how physical forces impact cellular function is therefore critical to preventing, arresting, or reversing many of these conditions.

While studying the influence of mechanical stimulation on cellular function *in vivo* would be the most direct way to examine the role of mechanotransduction in cell biology, it is currently hindered by the technical limits to monitor cell activity and the inherent difficulty of parsing out effects due solely to mechanical stimulation within the complex network of biological signaling that occurs within the body. Therefore, it is helpful to employ *in vitro* methods to examine the effects of mechanotransduction on cell biology. Bioreactors can serve as effective tools for studying the effects of mechanotransduction *in vitro,* as they can be designed with the capacity to

apply physiologically relevant mechanical stimulation to cells in a controlled, sterile cell culture environment [8].

To replicate the *in vivo* effects of mechanical stimulation on cell biology, bioreactors must be built specifically to apply precise user-defined physical stimuli. 3-D printing has become an increasingly attractive option for manufacturing custom bioreactors for studying mechanobiology due to the low cost of production and the design flexibility it offers to researchers. Several studies have recently been published highlighting the efficacy of 3-D printed bioreactors, each with their own specific use case [9-11]. Cook et al. designed a novel bioreactor to stretch up to four electrospun biological scaffolds concurrently using a system that incorporated load cells to monitor tissue stress during loading. It also had a specially designed pre-tensioning mechanism to ensure that the amount of pre-strain in each sample was consistent prior to loading [9]. Janvier et al. developed a custom 3-D printed bioreactor chamber with the goal of developing a universal modular platform capable of interfacing with a variety of different actuator systems. The purpose of this research was to create a low-cost system for early career and interdisciplinary researchers to use. This work established a robust platform for easy comparison of results across experiments as well as enabling reproducibility between experiments [10]. Putame et al. utilized fused deposition modeling (FDM) 3-D printing to create a single chamber tensile bioreactor that used a custom-designed clamping system to stretch multiple types of samples, including hydrogels and electro spun biological samples [11].

While each of these systems were able to test multiple samples independently, monitor local strains, acquire real-time fluorescent images, and clamp various tissue samples; no single system incorporated all of these design criteria into the same platform. In this study, our primary goal was to build upon these existing bioreactor system designs and develop a low-cost, 3-D

printed bioreactor that could combine all of these features into a single system. More specifically, this involves the capability to stretch multiple independent biological samples simultaneously, allow for real-time fluorescent imaging to assess cell viability and morphology, and enable evaluation of local tissue strains at different time points during long-term experiments. We hypothesized that a custom bioreactor chamber can be manufactured using 3-D printing to meet the design criteria outlined above. Biocompatibility tests, fluorescent imaging, protein assays, and strain validation analyses were utilized to verify that each of these conditions are met and validate the biological effects of the bioreactor on cell-seeded tissue samples.

3.2 Materials and Methods

3.2.1 Bioreactor Chamber Design Requirements and Fabrication

The overall goal of this 3-D printed bioreactor was to apply cyclic tensile strain to multiple biological scaffolds seeded with cells for an extended tissue culture period while simultaneously allowing for real-time imaging to monitor cell and tissue behavior over time. To achieve this goal, 8 distinct design criteria were defined to set the scope of the project (Table 3.1). More specifically, the first design criteria for this bioreactor was to enable up to 6 biological scaffolds to be tested simultaneously and independently of each other to allow for an effective number of experimental repeats per experimental run. Secondly, the bioreactor needed to apply a range of tensile strains on samples (1 – 10% tensile strain) to mimic the physiologically relevant stains that have been observed in a variety of different soft collagenous tissues within the body [18,19]. Because certain biological scaffolds are not necessarily homogeneous in nature [12,13], the bioreactor must be compatible with real-time imaging platforms allowing for fluorescent and brightfield images to assess both cell and tissue behavior.

Fluorescent imaging could be combined with live cell stains such as NucBlue to track cell count and orientation over time while digital image correlation (DIC) software could be utilized in conjunction with brightfield images of the biological samples to calculate the strain field distribution across each sample as it was stretched. Furthermore, to be able to understand how mechanical stretch affects cellular functioning and potential tissue/scaffold remodeling at different time points, it was decided that the bioreactor should be able to maintain cells in sterile tissue culture for extended periods of time without impacting cell viability. Thus, the bioreactor needed to be both biocompatible and autoclavable to ensure sterility and maintain the cells in a healthy culture environment. It was determined that each sample should be able to be submerged in 3-5 mL of cell culture media to match media volumes typically used in normal 6-well cell culture plates to ensure sufficient nutrient availability and buffering during culture while simultaneously minimizing the amount of media required for an experiment. The final design criteria was that the amount of displacement delivered to each sample via the mechanical actuator needed to be consistent across each sample to stretch each sample as consistently as possible.

Table 3.1: Bioreactor design criteria.

Number of samples	1 - 6
Tensile Strain Range	1 – 15%
Real-Time Imaging Capability	Yes
Long-Term Culture Capacity	> 7days
Biocompatibility	< 5% reduction in cell viability
Autoclavable	Yes
Culture Media Volume per Well	3 – 5 mL
Variability in Displacement	< 0.25 mm

With these criteria in mind, the initial design strategy for the bioreactor was to construct a chassis that could apply tensile stretching to biological scaffolds being cultured in a standard 6-well tissue culture plate (Corning) using a commercially available linear actuator (CellScale). The components needed for this design consisted of: a standard 6-well tissue culture plate, a 6-well insert that fit into each well and acted as an anchor for one end of the biological scaffolds, a housing unit that fit around the 6-well insert and allowed for the linear actuator to connect to the 6-well plate, a linear actuator, and a linear actuator arm that was used to transform the motion of the linear actuator into mechanical tension that stretched the biological scaffolds (Figure 3.1a).

Each of these components were modeled as individual parts using SolidWorks 2021 and compiled into a single assembly to visualize the entire system prior to manufacturing the components. After completing the initial assembly, the bioreactor system was simplified by combining the 6-well insert and the housing unit into a single piece for the final bioreactor design (Figure 3.1b). The individual part files were converted to STL files and uploaded onto a commercially available stereolithography (SLA) 3D printer (Formlabs). The parts were printed using BioMed Amber resin (Formlabs). After printing, each part was washed in isopropanol alcohol (IPA) for 20 minutes, allowed to air dry for 30 minutes, and then placed into an oven at 60 °C overnight to ensure that the part was completely dry. The following day, the print supports were trimmed away, and the parts were placed into a UV curing station (Formlabs) and fully cured for 30 minutes at 70 °C. The parts were then autoclaved for sterilization and the bioreactor was assembled in a cell culture laminar flow hood.

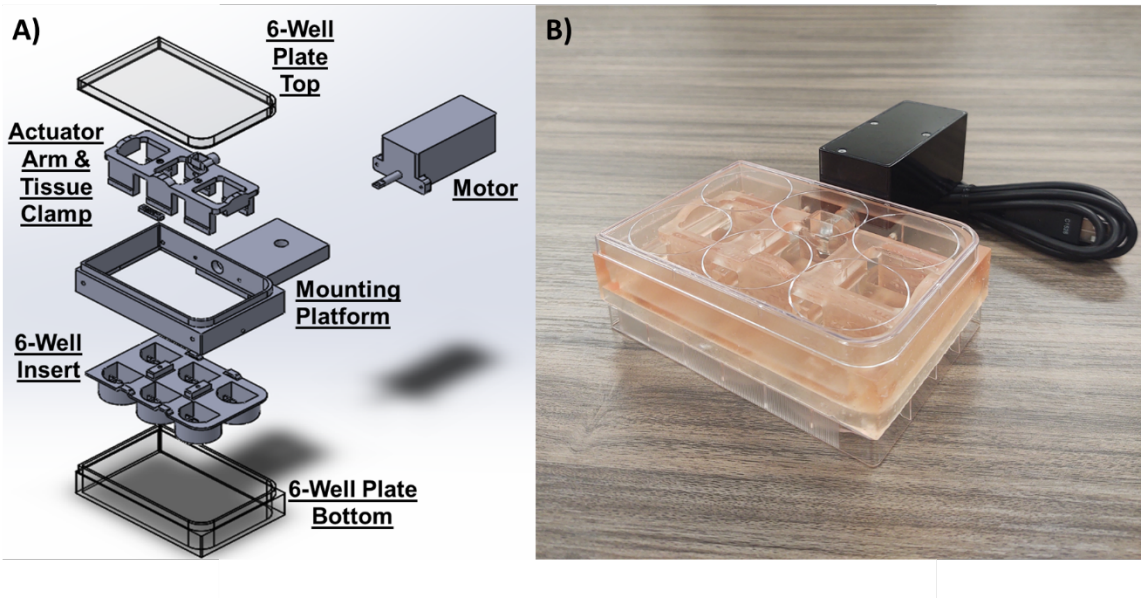

Figure 3.1. Final 3-D printed bioreactor design. (A) Exploded CAD drawing illustrating the different components of the 3D printed bioreactor, including: (i) the tissue culture chamber, (ii) the tensile arm that applies tensile strain to the collagenous tissue, (iii) the MCFX-2 motor unit used to actuate the tensile arm, (iv) the bottom half of a standard 6-well cell culture plate that attaches to the bottom of the tissue culture chamber and houses the collagenous samples

during culture, and (v) the top half of a standard 6-well cell culture plate that attaches to the top of the tissue culture chamber. **(B)** Representative image of the fully constructed bioreactor.

3.2.2 Biological Scaffold Preparation

Tissue samples consisting of porcine pericardium were obtained from a local abattoir, cleaned, and decellularized using a modified process previously described by Borem *et al.* [15]. Briefly, the pericardium was immersed in distilled water at 4 °C for 24 hours to lyse any viable cells in the tissue following dissection. The samples were then cut into pieces (~ 4 cm x 4 cm each) and washed in a decellularization (decell) solution (50 mM Tris, 0.15% v/v Trito X-100, 0.25% w/v deoxycholic acid, 0.1% ethylenediaminetetraacetic acid, and 0.02% w/v sodium azide) for 3 days at room temperature on a orbital shaker (150 rpm, Thomas Scientific) to remove cellular debris from the samples. On day 3, the decell solution was replaced with fresh solution and washed for another 3 days. After a total of 6 days, the samples were washed in 70% ethanol and distilled water (2X each for 10 minutes while agitated at room temperature). Then samples were placed in a DNase/RNase solution (7.5 pH, 720 U/ml each) containing 5 mM magnesium chloride at 37 °C for 24 hours at 150 rpm. Samples were sterilized in a 0.01% peracid acid solution for 2 hours on the orbital shaker at 150 rpm and then washed in sterile PBS 3 times (1 hour each at 37 °C and 150 rpm) before being placed in a neutralization solution (50% dulbecco's modified edge medium (DMEM), 48% fetal bovine serum (FBS), and 2% antibiotic/antimycotic) and stored at 4 ° C until experimental use.

3.2.3 Cytotoxicity Assay

The cytotoxicity of the bioreactor components was assessed using a modified *in vitro* test for cytotoxicity outlined in ISO 10993-5 [16]. For this study, GFP-labeled NIH-3T3 (Cell Biolabs, Inc.) were plated into 75 cm^2 cell culture flasks and cultured for 24 hours at 37 °C. Cells were

maintained in culture until confluent and were passaged at 72 h. Concurrently, test components (3-D printed resin sample, BioMed Amber; 316 Stainless steel screws and nuts; pericardium tissue sample) of the 3-D printed bioreactor were autoclaved, assembled within the confines of a laminar flow cabinet to maintain sterility, and placed within a cell culture dish with an appropriate amount of cell culture media and maintained at 37° C for 72 h to generate extract for the cytotoxicity assay. A total of 3 replicate extract samples were generated following this process. At 72 h, the cells were passaged into 6-well plates. Cells in 3 of the wells were submerged in regular cell culture media to act as experimental controls. Cells in the remaining 3 wells were submerged in extract from the bioreactor components to test for cytotoxicity. Cells in the 6-well plate were maintained in culture for 2 days to allow for any cytotoxic effects to become apparent. Cytotoxicity was assessed using an alamarBlue™ cell viability assay in conjunction with a visual inspection of cell morphology using brightfield imaging.

3.2.4 Sample Loading and Long-Term Viability Analysis

Prior to loading, samples were removed from the neutralization solution, placed into a 60 mm plastic tissue culture dish (Corning), and sectioned into segments approximately 25 mm x 10 mm in length. For each experiment, 6 segmented samples were then placed into 6-well plates with the fibrous side of the pericardium face up. Mesenchymal stromal cells derived from human adipose tissue (AD-MSCs) were seeded onto the samples in a dropwise fashion at approximately 3,500 cells/cm^2 in normal cell culture media (Dulbecco's Modified Eagles Media (DMEM), 10% fetal bovine serum (FBS), and 1% penicillin/streptomyocin) and incubated for 4 hours to allow the cells to adhere to the samples. Each well was then gently filled with 2 mL of normal cell culture media to ensure proper nutrient exchange and then incubated for an additional 20 hours.

Once the cells had sufficient time to adhere to the samples, they were loaded into the 3D printed bioreactor.

To load the samples into the bioreactor, the bioreactor was placed face down on a sterile drape within a cell culture laminar flow hood so that the clamp ends of the actuator arm and the 6-well insert were facing up. Sterile forceps were used to gently drape the cell-seeded samples across both clamp ends. A 3-D printed clamp end was used to secure the end of the sample corresponding to the 6-well insert. Slack was then removed from the sample by gently pulling the sample to remove the slack and then clamping the actuator end of the sample. Samples were loaded into the bioreactor with the fibrous side facing up so that the seeded cells could be easily imaged during tissue culture. Once all 6 samples had been loaded, the bioreactor was flipped right-side-up and attached to the bottom half of a standard 6-well plate so that the tissue samples could be submerged in media. 3 mL of standard cell culture media were then added to each well and the top half of the standard 6-well plate was placed on top of the bioreactor housing unit. An overview of the loading procedure is depicted in Figure 3.2.

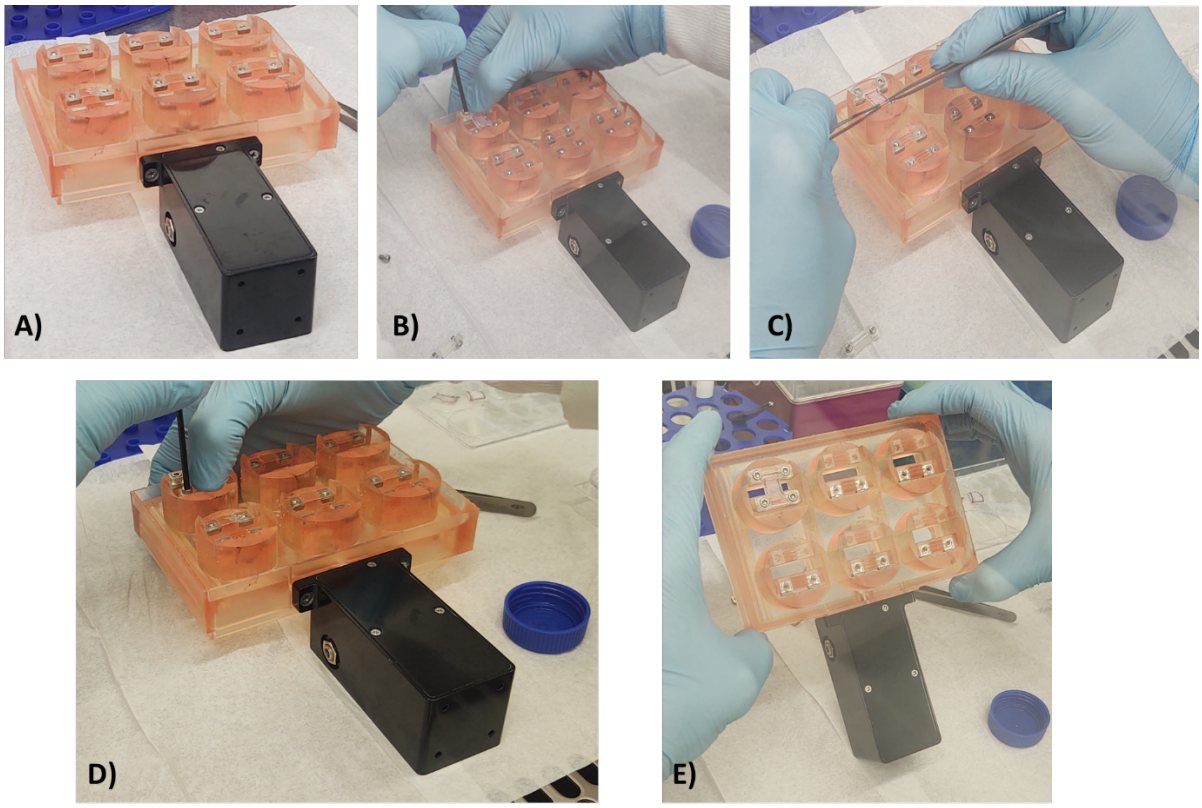

Figure 3.2. Protocol for loading collagenous tissue samples into the 3-D printed bioreactor. (A) The motor, tensile arm, and tissue culture chamber were assembled in a sterile tissue culture hood and flipped upside down so that the clamp ends of the tensile arm and the tissue culture chamber were face-up. **(B)** Pre-seeded tissue samples were carefully extracted from tissue culture plates and lain across the clamp ends of the tensile arm and tissue culture chamber so that the side of the tissue seeded with cells was face-up. The tissue samples were clamped down at the end of the sample that was lying draped across the tissue culture chamber using a sterilized Alan wrench tool. **(C)** Forceps were used to pull the slack out of the tissue sample and keep it held in place across the tensile arm clamp for each well. **(D)** The remaining end of the tissue sample was clamped down to ensure that the collagenous sample was being stretched immediately upon actuation of the tensile arm. **(E)** Representative image of sample fully loaded in the bioreactor.

The bioreactor was then placed in an incubator and connected to an external computer via a wire fed through a port on the back of the incubator. A custom loading protocol written using commercially available software (CellScale) was used to apply ~5% cyclic tensile strain to the samples for 16 hours and then 1% static tensile strain for 8 hours. This loading protocol was repeated each day for 12 days. Media changes were performed every 3 days to maintain proper cell culture nutrient conditions during the experiment.

At day 12, cell morphology, alignment, and viability were assessed using a Live/Dead staining assay. Live cell membranes were stained with calcein AM while dead cell membranes were stained ethidium homodimer-1 (EthD-1) following the manufacturer's instructions: 2 uM of calcein AM and 4 uM of EthD-1, 15 min incubation at 37 °C. 2X Images were obtained using a fluorescent microscope (EVOS Auto FL) and analyzed using an open image analysis software (CellProfiler). Briefly, images were loaded into the software and an automatic segmentation algorithm (Ostu) was used to identify the DAPI images of the cell nuclei. These objects were then used as seeds in a subsequent step to segment the live cell outlines using a watershed propagation algorithm. The number of live cells was determined as the number of outlines identified using this method. The shape of each outline was also used to determine cell morphology and alignment relative to the direction of stretch. The number of dead cells was determined by counting the number of objects obtained by an automatic segmentation algorithm (Manual, threshold = 0.255). The viability was determined as the percentage of live cells to the total number of cells. Cells plated in standard 6-well plates and cultured with standard cell culture media were used as controls.

3.2.5 Tissue Strain Validation and Distribution Analysis

A second experiment was performed to quantify the average magnitude and typical distribution of tissue strain in a pericardium sample while it was being stretched in the bioreactor. For this experiment, the strain distribution in the tissue samples was evaluated using the methodology for evaluating soft tissue strain fields described by Lionelle et al. [17] Briefly, acellular pericardium samples were loaded into the bioreactor as described above in *3.2.4 Sample Loading and Long-Term Viability Analysis* and an airbrush was used to apply a speckled pattern

of black, matte paint to the surface of each sample. The bioreactor was then loaded onto a brightfield microscope for imaging and was connected to an external computer to control the actuator motor. Then a 2X image of the unstretched speckled sample was obtained to serve as a reference image. The computer was then used to actuate the motor and stretch the samples in 0.25 mm increments from the unstretched position to 2 mm in tension. At each 0.25 mm increment, a subsequent brightfield image was taken at 2X to obtain a series of images of the tissue being stretched. The dense speckle pattern tracked across subsequent images to evaluate the strain distribution field using digital image correlation (DIC) techniques while the samples were stretched. This process was repeated for each sample (n = 6). Once a complete stack of images was obtained for each sample, they were loaded into an open source, Matlab based, DIC analysis software (ncorr). Using standard settings for the software, strain distribution maps were computed at each increment of stretch relative to the initial unstretched reference image. These maps were then analyzed to compute the median tensile strain in the direction of stretch for each sample and plotted against the distance the sample was stretched to obtain average tissue strain as a function of applied stretch. The details of the analysis process are depicted in Figure 3.3.

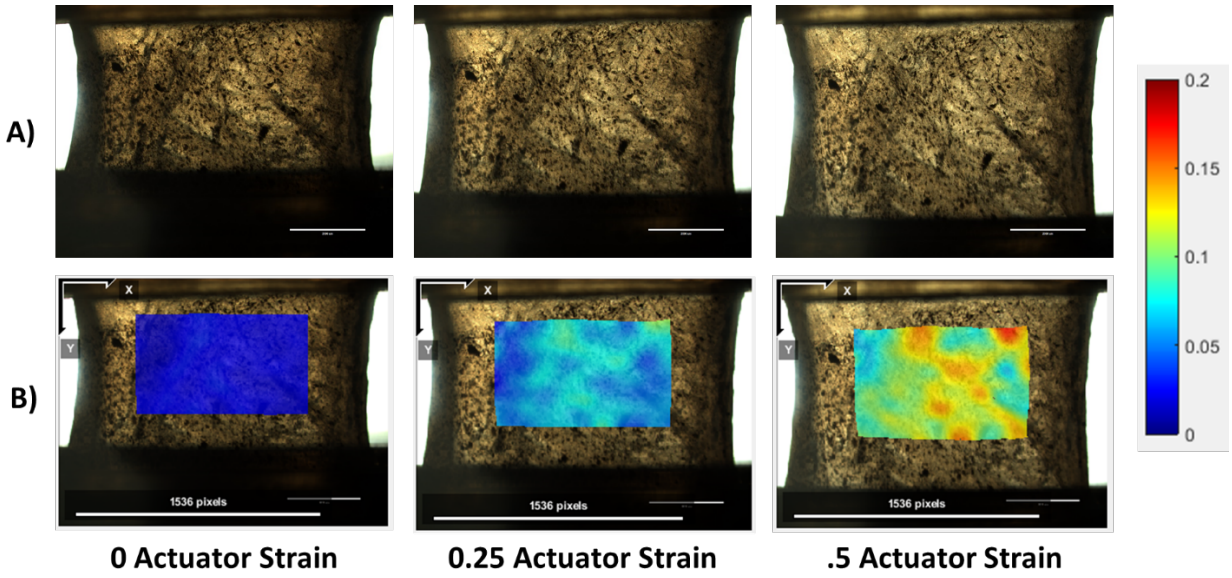

Figure 3.3. Example of total tissue strain distribution analysis procedure. Samples were stretched from 0 actuator strain to 0.5 actuator strain (Change in grip-to-grip distance over initial grip-to-grip distance) in increments of 0.0625 actuator strain. **A)** An image of the sample was taken at each increment of stretch to obtain a stack of images. **B)** The stack of images was loaded into a MATLAB based, open source DIC software (ncorr) and analyzed at each increment to obtain strain distribution maps of the tissue at each increment of strain.

To evaluate how the strain distribution varied in different regions of each sample, the strain distribution maps were segmented into 7 different distinct regions: a 1 pixel x 1 pixel region at the center of the strain field, a 10 pixel x 10 pixel region at the center of the strain field, a 20 pixel x 20 pixel region at the center of the strain field, and four rectangular regions at the top, bottom, left, and right portions of the strain field. The median strain values for each of these regions was calculated and used to compare how strain varied throughout the strain field distribution. A summary of the tissue strain distribution analysis protocol is outlined in Figure 3.4.

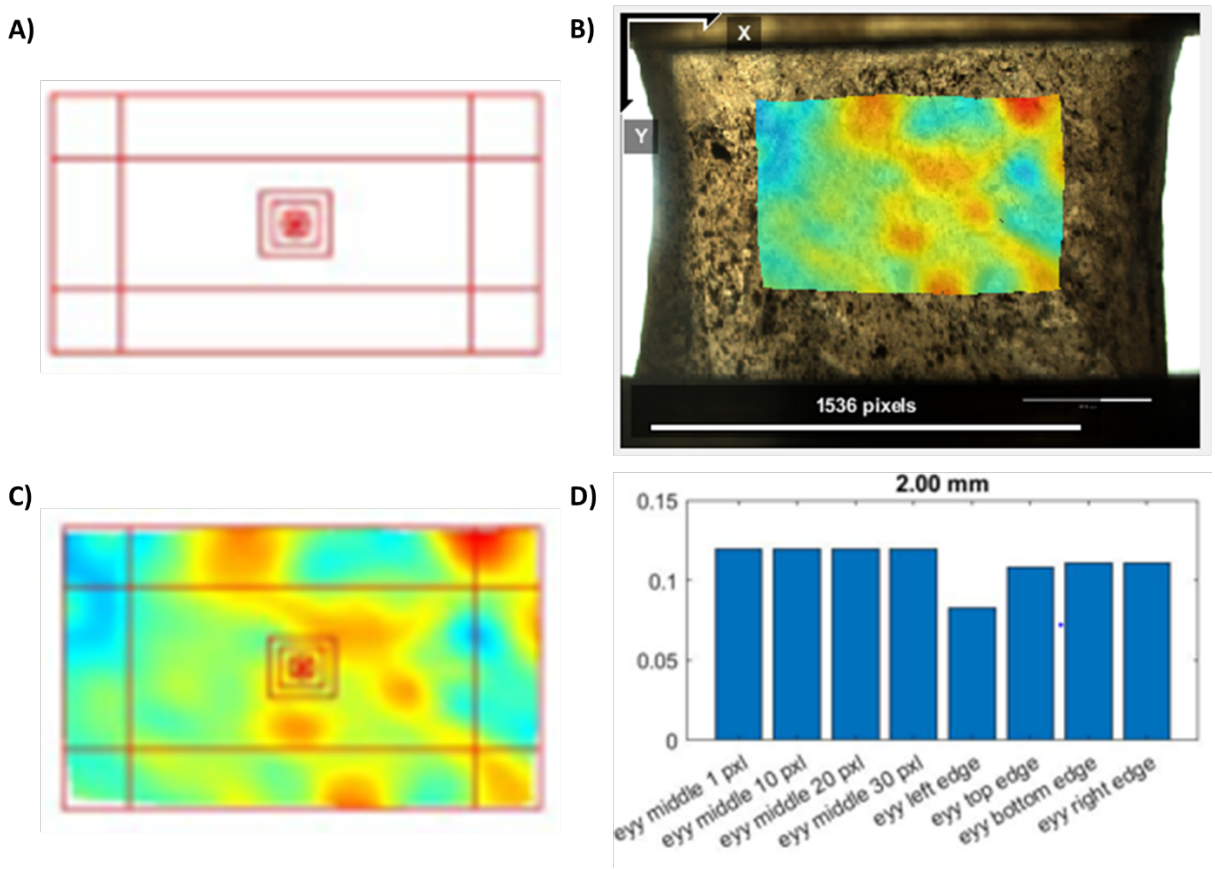

Figure 3.4. Regional tissue strain distribution analysis procedure. A) Illustration of different regions. **B)** Example of strain map used for analysis. **C)** Illustration of how the strain map was divided into different regions. **D)** Median strain values of each region plotted together for comparison.

3.2.6 Cell Strain Analysis

To verify that cells were being stretched using the prescribed loading method and to assess the relationship between cell strain and tissue strain, a protocol to non-directly measure cell strain was developed. Fluorescently labeled NIH-3T3 cells were seeded onto pericardium patches at a density of 100,000 cells per cm^2 and cultured for 24 hours to allow the cells to adhere. After 24 hours, the seeded patches were loaded into the 3D printed bioreactor as described above in *3.2.4 Sample Loading and Long-Term Viability Analysis*. Cell culture media was added to each well to maintain cell viability during the experiment. The bioreactor was placed onto an Evos Auto FL for imaging. Fluorescent images of the cells in the were taken

using a 10X objective (AMEP 4933, ThermoFisher Scientific) and a GFP Light Cube (AMEP 4951, ThermoFisher Scientific). Cell seeded samples were stretched from 0 mm to 2 mm in increments of 0.25 mm to match the increments of stretch that were applied during the strain validation protocol in *3.2.5 Tissue Stain Validation and Distribution Analysis*. Once images had been taken, the membranes of each cell were automatically identified and segmented using an open-source cell segmentation software (CellProfiler). The white segmented outlines were overlain on a blank black image. These binary images were then loaded into a custom software script (MATLAB) written specifically to identify each cell using the cell segmented boundaries, match each cell from image to image, and measure the changes in the cell morphological properties to evaluate cell strain. This process can be outlined in the following steps:

1. Load an image sequence in order of increased stretch from 0 mm to 2 mm
2. For each cell in the 0 mm image:
    - Trace the interior of the cell outline
    - Fill in the trace
    - Measure the morphological properties of the trace object
    - Identify potential cell matches in the subsequent image by comparing the morphological properties and location of the cell matches relative to the initial cell location in the original image
    - Select the cell that is the most likely match to initial cell based on the largest similarity in morphological properties
    - Fit an ellipse to the traces of the initial cell and the matched cell
    - Obtain measures of the long axis, short axis, and orientation of the ellipse for both cells

- Calculate points on the boundary of the ellipse for both cells
- Determine the deformation gradient matrix to transform the cell

Examples of how the cell strain analysis was conducted are outlined in Figure 3.5.

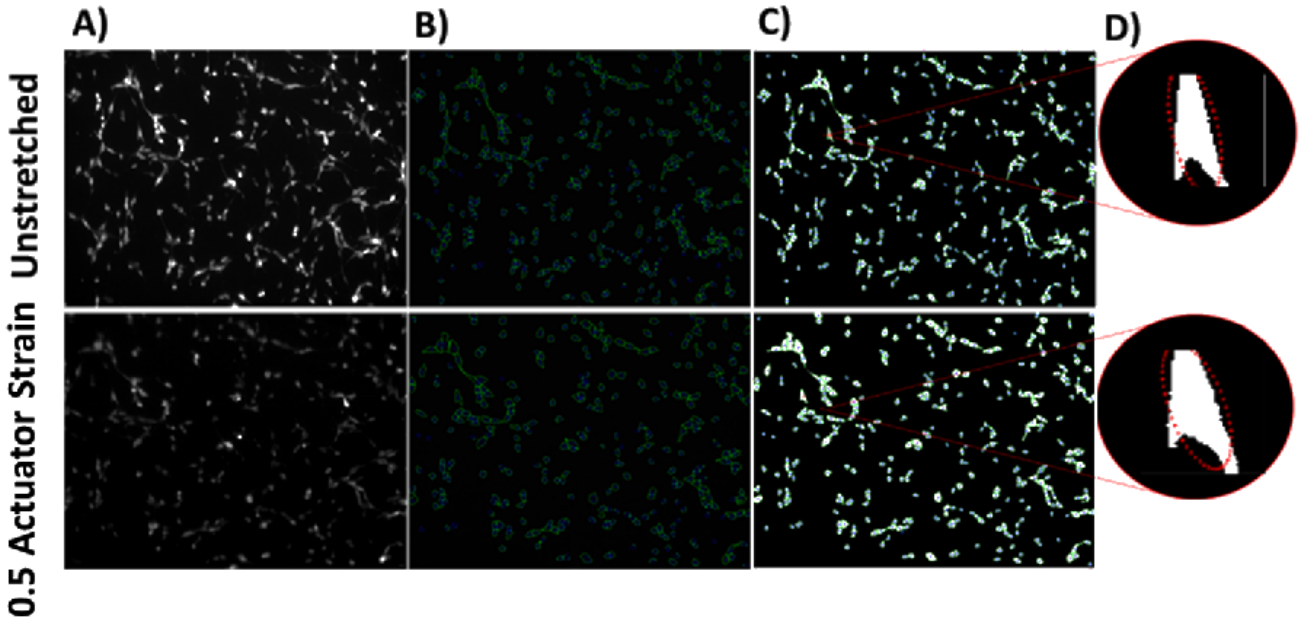

Figure 3.5. Overview of cell strain analysis. The unstretched image of a seeded sample is displayed in the top row and the corresponding sample is depicted in the bottom row. A) Original GFP images. B) Segmented cell outlines obtained using Cellprofiler. C) Cell objects created using segmented cell outlines. D) Selected cell overlain with best fit ellipse.

3.2.7 Statistical Analysis

Statistics for differences between the experimental groups in the cytotoxicity evaluation (n = 3 for alamarBlue™ assay measurements; n = 9 for cell count images) was calculated using ANOVA followed by Tukey's post hoc analysis. Statistics for the differences between the bioreactor experimental group and the control group were assessed using an independent student's t-test (n = 3).

3.3 Results

3.3.1 Cytotoxicity Assay

The results from the cytotoxicity assay revealed no significant cytotoxic effects of the media extract from the different bioreactor components on cell metabolic activity and cell count when compared to the positive control well (Figure 3.6). No noticeable difference in morphology between the cells cultured in the positive control extract wells, the BioMed Amber extract wells, the 316 stainless steel nut/screw extract wells, and pericardium tissue extract wells was observed.

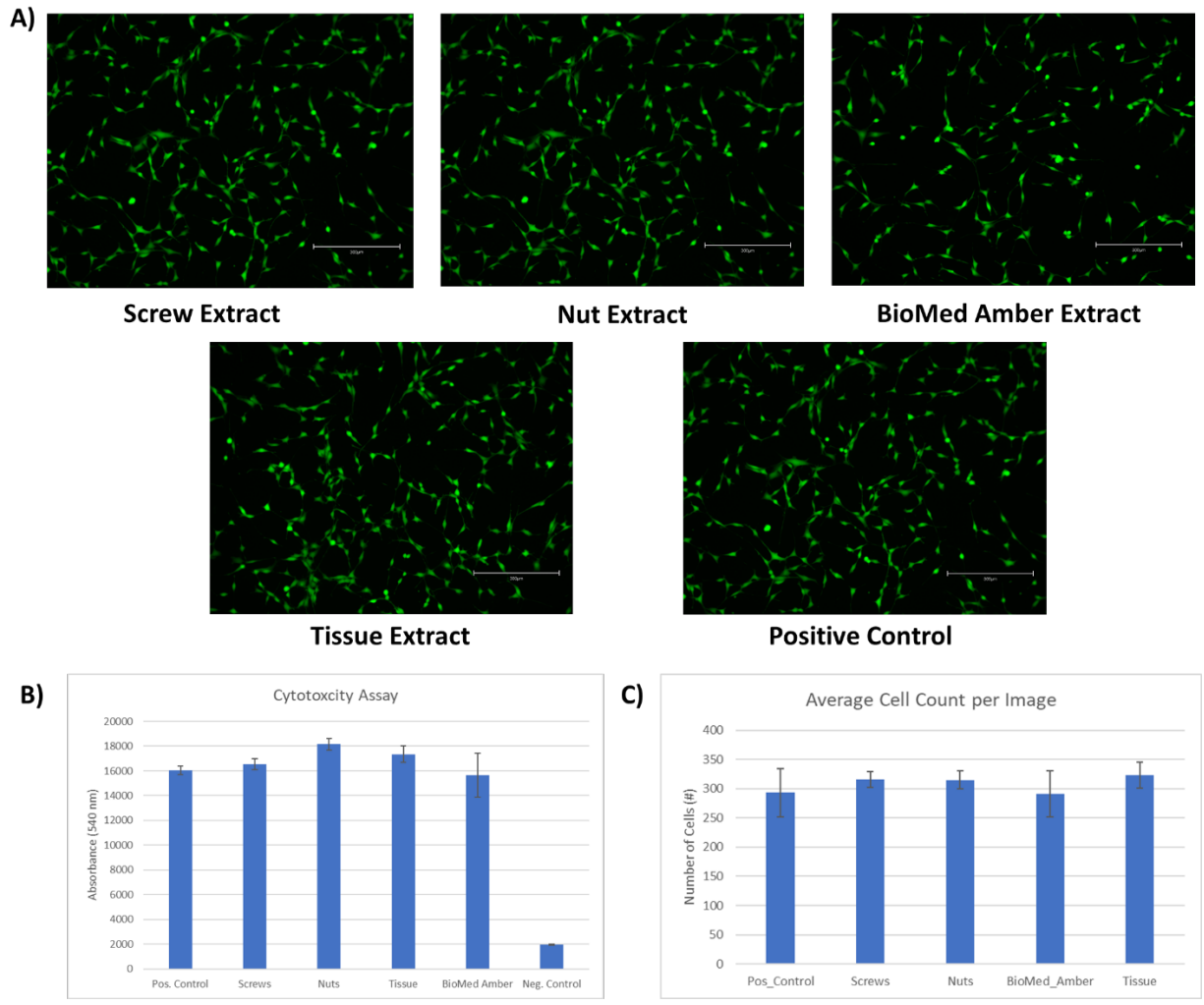

Figure 3.6. Cytotoxicity assay results. A) Representative Images from each experimental group depicting the cell morphology at the conclusion of the alamarBlue™ assay. **B)** Bar graphs showing results of the alamarBlue™

viability assay from the first trial that were assessed by measuring sample absorbance measurements (Fisher Scientific). **C)** Bar graphs depicting the average number of cells counted per image (n = 9).

3.3.2 Cell Viability Analysis

Fluorescent images of cell-seeded pericardium samples demonstrated cell viability at day 12 (Figure 3.7). Cellprofiler analysis demonstrated the number of counted cells in the pericardium sample images was significantly lower ($p < 0.05$) and analysis of mean vector length revealed that the cells seeded on the pericardium were more aligned ($p < 0.001$).

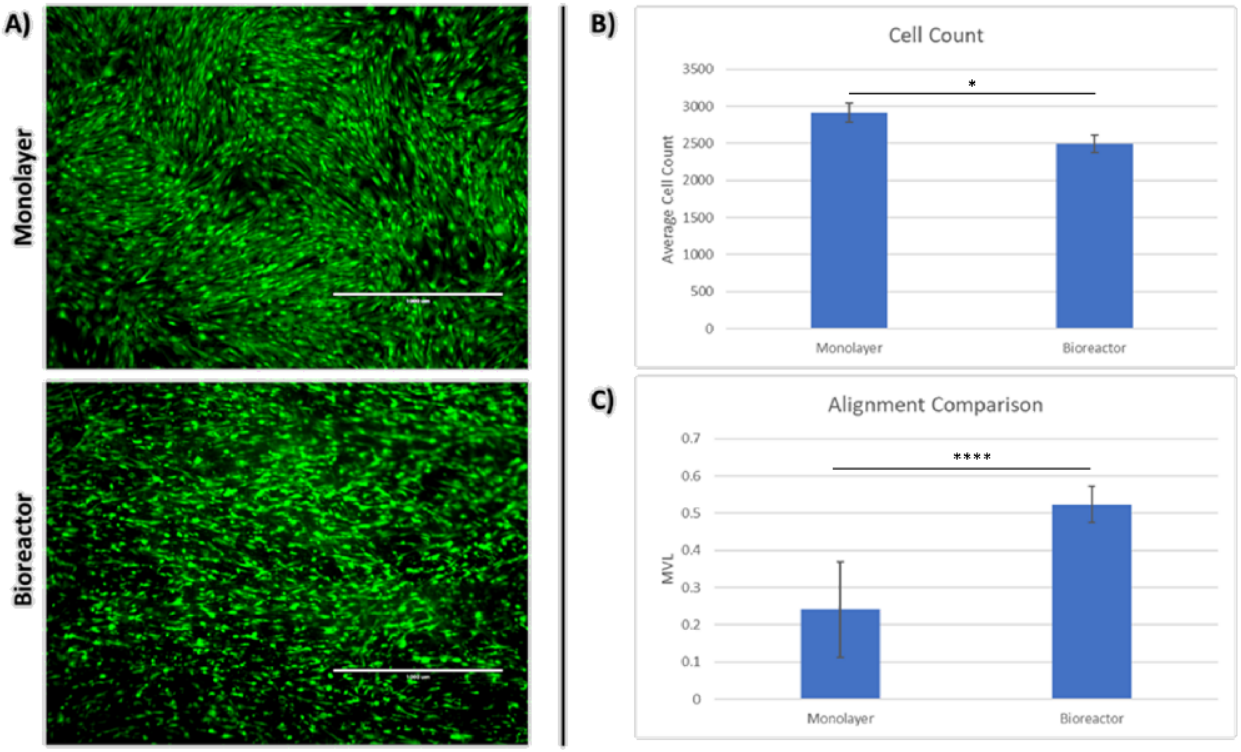

Figure 3.7. Long-term cell viability results. A) Representative Images from each experimental group depicting the cell morphology at the conclusion of the Long-Term Cell Viability Assay. **B-C)**. Bar graphs showing results of the average number of cells counted per image and the alignment of the cells (n = 3).

3.3.3 Tissue Strain Analysis

Tissue strain analysis was performed on each sample in the bioreactor for a total of 6 samples per experiment. The experiment was performed 4 times to collect 4 independent tissue

strain analysis datasets. For each experiment, the average median strain and standard deviation of the samples was calculated and plotted as a function of actuator strain (Figure 3.8a).

The results of this analysis demonstrated that there was variability between samples but the average strain values between experiments were not significantly different, indicating overall repeatability despite having slight individual differences between samples. The relationship between tissue strain and actuator strain was non-linear, following a logarithmic relationship where tissue strain increased quickly as a function of actuator strain and then gradually tapered off.

Regional strain analysis was conducted on each sample for each trial as described in *3.2.5 Tissue Stain Validation and Distribution Analysis*, providing a total of 24 samples analyzed (Figure 3.8b). A comparison of the regions revealed that median strain values did not differ significantly; however, the regions on the edges of the tissue sample did show a trend of having higher strains than the regions at the center of the tissue sample.

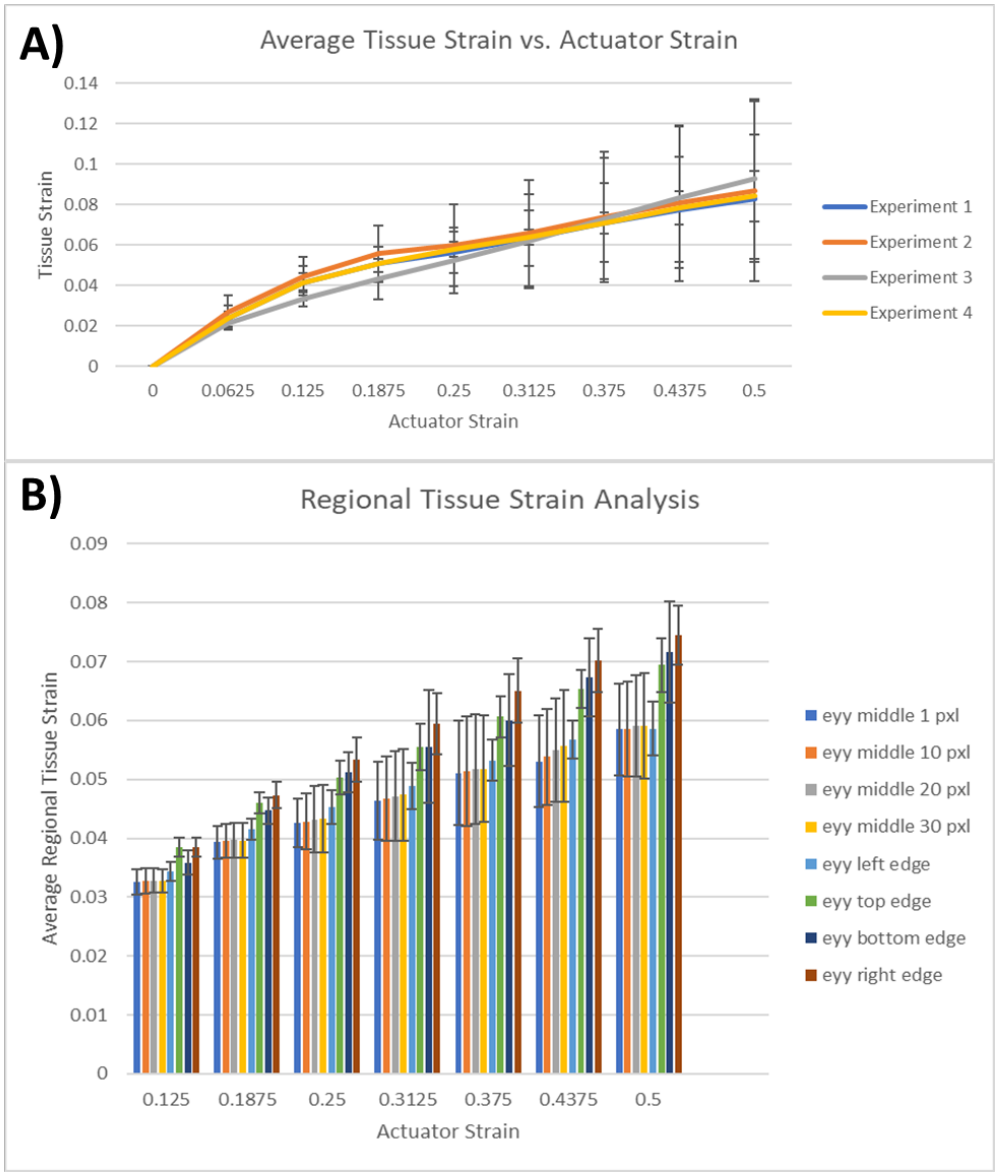

Figure 3.8. Tissue strain analysis results. A) Average median tissue strain values as a function of actuator strain. Values are plotted as the average median value and standard deviation for the strain field in the direction of stretch at each increment of actuator strain. The experiment was repeated 4 times independently to access repeatability. **B) Regional strain analysis.** Strain fields in the direction of linear actuation were calculated using ncorr. These fields were segmented into 7 different regions as defined in *3.2.5 Tissue Stain Validation and Distribution Analysis*: a single pixel center region (middle 1 pxl), a 10 pixel x 10 pixel center region (middle 10 pxl), a 20 pixel x 20 pixel center region (middle 20 pxl), a 30 pixel x 30 pixel center region (middle 30 pxl), a left rectangular region that covered the length of the left side by 20 pixels deep (left edge), a right rectangular region that covered the length of the right side by 20 pixels deep (right edge), a top right rectangular region that covered the length of the top side by 20 pixels deep (top edge), and a bottom rectangular region that covered the length of the bottom side by 20 pixels deep (bottom edge). For each experiment, the average strain and standard deviation for 6 samples for each region was calculated and reported at each increment of actuator strain.

3.3.4 Cell Strain Analysis

Results from the cell strain analysis demonstrated large variability in the measured cell strain (Figure 3.9). An examination of the cell morphological properties revealed that the measured cell strain was not impacted by orientation, size, or eccentricity. On average, cell strain increased exponentially.

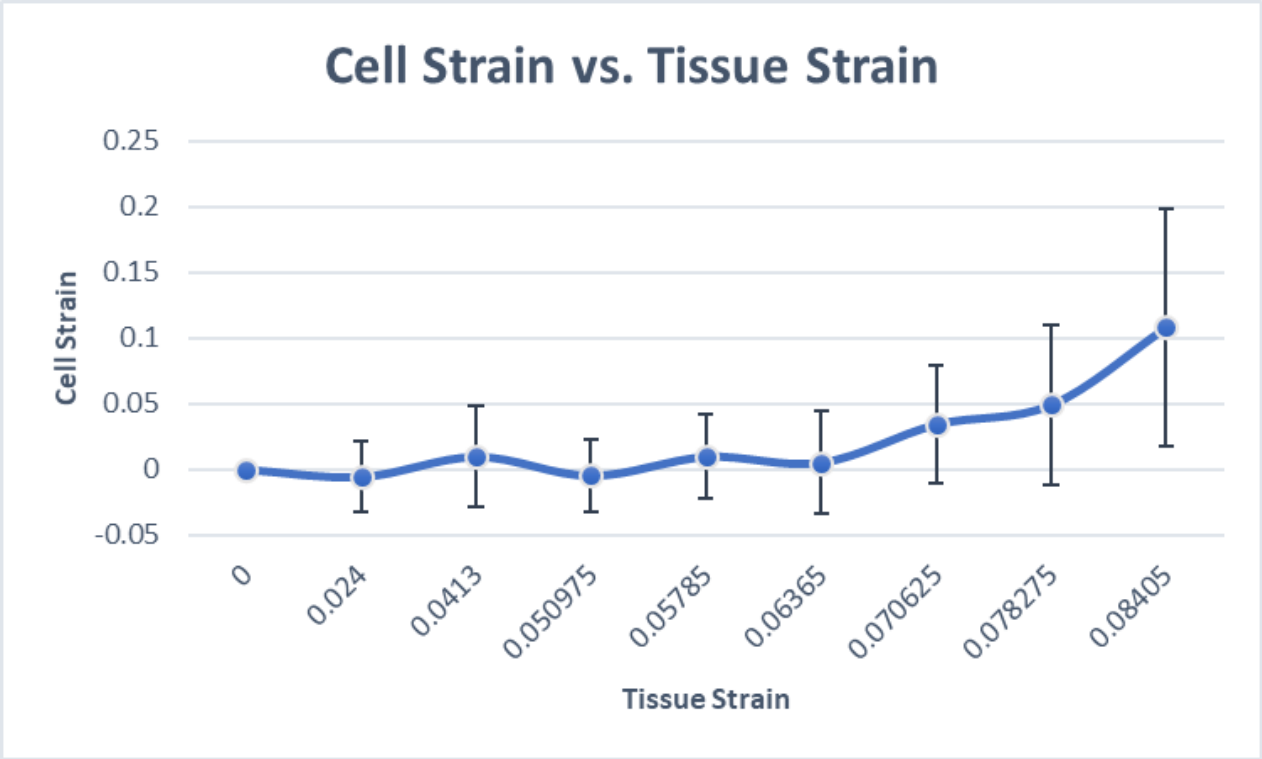

Figure 3.9. Cell strain analysis results. Average Cell strain plotted against the actuator strain.

3.4 Discussion

In this study, we presented a novel 3D printed bioreactor for conducting cellular mechanobiology studies. This system offers several advantages over other alternatives that are currently available. The first is that the use of 3D printing also enables this system to be easily modified to accommodate any design changes that may be required for a different experimental

setup. Second, this system provides the user with the ability to image cells in culture during mechanobiology studies in real time. Here, we utilized a fluorescently labeled cell line (GFP-labeled NIH-3T3s) and we were able to obtain high resolution images of cells at 10X that could be utilized for morphological analysis, strain analysis, and viability assessment. Third, we demonstrated that we could use this system to achieve tissue strains that are within physiologically relevant levels for collagenous tissues [20].

It is worth noting that while this system offers several advantages to other bioreactor systems for studying mechanobiology, there are several limitations that should be addressed in future iterations of the device in order to improve performance. First, the analysis of the tissue strains revealed a significant degree of variation between samples. This is likely due to human error involved with the sample loading protocol. With the current system setup described here, samples are loaded into the bioreactor by clamping one end of the tissue to the 6-well insert, stretching the sample out and then clamping it to the actuator arm without any method of verifying the initial level of tissue prestrain. Future iterations of this system should incorporate a pre-tensioning system which would allow the samples to be loaded into the bioreactor and then adjusted to ensure that all samples experience similar levels of strain during experimental studies.

Another limitation of this study is that the cell analysis was conducted on images that were captured using a standard fluorescence microscope. Furthermore, the samples that were imaged were 3D in nature. Thus, when the samples were stretched, cells were shifted slightly into and out of focus, resulting in variation in the cell membrane measurements due to the experimental protocol. This issue was further exacerbated by diffuse background light from fluorescent cells that were not in the focal plane during imaging. While it is difficult to remedy

these issues when imaging cells in 3D, one simple way to improve the quality of the fluorescent images during an experiment would be to utilize a confocal microscope rather than a basic fluorescent microscope like the one used in this study. The use of a confocal would help filter out diffuse light from out-of-focus fluorescent cells, ensuring that any fluorescent signal that was captured was only from the cells that were being imaged within a consistent focal plane.

3.5 Conclusion

The bioreactor system presented here serves as a low-cost system for performing mechanobiological studies and enables real-time monitoring of strain at the tissue and cellular level. The 3D printed design can serve as an example of how 3D printing can be utilized in a research study to efficiently create custom components that can meet the specific experimental criteria for the study based on the researcher's domain knowledge, rather than being limited to rigid commercially available components.

3.6 References

1. Dahl, Kris Noel, Alexandre JS Ribeiro, and Jan Lammerding. "Nuclear shape, mechanics, and mechanotransduction." *Circulation research* 102.11 (2008): 1307-1318.

2. Jaalouk, Diana E., and Jan Lammerding. "Mechanotransduction gone awry." *Nature reviews Molecular cell biology* 10.1 (2009): 63-73.

3. Yamashiro, Yoshito, and Hiromi Yanagisawa. "The molecular mechanism of mechanotransduction in vascular homeostasis and disease." *Clinical Science* 134.17 (2020): 2399-2418.

4. Duscher, Dominik, et al. "Mechanotransduction and fibrosis." *Journal of biomechanics* 47.9 (2014): 1997-2005.

5. Kumar, Ashok, et al. "Loss of dystrophin causes aberrant mechanotransduction in skeletal muscle fibers." *The FASEB journal* 18.1 (2004): 102-113.

6. Papachroni, Katerina K., et al. "Mechanotransduction in osteoblast regulation and bone disease." *Trends in molecular medicine* 15.5 (2009): 208-216.

7. Chin, LiKang, et al. "Mechanotransduction in cancer." *Current opinion in chemical engineering* 11 (2016): 77-84.

8. Korossis, S. A., et al. "Bioreactors in tissue engineering." *Topics Tissue Eng* 2.8 (2005): 1-23.

9. Cook, Colin A., et al. "Characterization of a novel bioreactor system for 3D cellular mechanobiology studies." *Biotechnology and bioengineering* 113.8 (2016): 1825-1837.

10. Janvier, Adam J., Elizabeth Canty-Laird, and James R. Henstock. "A universal multi-platform 3D printed bioreactor chamber for tendon tissue engineering." *Journal of Tissue Engineering* 11 (2020): 2041731420942462.

11. Putame, Giovanni, et al. "Application of 3D printing technology for design and manufacturing of customized components for a mechanical stretching bioreactor." *Journal of Healthcare Engineering* 2019 (2019).

12. McGuire, Rachel, Ryan Borem, and Jeremy Mercuri. "The fabrication and characterization of a multi-laminate, angle-ply collagen patch for annulus fibrosus repair." *Journal of tissue engineering and regenerative medicine* 11.12 (2017): 3488-3493.

13. Costa, Alessandra, et al. "Biologic scaffolds." *Cold Spring Harbor perspectives in medicine* 7.9 (2017): a025676.

14. Badylak, Stephen F., Donald O. Freytes, and Thomas W. Gilbert. "Extracellular matrix as a biological scaffold material: Structure and function." *Acta biomaterialia* 5.1 (2009): 1-13.

15. McGuire, Rachel, Ryan Borem, and Jeremy Mercuri. "The fabrication and characterization of a multi-laminate, angle-ply collagen patch for annulus fibrosus repair." *Journal of tissue engineering and regenerative medicine* 11.12 (2017): 3488-3493.

16. Wallin, Richard F., and E. F. Arscott. "A practical guide to ISO 10993-5: Cytotoxicity." Medical Device and Diagnostic Industry 20 (1998): 96-98.

17. Lionello, Giacomo, Camille Sirieix, and Massimiliano Baleani. "An effective procedure to create a speckle pattern on biological soft tissue for digital image correlation measurements." *Journal of the mechanical behavior of biomedical materials* 39 (2014): 1-8.

18. Tavakoli, Javad, Dawn M. Elliott, and John J. Costi. "Structure and mechanical function of the inter-lamellar matrix of the annulus fibrosus in the disc." *Journal of Orthopaedic Research* 34.8 (2016): 1307-1315.

19. Maganaris, Constantinos N., Marco V. Narici, and Nicola Maffulli. "Biomechanics of the Achilles tendon." *Disability and rehabilitation* 30.20-22 (2008): 1542-1547.

20. Magnusson, S. Peter, et al. "Human tendon behaviour and adaptation, in vivo." *The Journal of physiology* 586.1 (2008): 71-81.

CHAPTER 4

Machine Learning Model for Detecting Masked Hypertension in Young, Apparently Healthy Adults

4.1 Introduction

4.1.1 Background

Cardiovascular disease (CVD) is the leading cause of death globally, claiming approximately 17.9 million lives annually (1,2). Hypertension (HT) is one of the strongest risk factors for CVD and is associated with coronary disease, left ventricular hypertrophy, valvular heart disease, cardiac arrhythmias, cerebral stroke, and renal failure (3). HT accounts for approximately 56% of all CVD-related deaths (10 million) and is incredibly prevalent, affecting an estimated 1.3 billion people worldwide (4). This number has been increasing and is expected to reach 1.56 billion deaths annually by the year 2025 due to multiple factors, including population aging, increased prevalence of chronic kidney disease (CKD), diabetes mellitus, and obesity, suboptimal clinical treatments, and poor adherence to treatment plans (5,6). HT is defined clinically as a blood pressure (BP) of 140/90 mmHg or higher and can be prevented or managed through lifestyle and pharmacological interventions (7,8).

While HT can be managed, it often remains untreated as several population studies have found 12.7% - 37.3% of all cases of HT were not diagnosed clinically (9-11). This is due in part to a subset of these patients (10% of the general population) having normotensive BP measurements within the clinic while their out-of-clinic BP as measured by ambulatory BP monitoring is actually elevated to the point of being considered hypertensive (12). This condition has been classified as masked hypertension (MHT) and has been shown to have equal, if not

increased, risk for adverse cardiovascular morbidity due to the lack of any clinical diagnosis and corresponding clinical intervention (13-15). Furthermore, this condition has been associated with increased organ damage and altered cardiovascular dysfunction and structural changes (15,16).

One recent meta-analysis has suggested that nearly one in three patients who have normotensive office blood pressure (OBP) measurements have MHT and while this condition is more commonly present in older populations, MHT has even been identified in young and apparently healthy populations (15). Other studies have also reported similar findings, with one study reporting that approximately 11% of children under the age of 15 had MHT (17) and another study reported the prevalence of MHT in young to middle-aged adults (44 ± 19 years of age) to be 23% (18). Other studies found MHT present in populations that appeared to be in peak physical condition, such as endurance runners and professional soccer players (16,19).

4.1.2 Clinical Need

These studies demonstrate the need to monitor the out-of-office BP of the general population in order to diagnose and treat MHT in a timely and effective manner. The most common methods for detecting MHT are ambulatory BP monitoring (ABPM) and home BP monitoring (HBPM), however both of these methods come with significant drawbacks (20,21). HBPM, while convenient and easy to obtain, has been shown to have a reduced ability to detect MHT when compared to ABPM and research published by Stergiou et al., suggests that HBPM should only be used in conjunction with ABPM to detect MHT (22). ABPM, on the other hand, has been shown to successfully detect MHT with a high degree of accuracy; however, it requires the use of cumbersome equipment that may not be available to certain population groups,

particularly in children or in populations in low-and-middle income countries (LMICs) (23-26). Thus, the feasibility of ABPM for population-level detection of MHT is unknown (26).

4.1.3 Study Objective

One potential solution to the limited availability of ABPM in LMICs would be to develop a method for assessing risk for MHT based on clinical measurements obtained from a single outpatient visit. This could serve as a preliminary screening method that would allow for the patients most at risk for MHT to be identified while reducing the need for all patients to undergo ABPM. Patients that are then classified as being at risk for MHT could then undergo ABPM to confirm the presence of MHT and the need for further medical and lifestyle intervention to prevent the advent of CVD.

Several studies have recently developed machine learning (ML) models to predict adverse cardiovascular events such as coronary heart disease, heart failure, and stroke that have shown potential to assist clinicians in early disease detection and diagnosis (27, 28). These models evaluate clinical features to determine which patients are most at risk for these events using a combination of statistical methods and computational algorithms that can be automatically fine-tuned to these specific applications based on the input data. Researchers have found that these data-driven models can outperform traditional mechanistic models in applications involving a multitude of different variables due to their inherent ability to capture the non-linear relationships between these features and the variable that is being predicted (28-30). ML models are also useful for establishing a predictive model in which an experimentally validated mechanistic model is not readily available. Thus, a ML model that could potentially be used to create a method for predicting a patient's risk for MHT and identifying which patients in

LMICs are most at-risk for MHT and require ABPM to receive a definitive diagnosis. However, the feasibility of such a model remains unknown. Therefore, the goal of this study was to develop a ML that could be utilized in a LMIC setting to detect patients with MHT in the hopes of improving MHT detection, diagnosis, and treatment in a young and relatively healthy population.

4.2 Materials and Methods

4.2.1 Model Overview

Figure 4.1 depicts the process implemented during model development. The original dataset was obtained from study collaborators and contained health records with 420 different features from a cohort of 1,202 de-identified patients. This dataset was preprocessed prior to modeling by cleaning the data, performing feature engineering, imputing missing values, and scaling the continuous features. The preprocessed dataset was then split into a training set for constructing the ML model and a testing set for validating the model. Automatic feature selection was performed using the BorutaSHAP algorithm to identify the relevant features in the dataset and eliminate the unnecessary features. Twelve popular classifiers were trained on this optimized dataset. The performance of these classifiers was evaluated, and the five best-performing models were selected as base classifiers to using in a stacking classifier. The performance of the resulting stacking classifier was then compared to the performance of each base classifier. The best classifier was then tuned and the results of the final model were reported. Shapley Additive exPlanations (SHAP) values were used to investigate the individual contributes of each feature to the model predictions and partial dependence plots (PDPs) were created to assess the relationship between each feature and the predicted outcome.

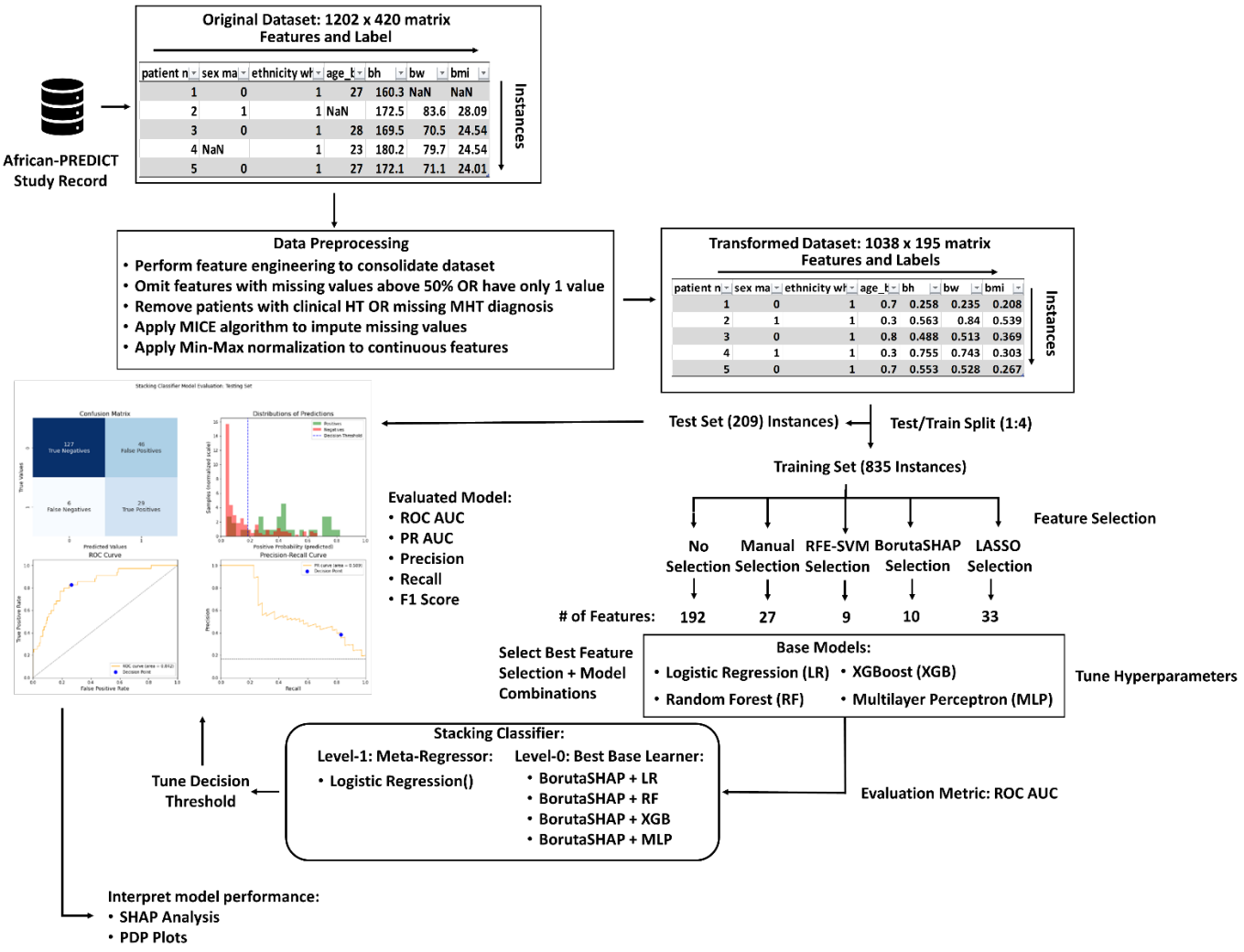

Figure 4.1. Overview of model development process. The original dataset was imported as a 1202 x 420 matrix. It underwent preprocessing that consisted of feature engineering, data cleaning, imputation, and scaling to make a transformed dataset that was a 1033 x 192 matrix. The data was split into testing and training sets using a 1:4 split ratio. 5 feature selection methods were evaluated on 4 different base ML models and a 5th ensemble ML model. The best performing model was evaluated and SHAP analysis was using in conjunction with PDP plots to interpret the model performance.

4.2.2 Original Dataset

The dataset used in this study was derived from the African-PREDICT study, a study aimed at preemptively identifying cardiovascular disease in young patients from South Africa, a country currently classified as a LMIC (31). The study design and research methods used to

collect the data have been described previously (32). All measurements were conducted in accordance with current gold standard methodologies and were carried out by trained research nurses, postgraduate students, and academic staff. The inclusion and exclusion criteria of the study as well as the patient data is summarized in Figure 4.2.

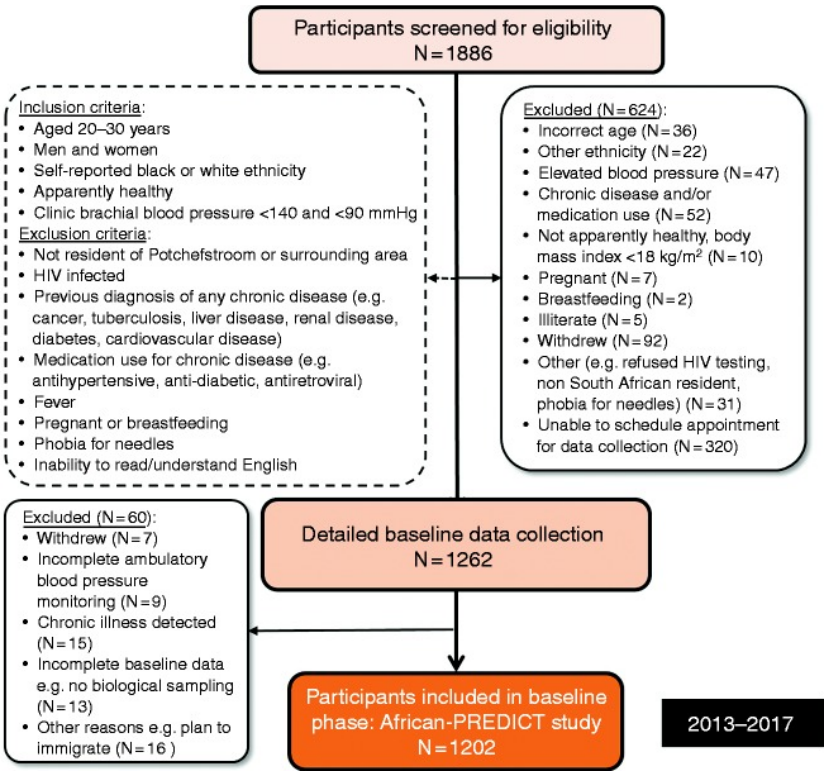

Figure 4.2. Summary of participant inclusion and data collection in the African PREDICT study. A total of 1,886 volunteers were screened for study eligibility. 624 were excluded by study criteria. 60 more were excluded during the study for multiple reason, resulting in an overall sample size of 1,202 participants.

In brief, a total of 1202 black (N = 606) and white (N = 596) young men and women (aged 20-30 years) in South Africa were screened to be healthy and clinically normotensive and without complicating factors such as pregnancy and previous diagnosis of any chronic diseases. Different clinical measures relevant to hypertension were collected from each patient. Each of the measured features could be broadly categorized into the following groups: 1. Questionnaire

data (e.g. medical history, social status, diet, psychosocial profile); 2. Biological data (e.g. serum, plasma, buffy coat, 24-hour urine); 3. Biomarker data (e.g. lipids, glucose, multiplex cytokines, RAS-Fingerprint, adipokines, oxidative stress, nitric oxide and coagulation markers, urinary sodium, metabolomics, proteomics); 4. Body composition data (physical measurements) ; 5. Physical activity (Vigorous, moderate, and sedentary activity levels); 6. BP (office, 24-hour, central, reactivity); and 7. Target organ damage (arterial stiffness, carotid wall thickness, electrocardiography, echocardiography, retinal microvasculature, renal function). The compiled dataset for each patient included a total of 420 different features (Appendix A-1.1. Complete List of African PREDICT Measured Features).

4.2.3 Data Preprocessing

The de-identified patient data were uploaded and stored as a data frame within a script written in Python (Python 3.8.10) using an open-source integrated development environment (IDE) (Spyder 5.3.3). The original data frame was a 1202 x 420 matrix. To prepare the data for analysis, the entire dataset was preprocessed by cleaning the data, dropping irrelevant features, feature engineering, imputing any missing values in the remaining data, and then scaling the continuous features as needed.

The first step in cleaning the data was to drop from the data frame any instance (row) that was completely empty, any instance that corresponded to patients who were clinically diagnosed with hypertension, or any instance where the patient was missing a diagnosis of MHT. The next step was to eliminate any instances or features (columns) that were missing a significant number of values (> 10%). Finally, any features that contained only 1 unique value were dropped from the dataset. Cleaning the data reduced the dataset to 1038 patients and 316 features.

After cleaning the data, the next preprocessing step was to drop irrelevant features and feature engineering to reduce the number of features in the dataset. The purpose of this step was two-fold: first, reducing the number of features in the dataset by removing unnecessary features helps to improve model performance and reduce computational cost (33), and second, this step helps to improve the overall interpretability of the model. Several features in the original dataset were not considered relevant to the model based on the intended use case for the model. These features could be broadly classified as socio-economic features, ambulatory blood pressure measurement features, and certain questionnaire features. The list of features manually removed from the dataset is listed in Appendix A-1.2. List of Features Dropped from the Original African PREDICT Dataset. Next, a literature review of the risk factors associated with MHT was conducted and several categorical features that were not included in the dataset were identified (34-37). Of these, several categorical features were identified that could be determined from numerical features in the dataset. Given that these categorical features could potentially help improve the performance of certain ML models and potentially serve as important predictors of MHT, these features were created using the criteria outlined in Table 4.1 and included as additional features in the data frame.

Table 4.1. List of engineered categorical features and corresponding definitions.

Feature Name	Variable Type	Classification	Definition
Prehypertensive ('prehypertensive')	Categorical	0, healthy; 1, diseased	140 mmHg > SBP > 119 mmHg OR 90 mmHg > DBP > 79 mmHg
Obese ('Obese')	Categorical	0, healthy; 1, diseased	bmi > 25
Left Ventricular Hypertrophy ('lvh')	Categorical	0, healthy; 1, diseased	Left Ventricular Mass Index > 115 g/m^2 AND Sex is Male OR Left Ventricular Mass Index > 95 g/m2 AND Sex is Female
Physically Inactive ('sedentary')	Categorical	0, healthy; 1, diseased	Sum of total vigourous, moderate, and travel MET minutes < 600
Smoker ('smoker')	Categorical	0, healthy; 1, diseased	Self-identified smoker AND cotinine > 10
Excessive Alcohol Use ('excessive_alcohol')	Categorical	0, healthy; 1, diseased	Self-identified alcohol consumption AND ggt > 48
Dyslipidemic ('dislipidemic')	Categorical	0, healthy; 1, diseased	LDL > 3.4

Finally, in this dataset there were several features with multiple repeated measurements of the same feature, in which case the repeated measurements were averaged. This process was performed on a total of 44 different measurements to bring the total number of remaining features in the dataset to 195. Several nominal categorical features were converted to binary categorical features for the purposes of simplifying the model as well. For example, information regarding the hypertensive status of a patient's parent was encoded from 5 categories (0 – No, 1 – Don't Know, 2 – Yes, under 60, 3 – Yes, over 60, 4 – Yes, don't know) to 2 categories (0 – No, 1 – Yes). A complete list of the transformed categorical variables is presented in Appendix A-1.3. Complete List of Transformed African PREDICT Categorical Features).

Once the dataset had been cleaned and feature engineering had been conducted to reduce the number of features in the dataset, any remaining missing values in the dataset were filled in using imputation. For this study, Multivariate Imputation by Chained Equations (MICE) algorithm was used to impute missing values in the dataset (40). The MICE algorithm assumes that the missing values are Missing At Random (MAR) without any underlying relationship between the instances where features are missing and can summarized in the following steps:

1. For each feature in the dataset, replace the missing values in that feature set with the mean of that feature.
2. Revert the imputed values of one feature back to missing values.
3. The missing value is then treated as a dependent variable while the other features in the instance are treated as independent variables. A regression model is then constructed from these variables using the values from each instance.
4. The regression model is then used to calculate the missing value for the feature in question. This value is the new imputed feature value.

5. A new feature that had missing values is then selected and this process is repeated.
6. This step is then repeated for a predefined number of cycles, with the imputed values for each feature being updated with each cycle.

A more detailed explanation of the algorithm has been described by Azur et al., (41). For this work, MICE imputation was implemented using the iterative imputer class in scikit-learn. More details of how this algorithm was implemented in the scikit-learn library are described elsewhere (38, 40).

After imputing any missing values in the dataset, the next step in the data preprocessing stage was to scale the continuous features to increase model efficiency and prevent feature bias from occurring in the ML algorithms that weigh feature importance based on Euclidean distance measures. This is a crucial step for those ML algorithms, as features of high magnitude can be biased toward higher weights than features of lower magnitude (42). The two most common methods for scaling features include standardization and normalization. Standardization involves centering the values of each feature around the mean value of that feature with respect to the standard deviation of the feature values. Mathematically, this can be expressed as:

$$X' = \frac{X - \mu}{\sigma}, \tag{4.1}$$

Where μ represents the mean and σ the standard deviation the feature set values. This scaling technique is useful when the feature set is normally distributed and offers the advantage of retaining the relationship between data points and being less sensitive to outliers. Normalization, on the other hand, adjusts the range of values in the feature set to a common scale to prevent features with different magnitudes from inadvertently biasing the ML model towards a

particular feature. Using normalization, each feature is scaled to a value between 0 and 1 using equation 2.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min},} \tag{4.2}$$

This technique offers the advantage of retaining the original feature distribution after transformation and is therefore useful when the data is not normally distributed. The drawback to this method is that it can be sensitive to any outliers in the data.

The distribution of each continuous feature in the dataset was evaluated using a Shapiro-Wilk test for normality. Based on the results of this analysis, it was determined that normalization was the most appropriate method for scaling the data (Appendix A-1.4. Continuous Feature Normality Test Results). Scaling the dataset was implemented using a *MinMaxScaler* class from the *scikit-learn.preprocessing* library (scikit-learn v. 1:3:0) (38).

4.2.4 Feature Selection

Feature selection is a crucial step in preparing a dataset for training a ML model (44). This process involves reducing the number of features in a dataset in order to reduce the complexity of the ML model and increase model interpretability, increase the speed of training the model, and improve model performance by eliminating features that are statistically redundant or do not contribute to the significantly to making a model prediction. In practice, there are several methods for performing features selection. These methods are either manual or automated and can be combined to determine the optimal set of features for a ML problem.

Manual feature selection involves choosing the relevant features to include based on domain knowledge or experience while automatic feature selection utilizes a variety of mathematical calculations and ML algorithms to select relevant features based on the raw values in the dataset. Automatic feature selection is generally classified as one of three categories based on the underlying methodology of the technique that is employed. These classes include filter methods, wrapper methods, and embedded methods. Briefly, filter methods use statistical tests such as the Chi-Squared test or the fisher's score to evaluate the relevance of each feature based on the univariate statistics of that feature. Common filter methods include information gain, the Chi-Squared test, Pearson's correlation coefficient, fisher's score, variance threshold, mean absolute distance (MAD), and Information Gain (46). Wrapper methods employ a specific ML algorithm to evaluate the importance of each feature relative to a specific evaluation metric. These methods can iteratively evaluate all possible subsets of features and can therefore assess complex non-linear relationships between features that are not inherently obvious using basic statistical tests alone. Thus, they often result in selecting feature sets that lead to models that have better overall performances than feature sets created by filter methods. However, these methods are very computationally expensive. Popular embedded feature selection techniques include forward feature selection, backward feature elimination, exhaustive feature selection, recursive feature elimination, and the Boruta feature selection method (45,46). Embedded feature selection methods combine the benefits of both filter feature selection methods and embedded feature selection methods in that they account for complex interactions between features but maintain a reasonable computational cost. Two common examples of embedded methods include LASSO regularization and random forest importance (46, 47). The choice of which feature

selection method to use depends on the user needs for model performance and computational cost.

For this work, a total of 3 different automatic feature selection methods were investigated alongside manual feature selection utilizing domain knowledge and the case in which all features were included. The 3 automatic feature selection methods that were investigated included: 1) a wrapper feature selection method known as recursive feature elimination which utilized a support vector machine classifier as a wrapper (RFE-SVM), 2) a wrapper feature selection method known as Boruta-SHAP which utilized the Boruta algorithm wrapped around a extreme gradient boosted classifier (XGB) to select features based on their Shapley Additive exPlanation (SHAP) values, and 3) Least Absolute Shrinkage and Selection Operator (LASSO) regression, an embedded feature selection method that uses an L1 regularization technique to eliminate unimportant features by shrinking the coefficients of these features to zero and effectively removing them from the model.

4.2.4.1 Entire Feature Set

A complete list of the features included in the final feature set is detailed in Appendix A-1.3. Complete List of Transformed African PREDICT Categorical Features.

4.2.4.2 Manual Feature Selection

A literature review was conducted to determine which features in the original dataset were correlated with the incidence of MHT. Several papers highlighted the following features from our dataset as relevant to MHT: sex (34,48-56), ethnicity (34,55,56), age (48-50,53-56), body mass index (BMI)(49,53-56), waist hip ratio (WHR)(49,54), blood lipid levels (Glucose

(49,55), Triglycerides (49,55), LDL (49,55), HDL (49,55), Total Cholesterol (49,55)), clinical measurements (systolic BP (49,51-56), diastolic BP (49,54-56), mean arterial pressure (MAP) (49), pulse pressure (PP) (49), heart rate (54)), lifestyle factors (Smoking (34,48,50,53,54,56), alcohol consumption (48,50,51,53,54), activity level (48,50)), and personal/family medical history (comorbidities such as prehypertension (50,53,56), obesity (48,50), stroke (48,50), or diabetes mellitus (48,50,54-56), family history of hypertension), and echocardiogram measures (left ventricular mass (34,48,55), left ventricular hypertrophy (LVH), (34,54), intima-media carotid thickness (48)).

4.2.4.3 Automatic Feature Selection Technique #1: Recursive Feature Elimination

RFE-SVM is a popular wrapper feature selection method that has been used in multiple previous studies to select important features using the backward feature elimination algorithm in conjunction with a SVM linear classifier (80). The algorithm operates according to the following general steps:

1. The algorithm fits an SVM model to the training set.
2. The model performance is evaluated according to a user-specified evaluation metric and recorded.
3. During training, each feature is assigned a weight corresponding to its relative importance in the model.
4. The features are ranked by weight in descending order from largest to smallest weight.
5. The features with the smallest weight are eliminated from the feature set.
6. The process is repeated, and the model is re-trained on the reduced feature set.

7. This process continues until only 1 feature remains.
8. The features set with the highest evaluate metric score is the selected feature set.

SVM classifiers are a class of generalized linear classifiers that operate under the guiding principle of simultaneously minimizing classification error and maximizing the geometric margin by identifying the hyperplane that maximizes the Euclidian distance between the plane and the dataset features in a multidimensional space. A more detailed discussion of SVM is provided by Brereton and Gavin (81).

For this study, the Recursive Feature Elimination was implemented using the *RFE* class in the *scikit-learn.feature_selection* library (scikit-learn v. 1:3:0).

4.2.4.4 Automatic Feature Selection Technique #2: Boruta-SHAP

The 2nd automatic feature selection technique that was evaluated for this study was the Boruta-SHAP technique. Briefly, the Boruta method determines feature importance by comparing the relevance of real features in the dataset to randomized copies of the features. The algorithm can be outlined as follows:

1. Create a copy of each feature in the dataset and shuffle the arrangement of the instances within these copy features.
2. Fit a random forest classifier model to the new dataset with the original features and the shuffled copy features.
3. Rank the features of the random forest classifier using the inherent feature importance algorithm within the random forest classifier.

4. Set a threshold value that corresponds to the highest importance score from the created copy features.
5. Perform a two-sided T-test of equality on all of the features.
6. Remove features that are significantly lower than the threshold as they are deemed “unimportant” and features that are significantly higher are noted as “important”.
7. Remove all copy features and repeat the process until an importance has been assigned to each feature or the algorithm has reached a set number of iterations.

A more detailed overview of the Boruta algorithm has been described previously by Kursa et al., (58). For this study, the standard Boruta-SHAP algorithm was slightly modified to utilize SHAP values as the metric for importance scoring. This metric was selected based on previous research which has evaluated different automatic feature selection techniques and found that feature selection techniques based on SHAP values are more stable (less lightly to alter selected features for different permutations of the training data) and can improve overall model performance compared to other automatic feature selection techniques (47,57). Briefly, SHAP values are calculated by approximating each prediction *f(x)* in a dataset with a linear function *g(x)*, defined as:

$$g(z') = \varphi_0 + \sum_{i=1}^{M} \varphi_i z_i' \tag{4.3}$$

Where $z' \in \{0,1\}^z$, M is the number of explanatory variables being evaluated by the algorithm, and $\varphi_i \in \mathbb{R}$, are the Shapely values determined by:

$$\varphi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!} [f_x(z') - f_x(z' \backslash i)\,] \quad (4.4)$$

Where *f* is the base ML model that the algorithm is wrapped around, *x* is the number of available variables, and $f_x(z') - f_x(z' \backslash i)$ is the deviation of Shapley values from their mean for each single prediction; i.e. the contribution of the i-th variable. For a more detailed discussion of Shapley values, please refer to Hart, 1989 (59).

The Boruta-SHAP feature selection algorithm was implemented in python using the *BorutaSHAP* function *BorutaSHAP* library (BorutaSHAP v. 1.0.0) (60). A *Random Forest Classifier* class from the *scikit-learn.ensemble* library (scikit-learn v. 1.3.0) was fit to a training dataset containing all the features and implemented as the wrapper for the *BortuaSHAP* instance.

4.2.4.5 Automatic Feature Selection Technique #3: LASSO Regression

The 3rd and final automatic feature selection method that was assessed in this study was LASSO regression. At a high level, LASSO regression performs feature selection by shrinking the coefficient of unimportant features in the regression model towards zero through the introduction of an L1 regularization penalty term. This process was first proposed by *Tibshirani et al.,* in 1995 and is implemented through the following steps:

1. Construct a logistic regression model as:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \varepsilon, \quad (4.5)$$

Where y is the predicted feature, β are the regression coefficient terms to be estimated using ML techniques, x are the input features, and ε represents the L1 regularization penalty term.

2. Define the ε term: L1 regularization is defined as the sum of the absolute values of the coefficients, β, in the regression function multiplied by a tuning parameter λ. This is expressed as:

$$\varepsilon = \lambda * (|\beta_0| + \cdots + |\beta_n|) \tag{4.6}$$

The tuning parameter λ is a user defined parameter that is commonly determined through hyperparameter tunning, as discussed in *2.6 Hyperparameter Tuning.*

3. Define the objective function for estimating the values of the coefficients, β. In this case, the objective function is to minimize the sum of squares:

$$\arg\min \{\sum_i (\beta_1 x_1 - y_i)^2 + \varepsilon\} \tag{4.7}$$

4. Implement an optimization algorithm such as Coordinate Descent to minimize the objective function.

5. As λ is increased, the degree of regularization is increased and more of the feature coefficients are set to 0. The features that have non-zero coefficients are considered to be important features for the model.

LASSO regression feature selection was implemented using the *LassoCV* class in the *scikit-learn.linear_model* library (scikit-learn v. 1:3:0). For further reading, refer to Randstam and Cook (82).

4.2.5 Machine Learning Models

The goal of this model was to detect patients that were at risk for MHT. Therefore, patient classification was the ML task chosen to be modeled using the African-PREDICT dataset. Several ML classifier algorithms were employed and evaluated for this purpose. Those algorithms included a: 1. Multivariate logistic regression (LR) classifier, 2. Random forest (RF) classifier, 3. Support vector machine (SVM) classifier, 4. K-Nearest Neighbors (KNN) Classifier, 5. naïve Bayes (NB) classifier, 6. Extreme Gradient Boosting (XGB) classifier, 7. Artificial Neural Network (ANN) classifier, 8. LightGBM (GBM) classifier, and 9. a stacking (STK) classifier. All models were implemented using the *scikit-learn* (scikit-learn v. 1:3:0), *xgboost* (xgboost v. 1:7:6), and *lightgbm* (lightgbm v. 3.3.4) libraries (38). A short explanation of each model functions and was implemented is provided in the following sections: *2.4.1 – 2.4.9*.

4.2.5.1 Logistic Regression Classifier

Logistic Regression is a statistical model that is commonly used to estimate the probability of a binary (2-outcome) event based on a set of independent continuous and categorical variables and variable weights (63). This algorithm is defined mathematically as the natural logarithm of the odds of an event occurring as a regression function of a predictor variable (feature) or set of predictor variables:

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{4.8}$$

Here *p* represents the probability of an event occurring (outcome = 1), β_0 is the intercept term, β_n represents the regression coefficient that quantifies the change in the natural logarithm of the probability of an event occurring when the corresponding predictor variable changes value. Equation 5 can be rearranged to solve directly for the probability of an event occurring:

$$\hat{p}(x_i) = \frac{e^{(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}{1+ e^{(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}} \tag{4.9}$$

The values of the regression coefficients are calculated through computational numerical methods where initial guesses for each coefficient are initialized and then iteratively refined to improve the fit of the regression until the coefficients converge to stable values that minimize the following cost function:

$$\min C \ \sum_{i=1}^{n}(-y_i \log(\hat{p}(x_i)) - (1 - y_i)\log(1 - \hat{p}(x_i))) + r(\beta) \tag{4.10}$$

Here n is the total number of instances in the dataset, y is the true value of the outcome variable being predicted for a particular instance, $\hat{p}(x_i)$ is the predicted probability for the same instance, and $r(\beta)$ is a regularization term that is used to introduce a penalty argument into the equation in order to generalize the model and reduce the risk of over-fitting the data. For this work, the logistic regression model was implemented using the *LogisticRegression* class in the *scikit-learn.linear_model* library (scikit-learn v. 1:3:0).

4.2.5.2 Random Forest Classifier

A Random Forest Classifier is a tree-based ensemble ML algorithm (64). It consists of multiple independent Decision Tree Classifier that collaborate together to make a prediction. More specifically, each Decision Tree Classifier in the Random Forest is trained on a randomized subset of features and instances in the entire training dataset to create a classifier that captures a certain aspect of the data. As more of these Decision Tree Classifiers are trained, the more they can capture different aspects of the data. Once the ensemble of Decision Tree Classifiers in the Random Forest has been trained, they each make outcome feature predictions on the entire original training dataset, with each prediction from a classifier acting as a "vote". The outcome feature prediction with a majority of the votes from the collection of individual Decision Tree Classifiers becomes the final outcome feature prediction for the Random Forest Classifier. Random Forest Classifiers can also report outcome feature predictions as probabilities by averaging the probability outcome from each underlying Decision Tree Classifier. The concept behind this process can be visualized below in Figure 4.3.

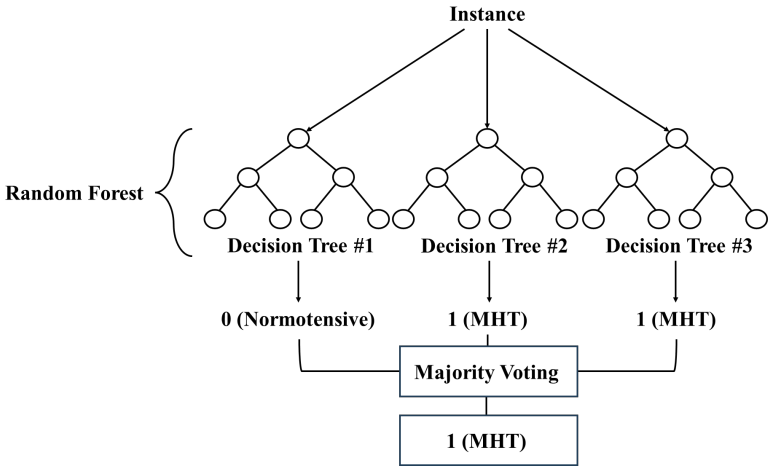

Figure 4.3. Random forest classifier schematic. The random forest classifier consists of multiple individual decision tree classifiers that make predictions on an instance. The prediction that is made the majority of the time is then selected as the overall prediction for the random forest.

In order to fully appreciate how the Random Forest Classifier operates, it is important to understand the underlying Decision Tree Classifier algorithm. Decision Tree Classifiers are intuitively easy to conceptualize. Each Decision Tree starts with a single base node that represents a single decision point that is then split into two branches depending on the output of that decision point. Each branch feeds into another node that represents another decision point. This process is repeated until a node that splits into a decision on the output feature is reached for each branch. An example of this process is shown in Figure 4.4.

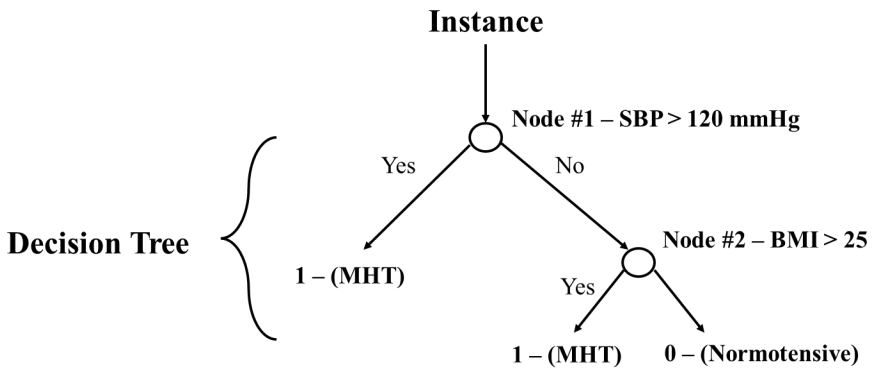

Figure 4.4. Example of a simple decision tree classifier. Decision trees make a prediction about an instance based on a series of decision nodes that pertain to a particular feature. The decision results in a binary decision of yes or no that then leads to more decision nodes that ultimately lead to a final decision of a predicted outcome.

The number of decisions, or splits, of the Decision Tree Classifier is determined by algorithmically evaluating all instances being evaluated at a given node. If the number of instances at the node do not all belong to the same class, then the node is split into two branches. The split value of the feature for the node is determined by selecting the value that maximizes a *goodness measure* such as Information gain, Gain Ratio, or Gini value. For more complete details, refer to a recent overview by S.B. Kotsiantis (65). For this study, the Random Forest Classifier was implemented using the *RandomForestClassifier* class in the *scikit-learn.ensemble* library (scikit-learn v. 1:3:0).

4.2.5.3 Extreme Gradient Boosting Classifier

XGBboost classifier models are another type of ensemble model based on decision tree classifiers. The primary difference between XGBoost classifiers and RF classifiers is the way in which the underlying decision tree classifiers are constructed. RF classifiers utilize a methodology known as bagging, which each of the decision tree classifiers in the ensemble are trained concurrently with random subsets of the training dataset. XGBoost classifiers, on the other hand, implement a strategy known as boosting, which decision tree classifiers in the

ensemble are trained sequentially so that each new iteration of decision tree classifiers can learn from the mistakes of the previous iteration of decision trees in an attempt to improve on overall model performance. Another aspect of XGBoost that differentiates them from RF classifiers is the incorporation of regularization terms in the model algorithm that help prevent the model from overfitting on the training data, allowing for a more generalize model. A more detailed explanation of the underlying algorithm has been presented previously by Tianqi *et al.,* (85). Here, the XGBoost classifier was implemented in this study using the *XGBClassifier* class in the *xgboost* library (xgboost v. 1:7:6).

4.2.5.4 Multilayer Perceptron Classifier

MLPs are a class of feed forward artificial neural networks (84). These neutral networks are comprised of many interconnected computation units called perceptrons (Figure 4.5a). Perceptrons take in input such as feature values and multiply those values by a specific weight. The sum of all the weighted inputs is applied to an activation function that results in an output value. This value can be fed into another perceptron in second layer along with any other output layers from other perceptrons in the first layer and the process is repeated for a user defined number of layers (Figure 4.5b). After the final layer, the output from the final layer of perceptrons is mapped to an output prediction. The MLP classifier was implemented in this study using the *MLPClassifier* class in the *sklearn.neural_network* library (scikit-learn v. 1:3:0).

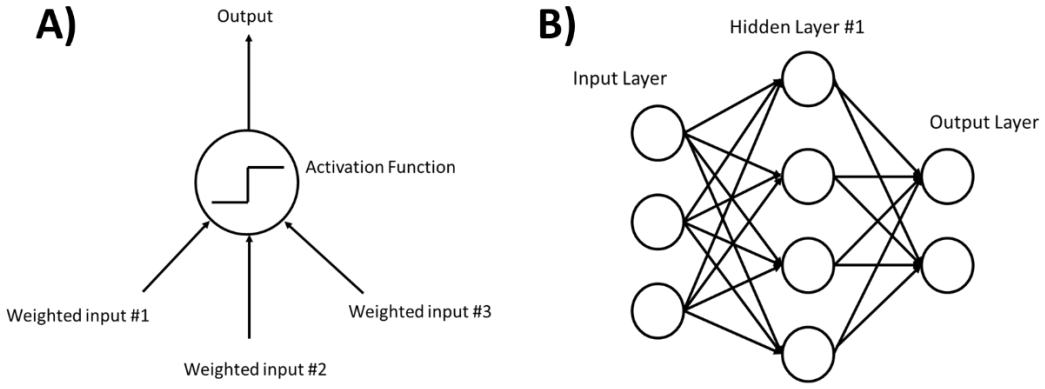

Figure 4.5. MLP classifier schematics. A) Schematic of a single perceptron. B) Schematic of a neural network formed by multiple layers of interconnected perceptrons

4.2.5.5 Stacking Classifier

Stacking Classifiers are another class of ensemble ML algorithms that operate similarly to Random Forest Classifiers where the final output feature prediction from the model is determined from the votes from each of the underlying base classifier models. However, there two significant differences between these classifiers that are worth noting. The first is that the base models for the Stacking Classifier consist of different ML Classifier algorithms, rather than multiple Decision Tree Classifier algorithms. The second is that the stacking classifier utilizes a secondary ML algorithm to evaluate the output feature predictions from each of the base classifiers and determine the final output feature prediction probability. For the purposes of this study, a Logistic Regression Classifier, Random Forest Classifier, Support Vector Machine Classifier, K-Nearest Neighbors Classifier, a Naïve Bayes Classifier, an Extreme Gradient Boosting Classifier, an ANN Classifier, and a LightGBM Classifier were implemented as base ML algorithms and a second Logistic Regression Classifier was implemented as the secondary ML algorithm for the Stacking Classifier. The base ML algorithms and The Stacking Classifier was implemented using the *StackingClassifier* class in the *scikit-learn.ensemble* library (scikit-learn v. 1:3:0).

4.2.6 Hyperparameter Tuning

Each ML model employed in this study has underlying mathematical parameters that influence how the model operates on the dataset. These parameters are called hyperparameters because they have to be chosen manually by the model developer rather than being automatically selected during the model training. Just as with feature selection, selecting the correct hyperparameters is a crucial step in model development as the right hyperparameters will enable the model to best capture the underlying patterns in the data while avoiding overfitting or underfitting. While knowing exactly what values to select for different model hyperparameters can be a daunting task, a grid search strategy can be employed to easily enable a wide range of model hyperparameters to be tested and evaluated to determine the correct values for a particular dataset.

In this study, a Bayesian Optimization grid search strategy was employed to evaluate a range of hyperparameters for each model (Table 4.2). In a normal grid search strategy, each hyperparameter combination defined by the developer is tested systematically one at a time. However, Bayesian Optimization utilizes a variant of Bayes' theorem to identify the best hyperparameter options in a set of hyperparameters without having to evaluate each combination, making it a much more efficient search method. The modified version of Bayes' theorem can be expressed as:

$$p(Score|Hyperparameters) = \frac{p(Hyperparameters|Score)_{*p(Score)}}{p(Hyperparameters)} \quad (4.11)$$

By implementing this modified version of Bayes' theorem, hyperparameters in a set are maps to a corresponding score probability to create a probabilistic model that enables the grid

search to converge to the optimal hyperparameter values, rather than blindly testing each combination of values individually. A more detailed discussion of this algorithm is presented by Wu *et al.,* (82). Bayesian Hyperparameter Tuning was implanted using the *BayesSearchCV* class from the *skopt* library (scikit-optimize v. 0:8:1).

Table 4.2. List of ML classifiers and the associated sets of evaluated hyperparameters

Machine Learning Model	Hyperparameters
LR	"C": [0.1, 1, 10], "penalty": ["l1", "l2"], "solver": ['liblinear', 'saga'], "class_weight":[None,"balanced"]
RF	"n_estimators":[100, 200, 300], "max_depth":[None, 5,10, 15], "min_samples_split": [2, 5, 10], "min_samples_leaf": [1,2,4], "max_features":['auto','sqrt','log2'], "criterion":['gini','entropy'], "class_weight":[None,'balanced']
XGB	"n_estimators":[100, 200, 300], "learning_rate":[0.1, 0.05, 0.01], "max_depth": [3, 5, 10], "subsample": [0.5,1,'uniform'], "gamma":[0, 5.0], "colsample_bytree":[0.8, 0.9, 1.0]
MLP	"activation":['relu','tanh'], "solver":['sgd','adam'], "learning_rate":['constant','adaptive'], "alpha":[0.0001, 0.001, 0.01]

4.2.7 Decision Threshold Tuning

Once the model had been fit to the training data and the model hyperparameters had been tuned on the validation data to optimize the model performance, the final step in constructing the model was to tune the decision threshold value for classifying patients as “at risk” for MHT. For each model, the predicted MHT probability for each patient was calculated. Then a threshold

value between 0 and 1 was then chosen and each probability that was above that threshold value was assigned an "at risk" for MHT designation and each probability below that threshold was considered "normotensive". The precision and recall of the model using this threshold was then calculated. From that value, the F1 score was determined. Precision is a quantitative measure of the ability of the ML model to correctly make a positive prediction and can be defined mathematically as:

$$Precision = \frac{True\ Positive\ Predictions}{(True\ Positive\ Predicitons + False\ Positive\ Predictions)}, \tag{4.12}$$

Similarly, recall is a quantitative measure of the ML model's ability to correctly identify a prediction belonging to the class of interest, in this case, MHT. This can be expressed mathematically as:

$$Recall\ = \frac{True\ Positive\ Predictions}{(True\ Positive\ Predicitons + False\ Negative\ Predictions)}, \tag{4.13}$$

The F1 score, defined as the harmonic mean of the precision and recall, was selected as the evaluation metric of choice due to the imbalanced nature of the dataset. This metric can be readily calculated from the precision and recall using the following equation:

$$F1\ Score\ = 2 * \frac{Precision * Recall}{(Pecision + Recall)}, \tag{4.14}$$

The F1 scores for threshold values ranging from 0 to 1 in increments of 0.01 were calculated and the threshold that had the highest F1 was deemed the optimal decision threshold. To ensure that the optimal threshold would generalize well to the test dataset, the training dataset was cross validated using k-fold cross validation. Briefly, the dataset was split into equally partitioned subsets based on the number of splits defined by the user. For this study, 7 was chosen as the number of splits given that previous studies have found this value to be the recommended number of splits for optimizing model performance across multiple different types of ML algorithms (62). The probabilities of MHT for each dataset were then calculated and the F1 score values for each threshold for each subset was determined. The F1 score values for each threshold value were averaged across the 7 subsets and the optimal threshold was chosen based on the threshold that corresponded to the highest average F1 score.

4.2.8 Evaluation Metrics

This study evaluated several classification metrics commonly used to evaluate the performance of a ML model on an imbalanced dataset. These metrics include the area under the receiving operator curve (ROC AUC), area under the precision recall curve (PR AUC), the confusion matrix, the aforementioned F1 score, sensitivity, and specificity. Each of these metrics provides information on how well the model was able to identify patient outcomes based on the features in the dataset. ROC AUC is one of the most common evaluation metrics for binary classification problems and is constructed by plotting the true positive rate (TPR) vs the false positive rate (FPR) for a range of decision thresholds between 0 and 1 and then calculating the area under the resulting curve. The TPR and FPR are defined, respectively as:

$$TPR = \frac{True\ Positive\ Predictions}{True\ Positives\ Predictions - False\ Negatives\ Predictions}, \quad (4.15)$$

and

$$FPR = \frac{False\ Positive\ Predictions}{False\ Positive\ Predictions - Tru\ Negative\ Predictions}, \quad (4.16)$$

A confusion matrix is a 2 x 2 matrix that compares the positive and negative predictions of the ML model to the positive and negative values of the dataset that the model is being evaluated against. This metric helps to visually access how well the predictions from the model are aligning with the actual dataset values.

The F1 score was reported as the F1 score calculated using equation 8 at the optimal decision threshold, as explained in 4.*2.7 Decision Threshold Tunning*.

The sensitivity and specificity of the model are the reported TPR and FPR values for the optimal decision threshold. The sensitivity provides a quantitative metric for assessing how well the model can correctly identify positive results. Similarly, specificity is a quantitative metric for assessing how well the model can correctly identify negative results. In the context of this study, sensitivity was used to assess how well the model could identify a patient as having MHT and specificity was used to assess how well the model could identify a patient as being normotensive. Furthermore, the greater the sensitivity, the less likely an individual with a negative predicted value will actually have MHT while, conversely, the greater the specificity, the less likely an individual with a positivity will actually be normotensive.

4.2.9 Model Interpretation

While the predictions of several standard ML algorithms such as Logistic Regression Classifiers and Decision Tree Classifiers can be readily interpreted to determine how different features in a dataset generate lead to predicted output feature from a model for a given instance, it can be difficult to interpret how the predictions are generated from more complex models such as ensemble models like the Random Forest Classifier or the Stacking Classifier. Model interpretability is a key factor when developing a ML model for clinical applications as end users such as clinicians and other medical professionals need to be able to understand how a prediction was made in order to confirm that the prediction is valid and aligns with their own medical knowledge and understanding. Thus, a method for helping model end users to make sense of the model is imperative when the final model involves more complex ML algorithms (66). Two common methods used in ML model development include SHAP Values and Partial Dependence Plots (PDPs). SHAP Values enable the end user to understand the relative importance of each feature in the model and can be used to assess how much a feature contributes to an outcome for a given instance. PDPs provide insight into the relationship between a predicted outcome and a specific feature.

4.2.9.1 SHAP Values

SHAP values were used to access the contribution of a feature to a model's predicted outcome for a particular instance is a concept rooted in game theory (69). In its original context, Shapley values were created as a means of evaluating a player's contribution to a game's final outcome. It has since been applied in the field of ML as a model-agnostic means of interpreting

how a model operates (70). In this case, each feature in the model is considered a player in the game and the model prediction is the final outcome.

4.2.9.2 Partial Dependence Plots

PDPs are a valuable tool for interpreting and explaining the behavior of ML models by providing insights into how the predicted outcome of the model changes as a function of a specific feature while holding other features constant. By isolating the effect of a single feature on the model's predictions, PDPs can highlight the relationship between that feature and the output feature in an intuitive and visual manner. These plots can then be used to identifying trends, patterns, and potential nonlinearities that might not be evident from simple summary statistics or coefficients. PDPs can also facilitate the detection of interaction effects between variables, showcasing how their combined influence impacts the model's output.

4.3 Results

4.3.1 Baseline Characteristics of Study Population

The demographic, clinical, biochemical, and family history characteristics of the study participants are summarized in Table 4.3. A total of 1,033 participants were included in the preprocessed dataset, of which 178 (17%) had a confirmed diagnosis of MHT. Compared to normotensive participants, participants with a confirmed diagnosis of MHT were more likely to be male, white, and have a father with hypertension. They also had significantly higher body mass indexes (BMI), waist-to-hip ratios (WHR), systolic blood pressure (SBP), diastolic blood pressure (DBP), mean arterial pressure (MAP), pulse pressure (PP), triglycerides (TG), glucose

(GLU), gamma-glutamyl transferase (GGT). Participants with MHT also had significantly lower levels of high-density lipoprotein cholesterol (HDL-C) then normotensive participants.

Table 4.3. Comparison of characteristics of normotensive and MHT study participants.

	Normotensive (n = 866)	Masked Hypertensive (n = 178)	P-value
Demographic data			
Age, years	24.41 ± (3.08)	24.29 ± (3.11)	0.686
Male sex, *n (%)*	339 (39.15%)	129 (72.47%)	<0.001
White ethnicity, *n (%)*	421 (48.61%)	116 (65.17%)	<0.001
BMI, kg/m2	24.28 ± (4.92)	28.03 ± (6.67)	<0.001
WHR, *%*	77.27 ± (7.26)	82.41 ± (8.26)	<0.001
Office BP parameters			
Office SBP, mmHg	113.71 ± (8.27)	123.73 ± (7.05)	<0.001
Office DBP, mmHg	74.65 ± (5.87)	78.35 ± (5.83)	<0.001
Office MAP, mmHg	90.98 ± (6.88)	97.76 ± (5.64)	<0.001
Office PP, mmHg	87.44 ± (16.95)	102.14 ± (17.00)	<0.001
Biochemical profiles			
TC, mmol/L	3.74 ± (1.20)	3.85 ± (1.21)	0.249
TG, mmol/L	0.79 ± (0.51)	1.05 ± (0.84)	<0.001
HDL-C, mmol/L	1.19 ± (0.41)	1.03 ± (0.40)	<0.001
LDL-C, mmol/L	2.43 ± (0.98)	2.57 ± (1.04)	0.171
Glucose, mmol/L	4.06 ± (1.03)	4.23 ± (1.18)	0.031
Cotinine, ng/mL	52.01 ± (114.32)	52.83 ± (103.79)	0.255
GGT, U/L	21.12 ± (17.73)	28.45 ± (28.50)	<0.001
Family History			
Mother Hypertension, *n (%)*	230 (26.56%)	52 (29.21%)	0.563
Father Hypertension, *n (%)*	184 (21.25%)	56 (31.46%)	0.006
Mother Heart Disease, *n (%)*	159 (18.36%)	27 (15.17%)	0.394
Father Heart Disease, *n (%)*	271 (31.29%)	66 (37.08%)	0.887
Mother Stroke, *n (%)*	84 (9.70%)	24 (13.48%)	0.971
Father Stroke, *n (%)*	136 (15.70%)	43 (24.16%)	0.458
Mother Diabetes, *n (%)*	240 (27.71%)	42 (23.60%)	1
Father Diabetes, *n (%)*	395 (45.61%)	65 (36.52%)	0.901

BMI, Body Mass Index; WHR, Waist Hip Ratio; SBP, Systolic Blood Pressure; DBP, Diastolic Blood Pressure; MAP, Mean Arterial Blood Pressure; PP, Pulse Pressure; TC, Total Cholesterol; TG, Triglycerides; HDL-C, High-Density Lipoprotein-Cholesterol; LDL-C, Low-Density Lipoprotein; GGT, gamma-glutamyl transferase.

The study cohort was split into training and testing cohorts for the purposes of developing and validating the ML model using a 4:1 split ratio. This resulting in a total of 826 participants in the training cohort and 207 participants in the testing cohort. The proportion of patients with MHT to normotensive patients was conserved in both groups. Comparisons of the demographic, clinical, biochemical, and physical activity characteristics between the training and testing

cohorts is summarized in Table 4.4. Characteristics between the two cohorts were not significantly different.

Table 4.4. Comparison of characteristics of training and testing cohort.

	Training Set (n = 835)	Testing Set (n = 209)	P-value
Demographic data			
Age, years	24.40 ± 3.10	24.38 ± 2.99	0.94
Male sex, *n (%)*	369 (44.19%)	99 (47.37%)	0.454
White ethnicity, *n (%)*	428 (51.23%)	109 (52.15%)	0.877
BMI, kg/m2	24.91 ± 5.31	24.95 ± 5.96	0.749
WHR, *%*	78.06 ± 7.55	78.51 ± 8.20	0.685
Office BP parameters			
Office SBP, mmHg	115.22 ± 8.85	115.88 ± 8.86	0.305
Office DBP, mmHg	75.15 ± 6.03	75.80 ± 5.98	0.158
Office MAP, mmHg	91.96 ± 7.10	92.82 ± 7.38	0.131
Office PP, mmHg	89.92 ± 17.85	90.08 ± 17.76	0.768
Biochemical profiles			
TC, mmol/L	3.77 ± 1.24	3.73 ± 1.08	0.959
TG, mmol/L	0.82 ± 0.51	0.89 ± 0.83	0.555
HDL-C, mmol/L	1.17 ± 0.43	1.13 ± 0.36	0.469
LDL-C, mmol/L	2.45 ± 1.02	2.44 ± 0.86	0.638
Glucose, mmol/L	4.09 ± 1.06	4.11 ± 1.08	0.977
Cotinine, ng/mL	50.46 ± 109.68	58.76 ± 123.06	0.518
GGT, U/L	22.37 ± 20.03	22.43 ± 20.77	0.876
Family History			
Mother Hypertension, *n (%)*	225 (26.95%)	57 (27.27%)	1
Father Hypertension, *n (%)*	189 (22.63%)	51 (24.40%)	0.887
Mother Heart Disease, *n (%)*	137 (16.41%)	49 (23.45%)	0.117
Father Heart Disease, *n (%)*	270 (32.34%)	67 (32.06%)	0.875
Mother Stroke, *n (%)*	88 (10.54%)	20 (9.57%)	1
Father Stroke, *n (%)*	152 (18.20%)	27 (12.92%)	0.12
Mother Diabetes, *n (%)*	226 (27.07%)	56 (26.79%)	0.714
Father Diabetes, *n (%)*	361 (43.23%)	99 (47.37%)	0.133

BMI, Body Mass Index; WHR, Waist Hip Ratio; SBP, Systolic Blood Pressure; DBP, Diastolic Blood Pressure; MAP, Mean Arterial Blood Pressure; PP, Pulse Pressure; TC, Total Cholesterol; TG, Triglycerides; HDL-C, High-Density Lipoprotein-Cholesterol; LDL-C, Low-Density Lipoprotein; GGT, gamma-glutamyl transferase.

4.3.2 Feature Selection and Relative Importance of Selected Features

Five different feature selection strategies were evaluated. These strategies included: 1) manually selecting all the available features in the dataset, 2) manually selecting features based on relevant predictors reported in the scientific literature, 3) automatically selecting features using the RFE algorithm wrapped around a SVM classifier, 4) automatically selecting features

using the Boruta SHAP algorithm wrapped around a RF classifier, and 5) automatically selected features using LASSO regression. The selected features for each feature selection strategy are reported in Table 4.5.

Table 4.5. Selected features for each feature selection strategy.

Feature Selection Strategy	Number of Features	Selected Features
No Feature Selection	192	See Appendix C. Complete List of Transformed African PREDICT Categorical Features
Manual Feature Selection	27	sex, ethnicity, age, bmi, whr, sbp, dbp, map, pp, hr, glu, trig, ldl, hdl, chol, cotinine, ggt, smoker, excessive alcohol, sedentary, prehypertensive, obese, stroke, diabetes, lvm_cube, left ventricular hypertrophy, intima-media thickness
RFE-SVM Feature Selection	18	bw, sbp, minutes_vig_exerc, map, lvm_cube, gm_csf, il_1beta, il_12, ang_1_10_pmol, ang_1_5_pmol, ang_iv_pmol, ras_ace, s_meds_d, s_meds_h, s_meds_c, smoke_type___4, heart_attack, stroke
Boruta SHAP Feature Selection	13	bw, bsa, wc, nc, sbp, dbp, map, pp, ad_max, lvm_cube, lvd_mass, lvs_mass, prehypertensive
LASSO Feature Selection	35	age, bw, sbp, minutes_vig_exerc, map, pp, imtn_mean, imtf_min, lvpwd, lvm_cube, lvs_mass, lad, mv_A_vel, ang_1_5_pmol, ras_renin, sex, ethnicity, smoke_type___1, smoke_type___6 , type_alcohol___1, type_alcohol___2, type_alcohol___5, type_alcohol___6, type_alcohol___7, strenuous exercise, moderate_exercise, mild_exercise, leisure-activity_times, mother_heart_disease, mother_stroke, father_stroke, father_diabeteres, excessive_alcohol, dyslipidemic

Body mass index (bmi), waist_hip_ratio (whr), systolic blood pressure (sbp), diastolic blood pressure (dbp), mean arterial pressure (map), pulse pressure (pp), heart rate (hr), glucose (glu), triglycerides (trig), low density lipoprotein(ldl), high density lipoprotein (hdl), total cholesterol (chol) gamma-glutamyl transferase (ggt), left ventricular mass (lvm_cube), body weight (bw), granulocyte-marcrophage colony-stimulating factor (gm_csf), body surface area (bsa), waist circumference (wc), neck circumference (nc), left ventrical diastolic mass (lvd_mass), left ventrical systolic mass (lvs_mass), adventitial diameter maximum value (ad_max), intima-media mean near wall thickness (imtn_mean), intima-media minimum far wall thickness (imtf_min) left ventricular posterior wall thickness at diastole (lvpwd), left atrium diameter (lad), mitrial velocity A wave (mv_A_vel).

There were several features that were identified as significant regardless of the feature selection strategy employed. These features included systolic blood pressure, mean arterial pressure, body weight, and left ventricular mass. Minutes of vigorous exercise, pulse pressure, and left ventricular systolic mass, were features that were commonly selected by at least 2 of the 3 automatic feature selection methods. Compared to the manually selected features, the RFE-SVM feature selection strategy selected 12 novel features: granulocyte-macrophage colony-stimulating factor, il_1beta, il_12, ang_1_10_pmol, ang_1_5_pmol, ang_iv_pmol, ras_ace, s_meds_d, s_meds_h, s_meds_c, smoke_type___4, and heart attack. The BorutaSHAP feature selection strategy identified both left ventricular diastolic mass and left ventricular systolic mass as import features compared to the manually feature selection strategy. The LASSO feature

selection strategy identified intima-media mean, intima-media minimum, left ventricular posterior wall thickness at diastole, left atrium diameter, and mitrial velocity A wave imtf_min, lvpwd, lvs_mass, lad, mv_a_vel, ang_1_5_pmol, ras_renin, smoke_type___1, smoke_type___6, type_alcohol___1, type_alcohol___2, type_alcohol___5, type_alcohol___6, type_alcohol___7, strenuous_exercise, moderate_exercise, mild_exercise, leisure_activitiy_times, mother_heart_disease, mother_stroke, father_stroke, father_diabetes, excessive_alcohol, and dyslipidemic as important.

4.3.3 Comparison of Base Classifiers and Stacking Classifiers Across Different Feature Selection Strategies

Each permutation of feature selection strategy and machine learning algorithm was implemented to create a total of 25 different MHT classifier models. 7-fold cross validation was used to obtain the mean and standard deviation of the ROC AUC for each model (Table 4.6). The BorutaSHAP feature selection strategy combined with a Stacking Classifier constructed using LR, RF, XGB, and MLP classifiers tuned using the features selected using BorutaSHAP obtained the highest average ROC AUC score out of the entire set of models when evaluated on the test set, with a ROC AUC of 0.86 ± 0.09. Conversely, the MLP classifier tuned using the features selected using LASSO obtained the lowest average ROC AUC score out of the entire set of models when evaluated on the test set, with a ROC AUC of 0.70 ± 0.16.

When no feature selection was employed, the XGB classifier performed the best while the LR classifier performed the worst (Figure 4.6). When manual feature selection was employed, the LR and STK classifiers performed the best and the MLP classifier performed the worst. Similarly, for RFE-SVM, the LR, RF, and STK classifiers performed the best and the MLP classifier again exhibited the worst ROC AUC score. For BorutaSHAP, the STK classifier

achieved the highest score, while the XGB classifier recorded the lowest score. Finally, for the LASSO feature selection method, the RF classifier had the highest performance and the MLP classifier had the lowest performance.

Table 4.6. Comparison of machine learning pipelines.

Machine learning pipeline	Training AUC	Test AUC
No Feature Selection + LR	0.82 ± (0.07)	0.72 ± (0.11)
No Feature Selection + RF	0.83 ± (0.07)	0.79 ± (0.15)
No Feature Selection + XGB	0.83 ± (0.07)	0.83 ± (0.12)
No Feature Selection + MLP	0.80 ± (0.07)	0.76 ± (0.18)
No Feature Selection + STK	0.83 ± (0.06)	0.80 ± (0.14)
Manual Selection + LR	0.83 ± (0.06)	0.85 ± (0.11)
Manual Selection + RF	0.82 ± (0.08)	0.84 ± (0.11)
Manual Selection + XGB	0.82 ± (0.07)	0.83 ± (0.13)
Manual Selection + MLP	0.82 ± (0.07)	0.78 ± (0.18)
Manual Selection + STK	0.83 ± (0.07)	0.85 ± (0.13)
RFE-SVM Selection + LR	0.85 ± (0.06)	0.85 ± (0.11)
RFE-SVM Selection + RF	0.85 ± (0.08)	0.85 ± (0.08)
RFE-SVM Selection + XGB	0.84 ± (0.08)	0.84 ± (0.11)
RFE-SVM Selection + MLP	0.85 ± (0.06)	0.82 ± (0.13)
RFE-SVM Selection + STK	0.86 ± (0.06)	0.85 ± (0.10)
BorutaSHAP Selection + LR	0.84 ± (0.06)	0.85 ± (0.10)
BorutaSHAP Selection + RF	0.83 ± (0.06)	0.84 ± (0.10)
BorutaSHAP Selection + XGB	0.83 ± (0.06)	0.83 ± (0.11)
BorutaSHAP Selection + MLP	0.83 ± (0.06)	0.85 ± (0.11)
BorutaSHAP Selection + STK	0.83 ± (0.06)	0.86 ± (0.09)
LASSO Selection + LR	0.85 ± (0.07)	0.78 ± (0.18)
LASSO Selection + RF	0.85 ± (0.07)	0.85 ± (0.08)
LASSO Selection + XGB	0.84 ± (0.07)	0.82 ± (0.11)
LASSO Selection + MLP	0.80 ± (0.07)	0.70 ± (0.16)
LASSO Selection + STK	0.85 ± (0.07)	0.82 ± (0.13)

AUC, area under the receiver operating characteristic curve; LR, logistic regression; RF, random forest; XGB, Extreme Gradient Boosting; MLP, multilayer preceptron; STK, stacking; RFE, recursive feature elimination; LASSO, least absolute shrinkage and selection operator

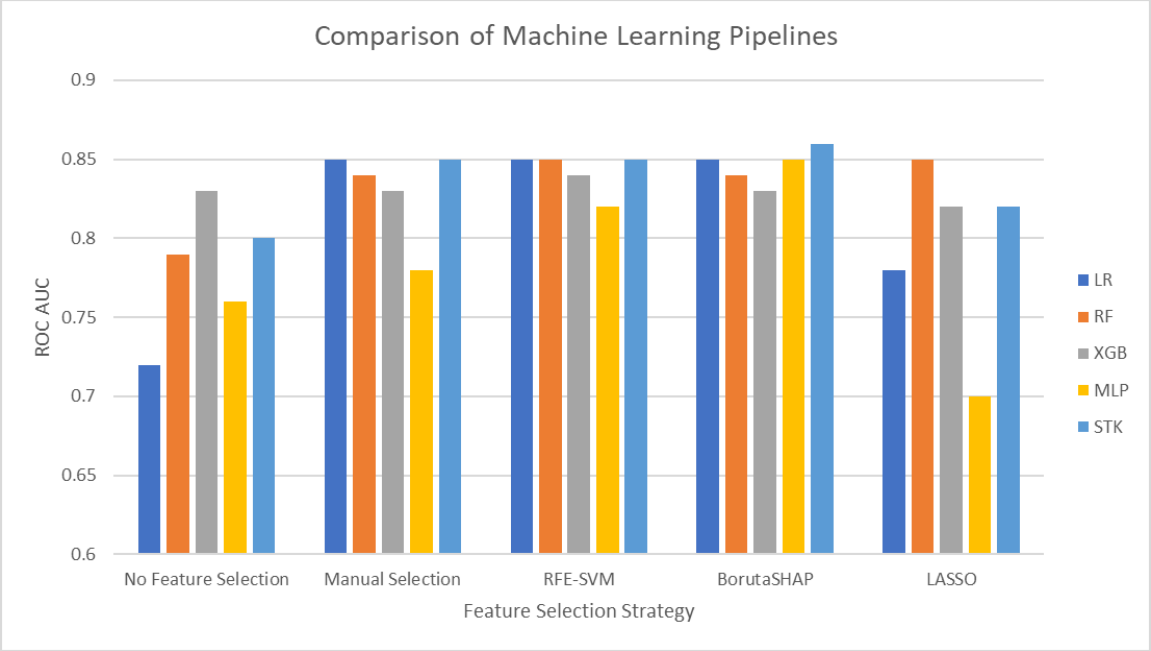

Figure 4.6. Comparison of machine learning pipelines. Machine learning pipeline performances are grouped by feature selection strategy.

4.3.4 Evaluation of Final Model Performance

The Stacking classifier with BorutaSHAP feature selection strategy was chosen as the final model for the MHT classifier due to its superior ROC AUC score. Decision threshold tuning was performed to maximize the model's F1 score on the training set. A threshold value of 0.186 was found to optimize model performance. To assess the overall model performance, the model's ROC AUC, PR AUC, F1 score, sensitivity, and specificity on the test set were calculated at this decision threshold and compared to the corresponding metrics calculated by a simple binary model that classifies patients as having MHT if they have an office blood pressure greater than 120 mmHg. This simple binary classifier was based on a suggestion proposed by Thompson et al., as a simple means for assessing a patient's risk for MHT (67). The results of the comparison are outlined in Table 4.7 and Figure 4.7. The Stacking classifier outperformed the binary classifier across all metrics evaluated; however, these differences were not statistically significant.

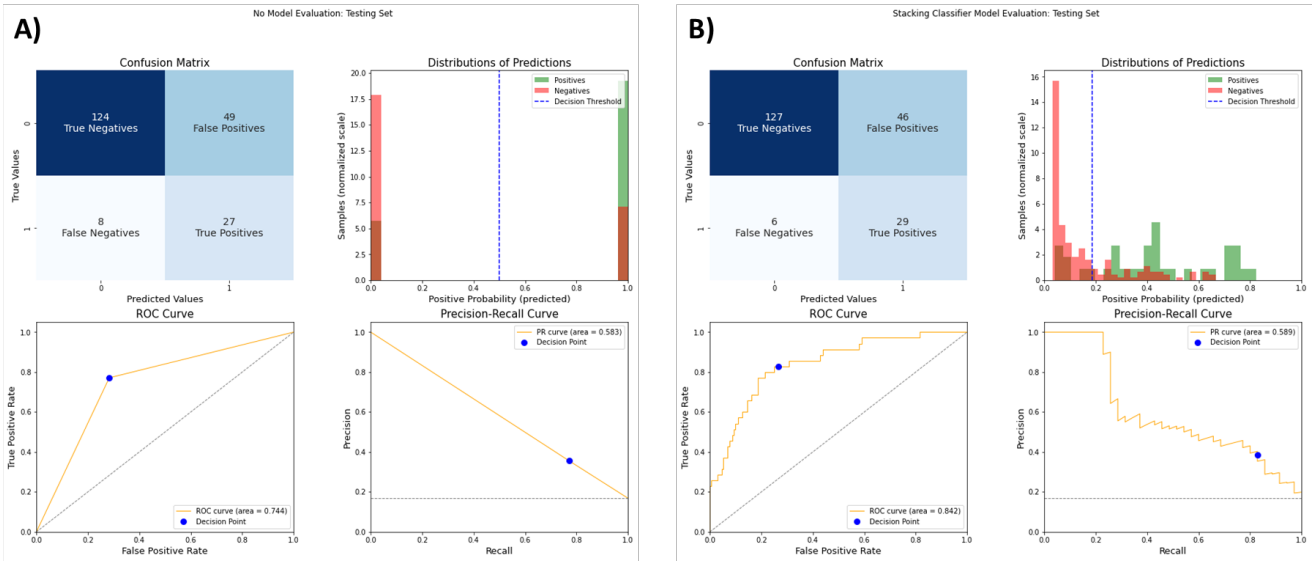

Figure 4.7. Comparison of MHT ML model performance vs. MHT binary model performance. A) The confusion matrix, distribution of predictions, ROC AUC curve, and PR AUC curve for the simple binary classifier. **B)** The confusion matrix, distribution of predictions, ROC AUC curve, and PR AUC curve for the ML model.

Table 4.7. ML model evaluation metrics vs. simple binary classifier evaluation metrics.

		ACTUAL MHT						
MODEL PREDICTION		0	1	ROC AUC	PR AUC	F1 Score	Sensitivity	Specificity
Binary Model	0	124	49	0.74	0.58	0.486	0.77	0.36
	1	8	26					
ML Model	0	127	46	0.83	0.59	0.527	0.83	0.39
	1	6	29					

4.3.5 Model Explanation

SHAP analysis was used to evaluate the overall contribute of each feature to the model's predictions. SHAP values were calculated for each feature for each instance in the optimized dataset. The absolute value of each SHAP value for a particular feature across all instances were summed together and the total value was used to indicate the importance of the feature relative to the other features in the optimized dataset (Figure 4.8).

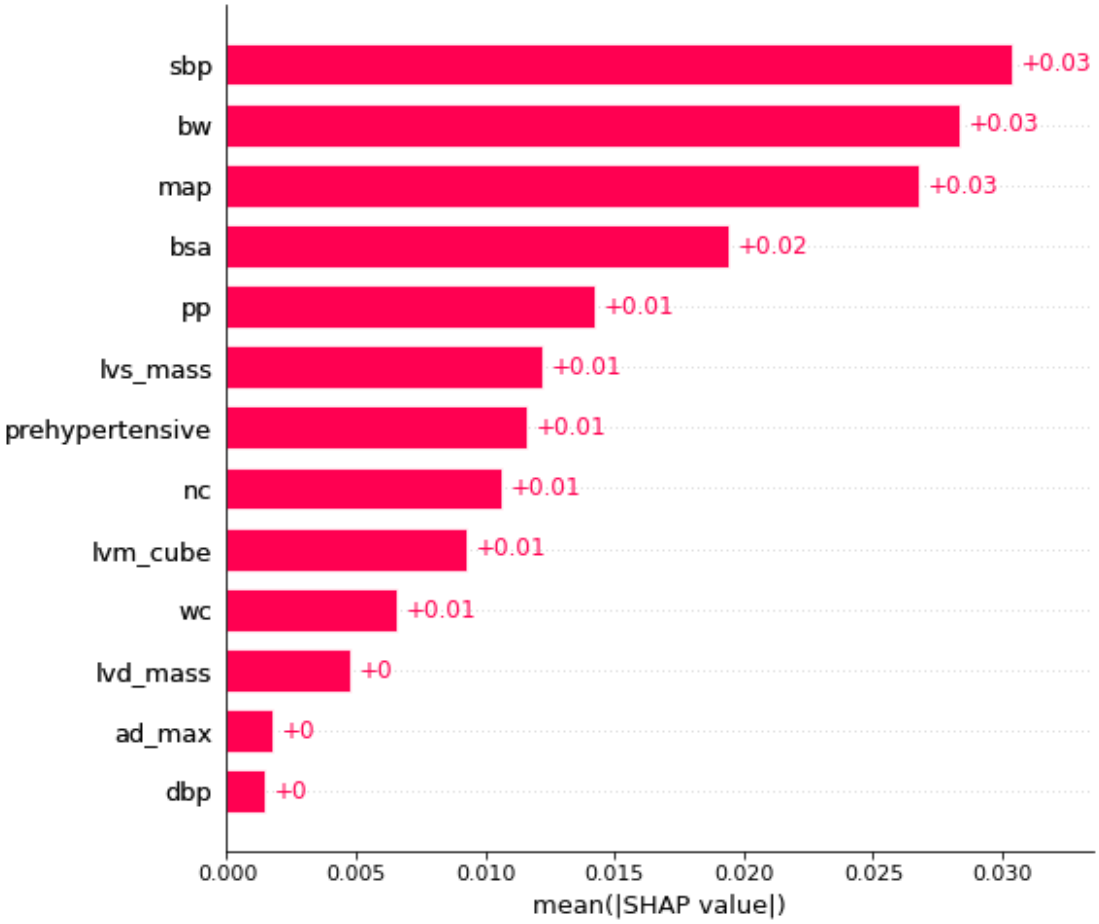

Figure 4.8. Features ranking according to mean absolute SHAP value. The absolute SHAP value of each feature for each instance was calculated and average to get the relative feature importance.

The relationship between each feature and the predicted outcome of the model were also assessed using PDPs (Figure 4.9). Figure 4.9a-e, g, h, j, and l indicate that the likelihood of the model classifying a patient with MHT increases nearly linearly as bw, bsa, wc, nc, sbp, pp, map, lvm_cube, and lvs_mass increases. Figure 4.9f indicates that dbp does not have a significant impact on the likelihood of a patient having MHT until the dbp measurement exceeds approximately 80 mmHg and then it begins to have a significant impact on the model prediction. Figure 4.9i indicates that the likelihood of the model classifying a patient as MHT decreases as

ad_max increases. Figure 4.9k incidates a gradual linear increase in likelihood of classifying a patient as MHT, but then the likelihood begins to increase significantly as lvd_mass exceeds approximately 210 g. Finally, Figure 4.9m indicates that the likelihood of having MHT increases if the patient is also considered prehypertensive.

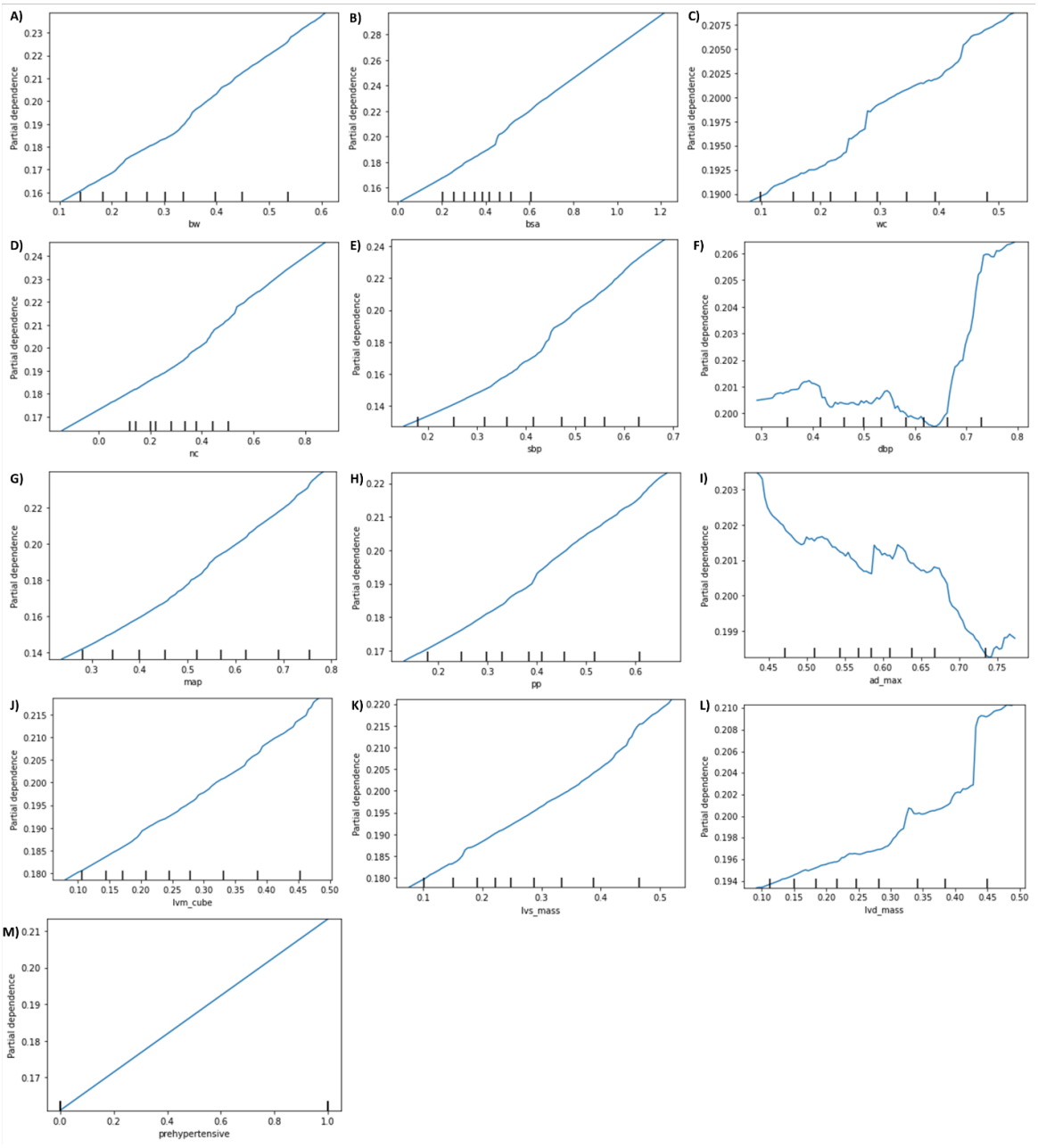

Figure 4.9. Partial Dependence Plots for features selected using the BorutaSHAP feature selection strategy. The features include: **A)** bw, **B)** bsa, **C)** wc, **D)** nc, **E)** sbp, **F)** dbp, **G)** map, **H)** pp, **I)** ad_max, **J)** lvm_cube, **K)** lvd_mass, **L)** lvs_mass, and **M)** prehypertensive.

4.4 Discussion

The aim of this study was to develop a machine learning model that would be able to detect MHT in a seemingly young and healthy individual from a LMIC. This would allow healthcare providers to conserve resources by identifying which patients are most at risk for MHT and should be evaluated further for MHT using ABPM. Data from the African-PREDICT study met the criteria for the aim of this study and served as the training set for developing the model. To develop the model, several feature popular feature selection methods and ML algorithms were evaluated. The results of this study found that the BorutaSHAP feature selection strategy in conjunction with a Stacking classifier resulted in a ML model that had the highest ROC AUC score out of the set of 25 different models that were constructed. The final ML model was compared to a simple binary classifier that serves as the current recommended "rule-of-thumb" for evaluating a patient's risk for MHT in LMIC and was found to perform better in each evaluation metric that was considered (67). SHAP analysis and PDP plots were also implemented to provide the model with interpretability. Overall, the results of this study show that a ML model could be utilized to identify patients most at risk for MHT and highlights which features out of the ones that were considers have the greatest impact on a patient's risk for MHT. It is worth noting that all of the features that were considered important for making predictions are easily obtained from a single outpatient visit, which enables the model to be implemented easily into a routine clinic evaluation. Interestingly, this study also demonstrated that the current recommended "rule-of-thumb" of classifying patients with greater than clinical systolic blood pressure measurement of greater than 120 mmHg is surprisingly effective at identifying patients at risk for MHT and could be used in instances where employing a ML model might not be feasible. Finally, the SHAP analysis revealed that the most important predictors of MHT for this

ML model were related to blood pressure and body weight, 2 factors which can be regulated through clinical intervention and lifestyle changes. Individual patient analysis using this model could be used to identify which of these features need to be focused on and could help clinicians in implementing a clinical action plan.

A review of the relevant literature reveals that several recent studies have developed ML models to predict MHT (68 - 72). Two of these studies used datasets focused on patients between 20 and 50 years of age from Taiwan, a high-income country (68, 70) and developed models based on single classifiers such as Logistic Regression or Multilayer Perceptron classifiers or homogeneous ensembles such as XGBoost and Random Forest classifiers. Another study focused on developing a ML training on a dataset consisting of medical information from patients younger than 16 years of age from the United States and Canada (69). Other studies focused on older patients from Wuhan, China and Glasgow, Scotland (71, 72). While these studies served as valuable foundational work for this study, to our knowledge, the work presented here is the first to focus specifically on identifying MHT in young, healthy adults from LMICs and is the first to employ a stacking ensemble framework to develop a ML model for detecting MHT. Here, it was demonstrated that the proposed stacking classifier consisting of several different base classifier models could, on average, outperform each of the base classifiers. This finding is consistent with other studies which developed heterogeneous ensemble models for binary disease classification (73, 74).

Feature selection is widely regarded as an important step in creating a ML model pipeline and the results of this study confirm this concept. Of all the feature selection strategies that were evaluated, not implementing any feature selection method, and utilizing all the available features in the dataset resulted, on average, in the lowest performing group of ML models. ROC AUC

scores increased in all other feature selection methods for all other ML algorithms, with the one exception of the MLP classifier trained on the features selected using the LASSO feature selection strategy. It is interesting to note that the ML models trained on a set of manually selected features performed, on average, better than the ML models trained on the set of features selected using LASSO, a feature selection method used commonly in many ML applications. This demonstrates that domain knowledge is necessary to ensure that the features used in a ML problem are actually the most relevant features to include. The two other automatic feature selection methods that were implemented, RFE-SVM and BorutaSHAP, performed on average better than the manually selected feature set. In the case of BorutaSHAP, nearly all of the selected features were also found in the manually selected feature set, suggesting that, for this particular dataset, some of the manually selected features such as age, stroke, or diabetes mellitus might not be pertinent to the population that was being evaluated as the inclusion criteria of the participants in this study were selected on the basis of being within a certain age range and health status where these factors might not have much of a correlation with MHT. For this reason, it is possible that the manually selected feature set could have performed similarly to the BorutaSHAP feature set if these features had been ignored. The RFE-SVM produced, on average, the second highest performing group of ML models. It is worth noting that this method of feature selection alone identified several biochemical features as important predictors of MHT.

The final model's performance was evaluated on the test set, resulting in an average ROC AUC score of 0.843, an average PR AUC score of 0.589, a precision score of 0.387, a recall score of 0.829, and an F1 score of 0.527. These evaluation metrics are comparable to the reported evaluation metrics for other recently published ML models for MHT (68, 70, 72) or

better (69,71). It is worth noting that the model developed on this study exhibited a high degree of recall, which is a desirable attribute for a ML model, as it indicates that the model will rarely diagnosis a patient as normotensive when they actually have MHT. The model in this study errors on the side of precision, resulting in patients being classified as "at-risk" for MHT when they are truly normotensive. In the case of MHT, this would be an acceptable outcome, as physicians could either use an ABPM to verify the model prediction or recommend lifestyle changes to the patient that could be potentially beneficial, regardless of a true diagnosis of MHT. It is also worth noting that, in a previous study that evaluated the African-PREDICT dataset, the authors mentioned that a systolic blood pressure measurement of 120 mmHg could be used as a cut-off value for classifying someone as "at-risk" for MHT. In this study, a simple binary model was created that classified patients as having MHT if they had a systolic blood pressure of 120 mmHg. With just this single predictor alone, this basic classification model was able to perform surprisingly well on the test set, resulting in a ROC AUC score of 0.744, a PR AUC score of 0.583, precision and recall scores of 0.355 and 0.771 respectively, and an F1 score of 0.486. This could suggest that either there is a high degree of overlap between systolic blood pressure and other predictors in the ML model, or that systolic blood pressure is the primary predictor of MHT. The ML model proposed here did perform better than this simple binary classifier; however, the performance was comparable enough to suggest that the simple binary classifier could be used in place of the proposed ML model if utilizing the ML model was not a feasible option.

As mentioned previously, model interpretability is crucial for promoting clinical use because it provides the clinician with insight into how the model works. Here SHAP analysis and PDPs were used to gain insight into how the model generally works. Using SHAP analysis, it

was shown that a patient's clinical systolic and diastolic blood pressures, mean arterial pressure, pulse pressure, body weight, and left ventricular mass are all important factors that are highly correlated with MHT. This finding is consistent with the reported literature, lending credibility to the model's underlying logic (48, 49, 53-56). Furthermore, neck circumference has also been shown to be a significant predictor of hypertension (75) and is associated with MHT in obese patients who present as clinically normotensive (76). The finding that max adventitial diameter is a significant predictor of MHT risk is noteworthy as, to our knowledge, there are no reported studies reporting a direct correlation between max adventitial diameter and MHT. While previous studies have demonstrated that adventitial diameter has been related to increase in cardiovascular disease in postmenopausal women (77) and has been associated with left ventricular mass (78), these studies did not directly focus on measurements of adventitial diameter.

An examination of the PDPs reveals how each feature in the model influences the model output. All but one the features that were selected were positively correlated with the risk of MHT. Interestingly, two of the features that were associated with diastole, the diastolic blood pressure, and the left ventricular mass at diastole, had a piece-wise linear features, where the risk of MHT increased only slightly or not at all and then drastically increased after a specific value was reached. It is worth noting that the diastolic blood pressure at which the risk of MHT started to increase drastically was approximately 80 mmHg, a value that has been reported by the Mayo clinic as the cutoff point where a patient is considered to have elevated blood pressure and a patient is classified as prehypertensive (79). It is also interesting that the relationship between the risk between MHT and diastolic blood differs from the relationship between the relationship between MHT and systolic blood pressure. Another finding that is worth noting is that there was

an observed decrease in the risk of MHT when the max adventitial diameter increased, making it the only selected feature to exhibit this inverse relationship.

There are several limitations to this study that are worth highlighting. First, there were several relevant features that previous studies have found to be associated with MHT that were not included in the dataset that the ML model was built on. Notable examples include atrial pulse wave velocity, monocyte chemoattractant protein-1, and C-reactive protein, which have been previously established as MHT predictors (67). Secondly, the size of the training set was relatively small for the number of features that were evaluated, and it is likely that the model performance would increase given a larger set of training data. Thirdly, the model was not validated on a test set from an external cohort; thus, the general utility of this model has not been tested. Fourth and finally, there are other feature selection methods and ML algorithms that were not included in the development of this model. Since each option that is currently available within the field of ML was not evaluated exhaustively, there could be potential feature selection methods or ML algorithms that could improve upon the performance of the model developed here. Overall, these study limitations highlight the need to expand upon the dataset by collecting more patient data, to include more relevant features into the dataset, to obtain more data from other sources to validate the model, and to continue to test and evaluate other ML algorithms that could improve the model performance.

4.5 Conclusion

This study proposed a heterogeneous stacking ensemble framework as a ML model for predicting the incidence of MHT in a young, apparently healthy population from a LMIC. The resulting model consisted of a two-layer stacking ensemble, with the first layer consisting of LR,

RF, XGB, and MLP classifiers as the base learners and the second layer consisting of a LR classifier as the meta learner. The proposed stacking ensemble model achieve a higher ROC AUC than each base model could achieve independently and demonstrated higher scoring metrics compared to the current "rule-of-thumb" classifier for MHT. The BorutaSHAP feature selection method identified several features for the model which could all be easily obtained from a single outpatient visit, making the adoption of the model in a clinical setting feasible. SHAP analysis found that systolic blood pressure and body weight were the two most important predictors of MHT. PDP revealed the relationships between each feature and the prediction of MHT and revealed that the features related to diastole had a relatively low impact on the presence MHT until they exceeded a set point where the influence of these features on MHT increased dramatically. Max adventitial diameter was identified as a novel predictor of MHT, suggesting that its exact relationship with MHT could be worth investigating in the future. Overall, this study demonstrated the promise of using a stacking ensemble learning ML model to detect MHT and further development of the model could potentially lead to a viable tool for aiding clinicians in identifying which patients are most at risk for MHT and need further evaluation through ABPM.

4.6 References

1. World Health Organisation Cardiovascular Disease 2023; https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1
2. Mathers, Colin D., Ties Boerma, and Doris Ma Fat. "Global and regional causes of death." *British medical bulletin* 92.1 (2009):7-32
3. Sverre E. Kjeldsen, "Hypertension and cardiovascular risk: General aspects." *Pharmacological research* 129 (2018): 95-99
4. World Heart Federation 2022; https://world-heart-federation.org/news/world-hypertension-day-taking-action-against-the-silent-epidemic-of-high-blood-pressure/
5. Kearney, Patricia M., et al. "Global burden of hypertension: analysis of worldwide data." *The lancet* 365.9455 (2005): 217-223.
6. Hunter, Paul G., Fiona A. Chapman, and Neeraj Dhaun. "Hypertension: Current trends and future perspectives." *British Journal of Clinical Pharmacology* 87.10 (2021): 3721-3736.
7. Messerli, Franz H., Bryan Williams, and Eberhard Ritz. "Essential hypertension." *The Lancet* 370.9587 (2007): 591-603.
8. Nguyen, Quang, et al. "Hypertension management: an update." *American health & drug benefits* 3.1 (2010): 47.
9. Huguet, Nathalie, et al. "Rates of undiagnosed hypertension and diagnosed hypertension without anti-hypertensive medication following the Affordable Care Act." *American Journal of Hypertension* 34.9 (2021): 989-998.)
10. Shukla, Anand N., et al. "Prevalence and predictors of undiagnosed hypertension in an apparently healthy western Indian population." *Advances in Epidemiology* 2015 (2015)

11. Essa, Enatnesh, et al. "Undiagnosed hypertension and associated factors among adults in Debre Markos town, North-West Ethiopia: A community-based cross-sectional study." *SAGE Open Medicine* 10 (2022): 20503121221094223.
12. Pickering, Thomas G., Kazuo Eguchi, and Kazuomi Kario. "Masked hypertension: a review." *Hypertension Research* 30.6 (2007): 479-488.
13. Stergiou, George S., et al. "Prognosis of white-coat and masked hypertension: International Database of HOme blood pressure in relation to Cardiovascular Outcome." *Hypertension* 63.4 (2014): 675-682.
14. Thakkar, Harsh V., Alun Pope, and Mahesan Anpalahan. "Masked hypertension: a systematic review." *Heart, Lung and Circulation* 29.1 (2020): 102-111.
15. Bobrie, Guillaume, et al. "Masked hypertension: a systematic review." *Journal of hypertension* 26.9 (2008): 1715-1725.
16. Trachsel, Lukas D., et al. "Masked hypertension and cardiac remodeling in middle-aged endurance athletes." *Journal of hypertension* 33.6 (2015): 1276-1283.
17. Stergiou, George S., et al. "White-coat hypertension and masked hypertension in children." *Blood pressure monitoring* 10.6 (2005): 297-300.
18. Ben-Dov, Iddo Z., et al. "In clinical practice, masked hypertension is as common as isolated clinic hypertension: predominance of younger men." *American journal of hypertension* 18.5 (2005): 589-593.
19. Berge, Hilde Moseby, et al. "High ambulatory blood pressure in male professional football players." *British journal of sports medicine* 47.8 (2013): 521-525.

20. Hermida, Ramón C., et al. "Ambulatory Blood Pressure Monitoring (ABPM) as the reference standard for diagnosis of hypertension and assessment of vascular risk in adults." *Chronobiology International* 32.10 (2015): 1329-1342.
21. Anstey, D. Edmund, et al. "Diagnosing masked hypertension using ambulatory blood pressure monitoring, home blood pressure monitoring, or both?." *Hypertension* 72.5 (2018): 1200-1207.
22. Stergiou, George S., et al. "Masked hypertension assessed by ambulatory blood pressure versus home blood pressure monitoring: is it the same phenomenon?." *American journal of hypertension* 18.6 (2005): 772-778.
23. Flynn, Joseph T., et al. "Update: ambulatory blood pressure monitoring in children and adolescents: a scientific statement from the American Heart Association." *Hypertension* 63.5 (2014): 1116-1135.
24. Shimbo, Daichi, et al. "Role of ambulatory and home blood pressure monitoring in clinical practice: a narrative review." *Annals of internal medicine* 163.9 (2015): 691-700.
25. Stergiou, George S., et al. "2021 European Society of Hypertension practice guidelines for office and out-of-office blood pressure measurement." *Journal of hypertension* 39.7 (2021): 1293-1302.
26. Abdalla, Marwah. "Ambulatory blood pressure monitoring: a complementary strategy for hypertension diagnosis and Management in low-Income and Middle-Income Countries." *Cardiology clinics* 35.1 (2017): 117-124.
27. Krittanawong, Chayakrit, et al. "Machine learning prediction in cardiovascular diseases: a meta-analysis." *Scientific reports* 10.1 (2020): 16057.

28. Sevakula, Rahul Kumar, et al. "State-of-the-art machine learning techniques aiming to improve patient outcomes pertaining to the cardiovascular system." *Journal of the American Heart Association* 9.4 (2020): e013924.
29. Motwani, Manish, et al. "Machine learning for prediction of all-cause mortality in patients with suspected coronary artery disease: a 5-year multicentre prospective registry analysis." *European heart journal* 38.7 (2017): 500-507.
30. Churpek, Matthew M., et al. "Multicenter comparison of machine learning methods and conventional regression for predicting clinical deterioration on the wards." *Critical care medicine* 44.2 (2016): 368.
31. Naidoo, Vivian, Fatima Suleman, and Varsha Bangalee. "The transition to universal health coverage in low and middle-income countries: new opportunities for community pharmacists." *Journal of Pharmaceutical Policy and Practice* 13 (2020): 1-3.
32. Schutte, Aletta E., et al. "2." *European journal of preventive cardiology* 26.5 (2019): 458-470.
33. Franklin, Stanley S., et al. "Masked hypertension: a phenomenon of measurement." *Hypertension* 65.1 (2015): 16-20.
34. Franklin, Stanley S., Eoin O'Brien, and Jan A. Staessen. "Masked hypertension: understanding its complexity." *European heart journal* 38.15 (2017): 1112-1118.
35. Booth III, John N., et al. "Evaluation of criteria to detect masked hypertension." *The Journal of Clinical Hypertension* 18.11 (2016): 1086-1094.
36. du Toit, Wessel L., et al. "Markers of arterial stiffness and urinary metabolomics in young adults with early cardiovascular risk: the African-PREDICT study." *Metabolomics* 19.4 (2023): 28.

37. Gramegna, Alex, and Paolo Giudici. "Shapley feature selection." *FinTech* 1.1 (2022): 72-80.

38. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

39. Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein and Russ B. Altman, Missing value estimation methods for DNA microarrays, BIOINFORMATICS Vol. 17 no. 6, 2001 Pages 520-525.

40. Stef van Buuren, Karin Groothuis-Oudshoorn (2011). "mice: Multivariate Imputation by Chained Equations in R". Journal of Statistical Software 45: 1-67.

41. Azur, Melissa J., et al. "Multiple imputation by chained equations: what is it and how does it work?." *International journal of methods in psychiatric research* 20.1 (2011): 40-49.

42. Jadhav, Anil, Dhanya Pramod, and Krishnan Ramanathan. "Comparison of performance of data imputation methods for numeric dataset." *Applied Artificial Intelligence* 33.10 (2019): 913-933.

43. Changyong, F. E. N. G., et al. "Log-transformation and its implications for data analysis." *Shanghai archives of psychiatry* 26.2 (2014): 105.

44. Kumar, Vipin, and Sonajharia Minz. "Feature selection: a literature review." *SmartCR* 4.3 (2014): 211-229.

45. Kursa, Miron B., Aleksander Jankowski, and Witold R. Rudnicki. "Boruta–a system for feature selection." *Fundamenta Informaticae* 101.4 (2010): 271-285.

46. Effrosynidis, Dimitrios, and Avi Arampatzis. "An evaluation of feature selection methods for environmental data." *Ecological Informatics* 61 (2021): 101224.

47. S. Ma and J. Huang, "Penalized feature selection and classification in bioinformatics", *Briefings in Bioinformatics*, vol. 9, no. 5, pp. 392-403, 2008.

48. Longo, Daniele, Francesca Dorigatti, and Paolo Palatini. "Masked hypertension in adults." *Blood pressure monitoring* 10.6 (2005): 307-310.

49. Hung, Ming-Hui, et al. "Prediction of masked hypertension and masked uncontrolled hypertension using machine learning." *Frontiers in Cardiovascular Medicine* 8 (2021): 778306.

50. Franklin, Stanley S., et al. "Masked hypertension: a phenomenon of measurement." *Hypertension* 65.1 (2015): 16-20.

51. Barochiner, Jessica, et al. "Predictors of masked hypertension among treated hypertensive patients: an interesting association with orthostatic hypertension." *American journal of hypertension* 26.7 (2013): 872-878.

52. Mallion, Jean-Michel, et al. "Predictive factors for masked hypertension within a population of controlled hypertensives." *Journal of hypertension* 24.12 (2006): 2365-2370.

53. Bobrie, Guillaume, et al. "Masked hypertension: a systematic review." *Journal of hypertension* 26.9 (2008): 1715-1725.

54. Hänninen, Marjo-Riitta A., et al. "Determinants of masked hypertension in the general population: the Finn-Home study." *Journal of hypertension* 29.10 (2011): 1880-1888.

55. Franklin, Stanley S., Eoin O'Brien, and Jan A. Staessen. "Masked hypertension: understanding its complexity." *European heart journal* 38.15 (2017): 1112-1118.

56. Booth III, John N., et al. "Evaluation of criteria to detect masked hypertension." *The Journal of Clinical Hypertension* 18.11 (2016): 1086-1094.

57. Gramegna, Alex, and Paolo Giudici. "Shapley feature selection." *FinTech* 1.1 (2022): 72-80.

58. Kursa, Miron B., Aleksander Jankowski, and Witold R. Rudnicki. "Boruta–a system for feature selection." *Fundamenta Informaticae* 101.4 (2010): 271-285.

59. Hart, Sergiu. "Shapley value." *Game theory*. London: Palgrave Macmillan UK, 1989. 210-216.

60. Keany, Eoghan. "BorutaShap: A wrapper feature selection method which combines the Boruta feature selection algorithm with Shapley values." *Zenodo* (2020).

61. Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems* 30 (2017).

62. Nti, Isaac Kofi, Owusu Nyarko-Boateng, and Justice Aning. "Performance of machine learning algorithms with different K values in K-fold cross-validation." *J. Inf. Technol. Comput. Sci* 6 (2021): 61-71.

63. LaValley, Michael P. "Logistic regression." *Circulation* 117.18 (2008): 2395-2399.

64. Breiman, Leo. "Random forests." *Machine learning* 45 (2001): 5-32.

65. Kotsiantis, Sotiris B. "Decision trees: a recent overview." *Artificial Intelligence Review* 39 (2013): 261-283.

66. Cinà, Giovanni, et al. "Why we do need explainable ai for healthcare." *arXiv preprint arXiv:2206.15363* (2022).

67. Thompson, Jane ES, et al. "Masked hypertension and its associated cardiovascular risk in young individuals: the African-PREDICT study." Hypertension Research 39.3 (2016): 158-165.

68. Hung, Ming-Hui, et al. "Prediction of masked hypertension and masked uncontrolled hypertension using machine learning." *Frontiers in Cardiovascular Medicine* 8 (2021): 778306.

69. Bae, Sunjae, et al. "Machine Learning–Based Prediction of Masked Hypertension Among Children With Chronic Kidney Disease." *Hypertension* 79.9 (2022): 2105-2113.

70. Shih, Ling-Chieh, et al. "Prediction of white-coat hypertension and white-coat uncontrolled hypertension using machine learning algorithm." *European Heart Journal-Digital Health* 3.4 (2022): 559-569.

71. Meng, Hong, et al. "Nomogram based on clinical features at a single outpatient visit to predict masked hypertension and masked uncontrolled hypertension: A study of diagnostic accuracy." *Medicine* 101.49 (2022).

72. Lip, Stefanie, et al. "Machine Learning Based Models for Predicting White-Coat and Masked Patterns of Blood Pressure." *Journal of Hypertension* 39 (2021): e69.

73. Zhu, Xiuqing, et al. "An interpretable stacking ensemble learning framework based on multi-dimensional data for real-time prediction of drug concentration: The example of olanzapin

74. Khan, Asfandyar, et al. "Cardiovascular and Diabetes Diseases Classification Using Ensemble Stacking Classifiers with SVM as a Meta Classifier." *Diagnostics* 12.11 (2022): 2595. e." *Frontiers in Pharmacology* 13 (2022): 975855.

75. Soitong, Panuwat, et al. "Association of neck circumference and hypertension among adults in a rural community Thailand: A cross-sectional study." *Plos one* 16.8 (2021): e0256260.

76. Cunha, A., et al. "[PP. 34.01] NECK CIRCUMFERENCE AND WAIST-HIP RATIO, INDEPENDENTLY OF BODY MASS INDEX, ARE MORE RELATED WITH MASKED HYPERTENSION IN OBESE NORMOTENSIVE INDIVIDUALS WITH OBSTRUCTIVE SLEEP APNEA." *Journal of Hypertension* 34 (2016): e329.

77. Patel, Ami S., et al. "Cardiovascular risk factors associated with enlarged diameter of the abdominal aortic and iliac arteries in healthy women." *Atherosclerosis* 178.2 (2005): 311-317.

78. Polak, Joseph F., et al. "Associations of cardiovascular risk factors, carotid intima-media thickness and left ventricular mass with inter-adventitial diameters of the common carotid artery: the Multi-Ethnic Study of Atherosclerosis (MESA)." *Atherosclerosis* 218.2 (2011): 344-349.

79. Mayo Clinc, 2023; https://www.mayoclinic.org/diseases-conditions/high-blood-pressure/symptoms-causes/syc-20373410

80. Samb, Mouhamadou Lamine, et al. "A novel RFE-SVM-based feature selection approach for classification." *International Journal of Advanced Science and Technology* 43.1 (2012): 27-36.

81. Brereton, Richard G., and Gavin R. Lloyd. "Support vector machines for classification and regression." *Analyst* 135.2 (2010): 230-267.

82. Ranstam, Jonas, and J. A. Cook. "LASSO regression." *Journal of British Surgery* 105.10 (2018): 1348-1348.

83. Wu, Jia, et al. "Hyperparameter optimization for machine learning models based on Bayesian optimization." *Journal of Electronic Science and Technology* 17.1 (2019): 26-40.

84. Hinton, Geoffrey E. “Connectionist learning procedures.” Artificial intelligence 40.1 (1989): 185-234.

85. Chen, Tianqi, et al. "Xgboost: extreme gradient boosting." R package version 0.4-2 1.4 (2015): 1-4.

CHAPTER 5

Conclusions and Future Recommendations

5.1 Conclusions

The objective of this dissertation was to expand the foundational knowledge of MSC mechanobiology and to develop novel experimental and computational platforms to aid researchers and clinicians in studying and understanding systems mechanobiology. In Chapter 2, mechanical stimulation was shown to influence MSC behavior by altering cell metabolic activity, proliferation, viability, and cytokine production. Furthermore, the biological response of each MSC type that was evaluated was dependent on the specific tissue source. In Chapter 3, a 3-D printed bioreactor was developed that could apply mechanical stimulation to multiple cell-seeded biological scaffolds for an extending period without harming cell viability. This study also demonstrated a process by which both tissue-level and cell-level strains could be evaluated without having to disrupt the cells in culture or terminate the experiment. In Chapter 4, a novel stacking ML model for MHT risk classification was created using patient healthcare data from a young, apparently healthy population from a LMIC. The stacking ML model was shown to outperform other ML models that were evaluated, including LR, RF, BGBoost, and MLP classifier models. This ML model also performed better than the current "rule-of-thumb" that has been suggested to access patient risk for MHT. The BorutaSHAP feature selection method demonstrated that a highly accurate model could be built using only a few, easy-to-obtain features from a single out-patient clinical visit. SHAP analysis and PDP evaluation revealed that the top 3 most important predictors of MHT were SBP, BW, and MAP and that there was a linear relationship between these factors and a patient's risk for MHT.

5.2 Study Limitations and Future Recommendations

Each of the studies presented in this work had limitations that should be acknowledged to provide insights that could be helpful in improving future studies related to these topics. In the first aim of this study, a commercial bioreactor was utilized to culture the cells in monolayer and apply cyclic tensile strain to the cells throughout the experiment. Previous research has shown that there are significant differences in how cells behave in 2-D vs. 3-D microenvironments and that the behavior of cells cultured in monolayer may not reflect the behavior of those cells in a physiological environment [1]. Therefore, in future studies, it would be important to modify the experimental setup to incorporate a 3D cell culture system instead of a 2D cell culture system. One way that this could be accomplished would be to culture the cells in a biological scaffold such as tissue sample similar to that used in Chapter 3, a collagen or fibrinogen hydrogel, or an electrospun polystyrene scaffold [2,3]. A bioreactor such as the one developed in Chapter 3 could then be utilized to culture these cell-seeded scaffold and apply mechanical stimulation. Another limitation of this study was means by which mechanical stimulation was applied to the cells. While the magnitude and frequency of the cyclic tensile stretch that was implemented in this study attempted to simulate the magnitude and frequency of tensile stretch seen in the AF of a degenerate IVD and the loading protocol that was implemented was designed to reflect the loading regime of an individual who was active for 16 hours a day and then resting for 8 hours a day, neither of these experimental parameters truly capture the complex and dynamic loads that cell experienced within the context of the native AF microenvironment. This is due to the limitations of the bioreactor, which could only apply simple cyclic tensile loading. Future studies could improve upon the work presented here by utilizing a more complex bioreactor that is

capable of incorporating more complex loading regimes such as uniaxial tension in conjunction with shear stress, biaxial loading, or various combinations of these three loading strategies. This would enable the experiment better reflect the physiological forces experienced in the AF microenvironment.

In the second aim of this study, the bioreactor that was developed could apply mechanical stimulation in the form of tensile strain and could culture cells in 3D on a biological scaffold; however, limitations in the loading process resulted in large sample-to-sample variation. As a result, the first recommendation for future work in this area would be to implement a pre-tensioning system into the actuator arms. This could easily be accomplished by implementing a design similar the bioreactor designed by Somers and Grayson [4]. Another recommendation would be to utilize a confocal microscope instead of a wide-field fluorescent microscope and perform a second study focused on the analysis of the cell strain using the procedure described in Chapter 3 to determine whether this method of imaging could help reduce the large variations in the observed cell strains by eliminating any diffuse light that could potentially confound the segmentation of the cell membrane during analysis.

In the final aim of this study, the primary limitations of the ML model were the limited number of training samples and the lack of an external cohort to validate the model. The solution to both of these limitations would be to identify a dataset with a similar patient population and incorporate that new dataset into the model. As an initial step, features from the new dataset that did not overlap with the features from the African PREDICT dataset could be removed ensure that both datasets shared common features. Then statistical comparisons between features from each dataset could be evaluated to check for statistical differences. If there were not statistical differences, then the model could be trained on the entire African PREDICT dataset and

evaluated on the new cohort. If there were statistical differences, then the two datasets could be combined, shuffled, and then split into testing and training sets. Another limitation was the number of ML classifiers that were evaluated. A future study could examine how the implementation of other ML classifier algorithms such as SVM, bagging, or adaboost classifiers into the ML pipeline could improve the overall performance of the stacking classifier. Finally, the model could be implemented into a web-based platform or application that could allow clinicians to try the model in their practice. This could enable real-time evaluation of the model performance as well as enable the model to re-train and learn from new data.

5.3 References

1. Duval, Kayla, et al. "Modeling physiological events in 2D vs. 3D cell culture." *Physiology* 32.4 (2017): 266-277.

2. Tibbitt, Mark W., and Kristi S. Anseth. "Hydrogels as extracellular matrix mimics for 3D cell culture." Biotechnology and bioengineering 103.4 (2009): 655-663.

3. Baker, Simon C., et al. "Characterisation of electrospun polystyrene scaffolds for three-dimensional in vitro biological studies." *Biomaterials* 27.16 (2006): 3136-3146.

4. Somers, Sarah M., and Warren L. Grayson. "Protocol for the Use of a Novel Bioreactor System for Hydrated Mechanical Testing, Strained Sterile Culture, and Force of Contraction Measurement of Tissue Engineered Muscle Constructs." *Frontiers in Cell and Developmental Biology* 9 (2021): 66103

APPENDICES

Appendix A

African PREDICT Feature Sets

A-1.1 Complete List of African PREDICT Measured Features

Category	Features
General	s_pp_number, dob, deceased, icd10_code, dod, sex, ethnicity, ses_skill, ses_education, ses_income, ses_score, ses_class, age_b, ear_temp
Anthropometry	bh, bw, bmi, bsa, rfm, wc, hc, waist_hip_ratio, nc
Office Blood Pressure	bp_apparatus, l_sbp_1, l_sbp_2, r_sbp_1, r_sbp_2, l_dbp_1, l_dbp_2, r_dbp_1, r_dbp_2, pulse_pressure, clinic_map, clinic_bp_status, bp_grade, isolated_sbp_ht, sustained_ht, white_coat, l_hr_1, l_hr_2, r_hr_1, r_hr_2
Medication	meds_names, s_meds_a, s_meds_b, s_meds_d, s_meds_g, s_meds_h, s_meds_j, s_meds_l, s_meds_m, s_meds_n, s_meds_p, s_meds_r, s_meds_s, s_meds_v, s_meds_c, s_meds_c_cardiac, s_meds_c_ht, s_meds_c_diur, s_meds_c_bb, s_meds_c_ccb, s_meds_c_ren, s_meds_c_lipid, s_meds_c_other
Physical Activity Data	days_actiheart, rmr, aee, dit, tee, pal, activity, physical_activity_actiheart_complete, vig_work_minutes, vig_work_met_min, minutes_moderate_work_week, moderate_work_met_minutes, travel_minutes_week, travel_met_minutes, minutes_vig_exerc, vig_exercise_met, minutes_moderate_exerc, moderate_exercise_met, sedentary_minutes, vig_act_day, mod_act_day, mvpa, sedentary_mvpa, infl_24
Ambulatory Blood Pressure Data	nr_infl, perc_succ_infl, seventy_perc_yn, day_infl, night_infl, abpm_24_s, abpm_24_d, abpm_24_pp, map_24h, abpm_overall_ht, abpm_24_ht, abpm_d_sbp, abpm_d_dbp, abpm_d_pp, map_day, abpm_d_ht, abpm_n_sbp, abpm_n_dbp, abpm_n_pp, map_night, pulse_24, pulse_d, pulse_night, hrv_ti, sphygmocor_sbp1, sphygmocor_dbp1, sphygmocor_sbp2, sphygmocor_dbp2, csbp_1, cdbp_1, csbp_2, cdbp_2, cpp1, cpp2, cmp_1, cmp_2, ap_1, ap_2, cai_1, cai_2, sphyg_hr1, sphyg_hr2
Sonar Data	l_imtn_mean, l_imtn_min, l_imtn_max, l_imtn_std, l_imtn_length, l_imtf_mean, l_imtf_min, l_imtf_max, l_imtf_std, l_imtf_length, l_ld_mean, l_ld_min, l_ld_max, l_ld_std, l_ld_legth, l_ad_mean, l_ad_min, l_ad_max, l_ad_std, l_ad_length, l_rough_near, l_rough_far, l_cca_ps, l_cca_ed, l_ica_ps, l_ica_ed, l_plaque_score, l_plaque_at, lcswa, l_osbp, l_odbp, l_pp, l_acimt, l_delta_ld, l_strain, l_distensibility, l_compliance, l_bsi, l_pem, l_yem, r_imtn_mean, r_imtn_min, r_imtn_max, r_imtn_std, r_imtn_length, r_imtf_mean, r_imtf_min, r_imtf_max, r_imtf_std, r_imtf_length, r_ld_mean, r_ld_min, r_ld_max, r_ld_std, r_ld_legth, r_ad_mean, r_ad_min, r_ad_max, r_ad_std, r_ad_length, r_rough_near, r_rough_far, r_cca_ps, r_cca_ed, r_ica_ps, r_ica_ed, r_plaque_score, r_plaque_at, rcswa, r_osbp, r_odbp, r_pp, r_acimt, r_delta_ld, r_strain, r_distensibility, r_compliance, r_bsi, r_pem, r_yem, ivsd, ivss, lvidd, lvids, lvpwd, lvpws, rwt, edv, esv, ef, sv, fs, lvm_cube, ilvm_cube, lvd_mass, lvd_mass_ase, lvs_mass_ase, lvs_mass, lad, aod, la_ao, mv_e_vel, mv_dect, mv_dec_slope, mv_a_vel, mv_ea_ratio, m_e_fill, e_e_prime, mv_vmax, mv_vmean, mv_maxpg, mv_meanpg, mv_vti, av_vmax, av_maxpg, av_vmean, av_meanpg, av_env_ti, av_vti, rvids, rvidd, ra_diam, ra_area, rald, raad, raedv_al, raedv_mod, rals, raas, raesv_al, raesv_mod, avc, glps_lax, glps_a4c, glps_a2c, glps_avg
Milliplex Data	fractalkine, gm_csf, ifn_gamma, il_1beta, il_2, il_4, il_5, il_6m, il_7, il_8, il_10, il_12, il_13, il_17a, il_21, il_23, itac, mip_1_alpha, mip_1_beta, mip_3_alpha, tnf_alpha_m, multiplex_analyses_results_complete
Biochemical Analysis Data	ggt, chol, hdl, ldl, trig, glu, crp, cotinine, cot_nominal
RAS Fingerprint Analysis	ang_ii_pmol, ang_1_7_pmol, ang_1_9_pmol, ang_1_10_pmol, ang_iii_pmol, ang_3_7_pmol, ang_1_5_pmol, ang_iv_pmol, ang_2_7_pmol, ang_2_10_pmol, ras_aldosterone_pmol, ras_ace, ras_renin, aa2_ratio, ang_ii_pg, ang_1_7_pg, ang_1_9_pg, ang_1_10_pg, ang_iii_pg, ang_3_7_pg, ang_1_5_pg, ang_iv_pg, ang_2_7_pg, ang_2_10_pg, ras_aldosterone_pg
Questionaire Data	smoke, smoke_past, smoke_quit, age_smoke_start, smoke_occasion, smoke_type___1, smoke_type___2, smoke_type___3, smoke_type___4, smoke_type___5, smoke_type___6, smoke_type___7, smoke_type___8, smoke_type___9, type_other_smoke, number_filter_cigarette, number_rolled_cigarette, number_chewing_tobacco, number_pipe, number_cigars, number_hubbly, number_snuff, number_dagga, number_other, alcohol, type_alcohol___1, type_alcohol___2, type_alcohol___3, type_alcohol___4, type_alcohol___5, type_alcohol___6, type_alcohol___7, type_alcohol___8, type_other, beer_dumpie_number, beer_quart_number, beer_boxes_number, beer_mahewe_number, wine_number, tots_spirits_number, cider_number, alcohol_other_number, age_alcohol_started, number_weekend_drinks, homemade_beer, homemade_beer_container, other_container, size_container, number_homemade_beer, strenuous_exercise, moderate_exercise, mild_exercise, leisure_activity_times, heart_attack, stroke, cancer, tuberculosis, hypertension, heart_disease, cholesterol, diabetes, health_problems_other, type_health_problems, year_health_problems, year_hypertension, year_heart_disease, year_high_cholesterol, diabetes_type, year_diabetes, diabetes_manage_other___1, diabetes_manage_other___2, diabetes_manage_other___3, diabetes_manage_other___4, diabetes_manage_other___5, other_manage_diabetes, mother_heart_disease, father_heart_disease, mother_stroke, father_stroke, mother_diabetes, father_diabetes, mother_hypertension, father_hypertension, mother_cholesterol, father_cholestrol, gender_self_reported, menstruate, contraceptive_pill, contraceptive_injection, contraceptive_implant, mirena, hysterectomy, hormone_replacement, age_hrt, hiv_status_first_follow_up, participant_fill_in_sheet_complete

A-1.2 List of Features Dropped from the Original African PREDICT Dataset

Category	Features
General	s_pp_number, dob, deceased, icd10_code, dod, ses_skill, ses_education, ses_income, ses_score, ses_class, ear_temp
Office Blood Pressure	bp_apparatus, clinic_bp_status, bp_grade, isolated_sbp_ht, sustained_ht, white_coat
Medication	meds_names
Physical Activity Data	days_actiheart, physical_activity_actiheart_complete, infl_24
Ambulatory Blood Pressure Data	nr_infl, perc_succ_infl, seventy_perc_yn, day_infl, night_infl, abpm_24_s, abpm_24_d, abpm_24_pp, map_24h, abpm_overall_ht, abpm_24_ht, abpm_d_sbp, abpm_d_dbp, abpm_d_pp, map_day, abpm_d_ht, abpm_n_sbp, abpm_n_dbp, abpm_n_pp, map_night, pulse_24, pulse_d, pulse_night, hrv_ti, sphygmocor_sbp1, sphygmocor_dbp1, sphygmocor_sbp2, sphygmocor_dbp2, csbp_1, cdbp_1, csbp_2, cdbp_2, cpp1, cpp2, cmp_1, cmp_2, ap_1, ap_2, cai_1, cai_2, sphyg_hr1, sphyg_hr2
Sonar Data	rvids, rvidd, ra_diam, ra_area, rald, raad, raedv_al, raedv_mod, rals, raas, raesv_al, raesv_mod, avc, glps_lax, glps_a4c, glps_a2c, glps_avg
Milliplex Data	multiplex_analyses_results_complete
Biochemical Analysis Data	cot_nominal
RAS Fingerprint Analysis	ang_ii_pg, ang_1_7_pg, ang_1_9_pg, ang_1_10_pg, ang_iii_pg, ang_3_7_pg, ang_1_5_pg, ang_iv_pg, ang_2_7_pg, ang_2_10_pg, ras_aldosterone_pg
Questionaire Data	smoke, smoke_past, smoke_quit, age_smoke_start, type_other_smoke, number_filter_cigarette, number_rolled_cigarette, number_chewing_tobacco, number_pipe, number_cigars, number_hubbly, number_snuff, number_dagga, number_other, type_other, beer_dumpie_number, beer_quart_number, beer_boxes_number, beer_mahewe_number, wine_number, tots_spirits_number, cider_number, alcohol_other_number, age_alcohol_started, number_weekend_drinks, homemade_beer, homemade_beer_container, other_container, size_container, number_homemade_beer, health_problems_other, type_health_problems, year_health_problems, year_hypertension, year_heart_disease, year_high_cholesterol, diabetes_type, year_diabetes, diabetes_manage_other___1, diabetes_manage_other___2, diabetes_manage_other___3, diabetes_manage_other___4, diabetes_manage_other___5, other_manage_diabetes, gender_self_reported, menstruate, contraceptive_pill, contraceptive_injection, contraceptive_implant, mirena, hysterectomy, hormone_replacement, age_hrt, hiv_status_first_follow_up, participant_fill_in_sheet_complete

A-1.3 Complete List of Transformed African PREDICT Categorical Features

Category	Features
Office Blood Pressure	sbp, dbp, hr, cpp, map, cmp, pp, ap, cai
Sonar Data	imtn_mean','imtn_min','imtn_max','imtn_std','imtn_length','imtf_mean','strain','distensibility','compliance','bsi','pem','yem' 'imtf_min','imtf_max','imtf_std','imtf_length','ld_mean','ld_min','ld_max','ld_std','ld_legth','ad_mean','ad_min','ad_max','ad_std','ad_length','rough_near','rough_far','cca_ps','cca_ed','ica_ps','ica_ed','cswa','acimt','delta_ld'
Questionaire Data	diabetes, cholesterol, heart_disease, mother_hypertension, father_hypertension, mother_cholesterol, father_cholestrol, mother_heart_disease, father_heart_disease, mother_stroke, father_stroke, mother_diabetes, father_diabetes

A-1.4 Continuous Feature Normality Test Results

Feature	Shapiro-Wilk Test P Value	Result
age_b	7.15E-16	Not Normal
sbp	1	Normal
dbp	1	Normal
map	0.0908278	Normal
pp	1.20E-06	Not Normal
bh	0.017438231	Not Normal
bw	1.66E-19	Not Normal
wc	1	Normal
hc	1.99E-11	Not Normal
nc	1	Normal
bmi	7.12E-23	Not Normal
waist_hip_ratio	1	Normal
chol	1	Normal
trig	1	Normal
hdl	1	Normal
ldl	1	Normal
glu	1	Normal
cotinine	1	Normal
ggt	1	Normal
lvm_cube	1.10E-14	Not Normal

Appendix B

Aim 2 MATLAB Scripts

B-1.1 Cell Strain Analysis Script

```
clc,clear,close all

% Set filter size
filter_size = 10;

% Set distance threshold
dist_thresh = 25;

% Set simliarity threshold
similiarity_thresh = 0.87;

% Set window dimensions
window_h = 50;
window_w = window_h;

% Load images
[original_image,original_cell_boundaries,image_h,image_w,path_dir,images] =
load_images(filter_size);

% Cycle through each of the cells in the original image
for i = 1:length(original_cell_boundaries)
    % Create folder for saving data
    mkdir(fullfile(path_dir,'Analysis_Output',string(i)));

    % For each cell in the original image, get the original mask, original
properties, original cropped mask, and original cropped properties

[original_cell_mask,original_cell_mask_props,original_cell_mask_cropped,origi
nal_cell_mask_cropped_props] =
process_image(original_cell_boundaries{i},image_h,image_w,window_h,window_w);

    first_cell_mask_cropped = original_cell_mask_cropped;

    % Cycle through each of the deformed images
    for j = 2:length(images)

        % Get deformed image and the cell boundaries in that image
        [deformed_image,deformed_cell_boundaries] =
process_deformed_image(path_dir,images(j).name,image_h,image_w,filter_size);

        % Identify the distances between the original cell being analyzed and
the cells in the deformed image
        [dists,idx] = get_cell_dists(original_cell_boundaries,
deformed_cell_boundaries, original_cell_mask_props, dist_thresh, image_h,
image_w);

        if ~isempty(idx)
            %similiarities = [];
            cell_props = [];
            similarity = [];
            for k = 1:length(idx)
                indice = idx(k);
```

```
[deformed_cell_mask,deformed_cell_mask_props,deformed_cell_mask_cropped,defor
med_cell_mask_cropped_props] =
process_image(deformed_cell_boundaries{indice},image_h,image_w,window_h,windo
w_w);
                similiarities(k) =
dice(original_cell_mask_cropped,deformed_cell_mask_cropped);
                if k == 1
                    props =
regionprops(original_cell_mask_cropped,'Centroid','Area','MajorAxisLength','M
inorAxisLength','Orientation','Eccentricity','Solidity','Circularity','Extent
');
                    if length(props) > 1
                        [~, props_idx] = max([props.Area]);
                        props = props(props_idx);
                    end
                    cell_props.properties(1) = props;
                    props =
regionprops(deformed_cell_mask_cropped,'Centroid','Area','MajorAxisLength','M
inorAxisLength','Orientation','Eccentricity','Solidity','Circularity','Extent
');
                    if length(props) > 1
                        [~, props_idx] = max([props.Area]);
                        props = props(props_idx);
                    end
                    cell_props.properties(k+1) = props;
                    eccentricity_similarity =
abs(cell_props.properties(k+1).Eccentricity-
cell_props.properties(1).Eccentricity)/cell_props.properties(1).Eccentricity;
                    solidarity_similarity =
abs(cell_props.properties(k+1).Solidity-
cell_props.properties(1).Solidity)/cell_props.properties(1).Solidity;
                    circularity_similarity =
abs(cell_props.properties(k+1).Circularity-
cell_props.properties(1).Circularity)/cell_props.properties(1).Circularity;
                    extent_similarity =
abs(cell_props.properties(k+1).Extent-
cell_props.properties(1).Extent)/cell_props.properties(1).Extent;

                    similarity(k) =
((dice(original_cell_mask_cropped,deformed_cell_mask_cropped) + (1-
(eccentricity_similarity + solidarity_similarity + circularity_similarity +
extent_similarity)/4)))/2;
                else
                    props =
regionprops(deformed_cell_mask_cropped,'Centroid','Area','MajorAxisLength','M
inorAxisLength','Orientation','Eccentricity','Solidity','Circularity','Extent
');
                    if length(props) > 1
                        [~, props_idx] = max([props.Area]);
                        props = props(props_idx);
                    end
                    cell_props.properties(k+1) = props;
                    eccentricity_similarity =
abs(cell_props.properties(k+1).Eccentricity-
cell_props.properties(1).Eccentricity)/cell_props.properties(1).Eccentricity;
```

```
                    solidarity_similarity =
abs(cell_props.properties(k+1).Solidity-
cell_props.properties(1).Solidity)/cell_props.properties(1).Solidity;
                    circularity_similarity =
abs(cell_props.properties(k+1).Circularity-
cell_props.properties(1).Circularity)/cell_props.properties(1).Circularity;
                    extent_similarity =
abs(cell_props.properties(k+1).Extent-
cell_props.properties(1).Extent)/cell_props.properties(1).Extent;

                    similarity(k) =
((dice(original_cell_mask_cropped,deformed_cell_mask_cropped) + (1-
(eccentricity_similarity + solidarity_similarity + circularity_similarity +
extent_similarity)/4)))/2;
                end
            end
            %[max_similiarity,max_similiarity_index] = max(similiarities);
            [max_similiarity,max_similiarity_index] = max(similarity);
            max_sim(j-1) = max_similiarity;

            if max_similiarity > similiarity_thresh
                indice = idx(max_similiarity_index);

[deformed_cell_mask,deformed_cell_mask_props,deformed_cell_mask_cropped,defor
med_cell_mask_cropped_props] =
process_image(deformed_cell_boundaries{indice},image_h,image_w,window_h,windo
w_w);
            else
                for l = j:length(images)
                    deformed_image =
imread(fullfile(path_dir,images(l).name));
                    fig1 = figure;
                    imshow(deformed_image)
                    image_name = strcat('Tracked-Image-',string(l-1),'-Cell-
',string(i));
                    saveas(fig1,
fullfile(path_dir,'Analysis_Output',string(i),strcat(image_name,'.tiff')));
                    close;
                end
                break
            end
        else
            for l = j:length(images)
                deformed_image = imread(fullfile(path_dir,images(l).name));
                fig1 = figure;
                imshow(deformed_image);
                image_name = strcat('Tracked-Image-',string(l-1),'-Cell-
',string(i));
                saveas(fig1,
fullfile(path_dir,'Analysis_Output',string(i),strcat(image_name,'.tiff')));
                close;
            end
            break
        end

[original_cell_mask,original_cell_mask_props,original_cell_mask_cropped,origi
```

```
nal_cell_mask_cropped_props] =
process_image(deformed_cell_boundaries{indice},image_h,image_w,window_h,windo
w_w);
        fig1 = figure;
        imshow(deformed_image);
        hold on
        for k = 1:length(idx)
            indice2 = idx(k);

[cell_mask,cell_mask_props,cell_mask_cropped,cell_mask_cropped_props] =
process_image(deformed_cell_boundaries{indice2},image_h,image_w,window_h,wind
ow_w);

plot(cell_mask_props.Centroid(1),cell_mask_props.Centroid(2),'b*');
        end
        hold off
        hold on

plot(deformed_cell_mask_props.Centroid(1),deformed_cell_mask_props.Centroid(2
),'r*');
        hold off
        image_name = strcat('Tracked-Image-',string(j-1),'-Cell-',string(i));
        saveas(fig1,
fullfile(path_dir,'Analysis_Output',string(i),strcat(image_name,'.tiff')));
        close;

        if j == 2
            original_props =
regionprops(first_cell_mask_cropped,'Centroid','Area','MajorAxisLength','Mino
rAxisLength','Orientation','Eccentricity','Solidity','Circularity','Extent');
            if length(original_props) > 1
                [~, original_props_idx] = max([original_props.Area]);
                original_props = original_props(original_props_idx);
            end
            cell_props_matrix(1,:) =
[original_props.Centroid(1),original_props.Centroid(2),original_props.Area,or
iginal_props.MajorAxisLength,original_props.MinorAxisLength,original_props.Or
ientation,original_props.Eccentricity,original_props.Solidity,original_props.
Circularity,original_props.Extent];
        end
        deformed_props =
regionprops(deformed_cell_mask_cropped,'Centroid','Area','MajorAxisLength','M
inorAxisLength','Orientation','Eccentricity','Solidity','Circularity','Extent
');
        if length(deformed_props) > 1
            [~, deformed_props_idx] = max([deformed_props.Area]);
            deformed_props = deformed_props(deformed_props_idx);
        end
        cell_props_matrix(j,:) =
[deformed_props.Centroid(1),deformed_props.Centroid(2),deformed_props.Area,de
formed_props.MajorAxisLength,deformed_props.MinorAxisLength,deformed_props.Or
ientation,deformed_props.Eccentricity,deformed_props.Solidity,deformed_props.
Circularity,deformed_props.Extent];

        t = linspace(0,2*pi,50);
        a = original_props.MajorAxisLength/2;
```

```
        b = original_props.MinorAxisLength/2;
        phi = deg2rad(-original_props.Orientation);
        ellipse_x1 = a*cos(t)*cos(phi) - b*sin(t)*sin(phi);
        ellipse_y1 = a*cos(t)*sin(phi) + b*sin(t)*cos(phi);
        orig_mat = [ellipse_x1;ellipse_y1];
        a = deformed_props.MajorAxisLength/2;
        b = deformed_props.MinorAxisLength/2;
        phi = deg2rad(-deformed_props.Orientation);
        ellipse_x2 = a*cos(t)*cos(phi) - b*sin(t)*sin(phi);
        ellipse_y2 = a*cos(t)*sin(phi) + b*sin(t)*cos(phi);
        deform_mat = [ellipse_x2;ellipse_y2];

        fig2 = figure;
        subplot(1,2,1)
        imshow(first_cell_mask_cropped);
        hold on
        plot(ellipse_x1+window_w/2,ellipse_y1+window_h/2,'r*');
        hold off
        subplot(1,2,2)
        imshow(deformed_cell_mask_cropped);
        hold on
        plot(ellipse_x2+window_w/2,ellipse_y2+window_h/2,'r*');
        hold off
        image_name2 = strcat('Cell_Comparison_Images',string(j-1),'-Cell-
',string(i));
        saveas(fig2,
fullfile(path_dir,'Analysis_Output',string(i),strcat(image_name2,'.tiff')));
        close;

        F = deform_mat/orig_mat;
        C = F*F';
        I = eye(length(F));
        e = ((1/2)*(C-I))*100;

        cell_F(j,:) = [F(1,1),F(1,2),F(2,1),F(2,2)];
        cell_E(j,:) = [e(1,1),e(1,2),e(2,1),e(2,2)];
    end
cell_props_mat = array2table(cell_props_matrix);
cell_props_mat.Properties.VariableNames =
{'Centroid_X','Centroid_Y','Area','MajorAxisLength','MinorAxisLength','Orient
ation','Eccentricity','Solidity','Circularity','Extent'};
writetable(cell_props_mat,fullfile(path_dir,'Analysis_Output',string(i),'cell
_props_mat.csv'));
writematrix(cell_F,fullfile(path_dir,'Analysis_Output',string(i),'cell_F_matr
ix.csv'));
writematrix(cell_E,fullfile(path_dir,'Analysis_Output',string(i),'cell_E_tota
l.csv'));
end
```

B-1.2 load images Function

```
function [image,cell_boundaries,image_h,image_w,path_dir,images] =
load_images(filter_size)
    % Load Images to Analyze
    disp('Select Folder with Images to Process');
    path_dir = uigetdir('Select Folder with Images to Process');
    images = dir(fullfile(path_dir,'*.tiff'));

    % Read in image to process
    image = imread(fullfile(path_dir,images(1).name));

    % Get dimensions of image
    [image_h,image_w] = size(image);

    % Filter image
    image = bwareafilt(imbinarize(image),[filter_size image_h*image_w]);

    % Get Cell Objects from First Image
    cell_boundaries = bwboundaries(image,'noholes');
end
```

B-1.3 process image Function

```
function
[cell_mask,cell_mask_props,cell_mask_cropped,cell_mask_cropped_props] =
process_image(cell_boundary,image_h,image_w,window_h,window_w)
    cell_x = cell_boundary(:,1);
    cell_y = cell_boundary(:,2);
    cell_mask = poly2mask(cell_y,cell_x,image_h,image_w);
    cell_mask_props = regionprops(cell_mask,'Centroid','BoundingBox','Area');
    if length(cell_mask_props) > 1
        [~, idx] = max([cell_mask_props.Area]);
        cell_mask_props = cell_mask_props(idx);
    end
    cell_mask = imtranslate(cell_mask,[round(-
cell_mask_props.Centroid(1)+window_w/2),round(-
cell_mask_props.Centroid(2)+window_h/2)]);
    cell_mask_cropped = imcrop(cell_mask,[0,0,window_h,window_w]);
    cell_mask_cropped_props =
regionprops(cell_mask_cropped,'Centroid','Area','MajorAxisLength','MinorAxisL
ength','Orientation','Eccentricity');
end
```

B-1.4 process deformed image Function

```
function [deformed_image,deformed_cell_boundaries] =
process_deformed_image(path_dir,image_name,image_h,image_w,filter_size)
    deformed_image = imread(fullfile(path_dir,image_name));

    % Filter out small objects (noise)
    deformed_image = bwareafilt(imbinarize(deformed_image),[filter_size
image_h*image_w]);

    % Get Cell Objects from First Image
    deformed_cell_boundaries = bwboundaries(deformed_image,'noholes');
end
```

B-1.5 get cell dists Function

```
function [dists,idx] = get_cell_dists(original_cell_boundaries,
deformed_cell_boundaries, original_cell_mask_props, dist_thresh, image_h,
image_w)
    for k = 1:length(deformed_cell_boundaries)
        deformed_cell_boundary = deformed_cell_boundaries{k};
        deformed_cell_x = deformed_cell_boundary(:,1);
        deformed_cell_y = deformed_cell_boundary(:,2);
        deformed_cell_mask =
poly2mask(deformed_cell_y,deformed_cell_x,image_h,image_w);
        deformed_cell_mask_props =
regionprops(deformed_cell_mask,'Centroid','BoundingBox','Area');
        if length(deformed_cell_mask_props) > 1
            [~, idx] = max([deformed_cell_mask_props.Area]);
            deformed_cell_mask_props = deformed_cell_mask_props(idx);
        end
        dists(k) = sqrt((deformed_cell_mask_props.Centroid(1)-
original_cell_mask_props.Centroid(1))^2+((deformed_cell_mask_props.Centroid(2
)-original_cell_mask_props.Centroid(2))^2));
    end
    idx = find(dists < dist_thresh);
    l1 = length(original_cell_boundaries);
    l2 = length(deformed_cell_boundaries);
    min_l = min([l1,l2]);
    idx = idx(idx<min_l);
end
```

Appendix C

Aim 3 Python Scripts

C-1.1 African PREDICT MHT Classification Algorithm

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,
cross_val_score, StratifiedKFold
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import RFECV
from sklearn.svm import SVC
from sklearn.ensemble import
RandomForestClassifier,StackingClassifier,
GradientBoostingRegressor, GradientBoostingClassifier
from BorutaShap import BorutaShap
from sklearn.linear_model import LassoCV
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.neural_network import MLPClassifier
from skopt import BayesSearchCV
from sklearn.metrics import roc_auc_score,
confusion_matrix, auc, roc_curve, precision_recall_curve
import shap
from sklearn.inspection import plot_partial_dependence

# Set User Defined Parameters
SEED = 42
cv_metric = StratifiedKFold(10, random_state=SEED,
shuffle=True)
scoring_metric = 'roc_auc'
split_ratio = 0.2
label = 'abpm_overall_ht'

# Load Data
file_path = 'C:/Users/brendym/Desktop/African PREDICT
dataset/PhD Files/'
file_name = 'Richardson_Oct 2021 Unprotected.xlsx'
df = pd.read_excel(file_path+file_name)
df_cleaned = df.copy()
```

```
sbp =
pd.DataFrame((df_cleaned['l_sbp_1']+df_cleaned['l_sbp_2']

+df_cleaned['r_sbp_1']+df_cleaned['r_sbp_2']

+df_cleaned['sphygmocor_sbp1']+df_cleaned['sphygmocor_sbp
2']

+df_cleaned['csbp_1']+df_cleaned['csbp_2']

+df_cleaned['l_osbp']+df_cleaned['r_osbp'])/10,
columns=['sbp'])
df_cleaned = pd.concat([df_cleaned,sbp],axis = 1)
df_cleaned =
df_cleaned.drop(['l_sbp_1','l_sbp_2','r_sbp_1','r_sbp_2',
'sphygmocor_sbp1',

'sphygmocor_sbp2','csbp_1','csbp_2','l_osbp','r_osbp'],ax
is=1)

dbp =
pd.DataFrame((df_cleaned['l_dbp_1']+df_cleaned['l_dbp_2']

+df_cleaned['r_dbp_1']+df_cleaned['r_dbp_2']

+df_cleaned['sphygmocor_dbp1']+df_cleaned['sphygmocor_dbp
2']

+df_cleaned['cdbp_1']+df_cleaned['cdbp_2']

+df_cleaned['l_odbp']+df_cleaned['r_odbp'])/10,
columns=['dbp'])
df_cleaned = pd.concat([df_cleaned,dbp],axis = 1)
df_cleaned =
df_cleaned.drop(['l_dbp_1','l_dbp_2','r_dbp_1','r_dbp_2',
'sphygmocor_dbp1',

'sphygmocor_dbp2','cdbp_1','cdbp_2','l_odbp','r_odbp'],ax
is=1)

# Create average hr metric
```

```
hr =
pd.DataFrame((df_cleaned['l_hr_1']+df_cleaned['l_hr_2']

+df_cleaned['r_hr_1']+df_cleaned['r_hr_2']

+df_cleaned['sphyg_hr1']+df_cleaned['sphyg_hr2'])/6,
columns=['hr'])
df_cleaned = pd.concat([df_cleaned,hr],axis = 1)
df_cleaned =
df_cleaned.drop(['l_hr_1','l_hr_2','r_hr_1','r_hr_2','sph
yg_hr1','sphyg_hr2'],axis=1)

# Create average cpp metric
cpp =
pd.DataFrame((df_cleaned['cpp1']+df_cleaned['cpp2'])/2,
columns=['cpp'])
df_cleaned = pd.concat([df_cleaned,cpp],axis = 1)
df_cleaned = df_cleaned.drop(['cpp1','cpp2'],axis=1)

# Create average map metric
map = df_cleaned['clinic_map'].rename('map')
df_cleaned = pd.concat([df_cleaned,map],axis = 1)
df_cleaned = df_cleaned.drop(['clinic_map'],axis=1)

# Create average cmp metric
cmp =
pd.DataFrame((df_cleaned['cmp_1']+df_cleaned['cmp_2'])/2,
columns=['cmp'])
df_cleaned = pd.concat([df_cleaned,cmp],axis = 1)
df_cleaned = df_cleaned.drop(['cmp_1','cmp_2'],axis=1)

# Create average pp metric
pp =
pd.DataFrame((df_cleaned['pulse_pressure']+df_cleaned['l_
pp']+df_cleaned['r_pp']/3),columns=['pp'])
df_cleaned = pd.concat([df_cleaned,pp],axis = 1)
df_cleaned =
df_cleaned.drop(['pulse_pressure','l_pp','r_pp'],axis=1)

# Create average ap metric
```

```
ap =
pd.DataFrame((df_cleaned['ap_1']+df_cleaned['ap_2'])/2,
columns=['ap'])
df_cleaned = pd.concat([df_cleaned,ap],axis = 1)
df_cleaned = df_cleaned.drop(['ap_1','ap_2'],axis=1)

# Create average cai metric
cai =
pd.DataFrame((df_cleaned['cai_1']+df_cleaned['cai_2'])/2,
columns=['cai'])
df_cleaned = pd.concat([df_cleaned,cai],axis = 1)
df_cleaned = df_cleaned.drop(['cai_1','cai_2'],axis=1)

# Create average imtn_mean metric
imtn_mean =
pd.DataFrame((df_cleaned['l_imtn_mean']+df_cleaned['r_imt
n_mean'])/2, columns=['imtn_mean'])
df_cleaned = pd.concat([df_cleaned,imtn_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_mean','r_imtn_mean'],axis=1)

# Create average imtn_min metric
imtn_min =
pd.DataFrame((df_cleaned['l_imtn_min']+df_cleaned['r_imtn
_min'])/2, columns=['imtn_min'])
df_cleaned = pd.concat([df_cleaned,imtn_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_min','r_imtn_min'],axis=1)

# Create average imtn_max metric
imtn_max =
pd.DataFrame((df_cleaned['l_imtn_max']+df_cleaned['r_imtn
_max'])/2, columns=['imtn_max'])
df_cleaned = pd.concat([df_cleaned,imtn_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_max','r_imtn_max'],axis=1)

# Create average imtn_std metric
imtn_std =
pd.DataFrame((df_cleaned['l_imtn_std']+df_cleaned['r_imtn
_std'])/2, columns=['imtn_std'])
df_cleaned = pd.concat([df_cleaned,imtn_std],axis = 1)
```

```
df_cleaned =
df_cleaned.drop(['l_imtn_std','r_imtn_std'],axis=1)

# Create average imtn_length metric
imtn_length =
pd.DataFrame((df_cleaned['l_imtn_length']+df_cleaned['r_i
mtn_length'])/2, columns=['imtn_length'])
df_cleaned = pd.concat([df_cleaned,imtn_length],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_length','r_imtn_length'],axis=1)

# Create average imtf_mean metric
imtf_mean =
pd.DataFrame((df_cleaned['l_imtf_mean']+df_cleaned['r_imt
f_mean'])/2, columns=['imtf_mean'])
df_cleaned = pd.concat([df_cleaned,imtf_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_mean','r_imtf_mean'],axis=1)

# Create average imtf_min metric
imtf_min =
pd.DataFrame((df_cleaned['l_imtf_min']+df_cleaned['r_imtf
_min'])/2, columns=['imtf_min'])
df_cleaned = pd.concat([df_cleaned,imtf_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_min','r_imtf_min'],axis=1)

# Create average imtf_max metric
imtf_max =
pd.DataFrame((df_cleaned['l_imtf_max']+df_cleaned['r_imtf
_max'])/2, columns=['imtf_max'])
df_cleaned = pd.concat([df_cleaned,imtf_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_max','r_imtf_max'],axis=1)

# Create average imtf_std metric
imtf_std =
pd.DataFrame((df_cleaned['l_imtf_std']+df_cleaned['r_imtf
_std'])/2, columns=['imtf_std'])
df_cleaned = pd.concat([df_cleaned,imtf_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_std','r_imtf_std'],axis=1)
```

```
# Create average imtf_length metric
imtf_length =
pd.DataFrame((df_cleaned['l_imtf_length']+df_cleaned['r_i
mtf_length'])/2, columns=['imtf_length'])
df_cleaned = pd.concat([df_cleaned,imtf_length],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_length','r_imtf_length'],axis=1)

# Create average ld_mean metric
ld_mean =
pd.DataFrame((df_cleaned['l_ld_mean']+df_cleaned['r_ld_me
an'])/2, columns=['ld_mean'])
df_cleaned = pd.concat([df_cleaned,ld_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_mean','r_ld_mean'],axis=1)

# Create average ld_min metric
ld_min =
pd.DataFrame((df_cleaned['l_ld_min']+df_cleaned['r_ld_min
'])/2, columns=['ld_min'])
df_cleaned = pd.concat([df_cleaned,ld_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_min','r_ld_min'],axis=1)

# Create average ld_max metric
ld_max =
pd.DataFrame((df_cleaned['l_ld_max']+df_cleaned['r_ld_max
'])/2, columns=['ld_max'])
df_cleaned = pd.concat([df_cleaned,ld_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_max','r_ld_max'],axis=1)

# Create average ld_std metric
ld_std =
pd.DataFrame((df_cleaned['l_ld_std']+df_cleaned['r_ld_std
'])/2, columns=['ld_std'])
df_cleaned = pd.concat([df_cleaned,ld_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_std','r_ld_std'],axis=1)

# Create average ld_legth metric
```

```
ld_legth =
pd.DataFrame((df_cleaned['l_ld_legth']+df_cleaned['r_ld_l
egth'])/2, columns=['ld_legth'])
df_cleaned = pd.concat([df_cleaned,ld_legth],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_legth','r_ld_legth'],axis=1)

# Create average ad_mean metric
ad_mean =
pd.DataFrame((df_cleaned['l_ad_mean']+df_cleaned['r_ad_me
an'])/2, columns=['ad_mean'])
df_cleaned = pd.concat([df_cleaned,ad_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_mean','r_ad_mean'],axis=1)

# Create average ad_min metric
ad_min =
pd.DataFrame((df_cleaned['l_ad_min']+df_cleaned['r_ad_min
'])/2, columns=['ad_min'])
df_cleaned = pd.concat([df_cleaned,ad_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_min','r_ad_min'],axis=1)

# Create average ad_max metric
ad_max =
pd.DataFrame((df_cleaned['l_ad_max']+df_cleaned['r_ad_max
'])/2, columns=['ad_max'])
df_cleaned = pd.concat([df_cleaned,ad_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_max','r_ad_max'],axis=1)

# Create average ad_std metric
ad_std =
pd.DataFrame((df_cleaned['l_ad_std']+df_cleaned['r_ad_std
'])/2, columns=['ad_std'])
df_cleaned = pd.concat([df_cleaned,ad_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_std','r_ad_std'],axis=1)

# Create average ad_legth metric
```

```
ad_length =
pd.DataFrame((df_cleaned['l_ad_length']+df_cleaned['r_ad_
length'])/2, columns=['ad_length'])
df_cleaned = pd.concat([df_cleaned,ad_length],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_length','r_ad_length'],axis=1)

# Create average rough_near metric
rough_near =
pd.DataFrame((df_cleaned['l_rough_near']+df_cleaned['r_ro
ugh_near'])/2, columns=['rough_near'])
df_cleaned = pd.concat([df_cleaned,rough_near],axis = 1)
df_cleaned =
df_cleaned.drop(['l_rough_near','r_rough_near'],axis=1)

# Create average rough_far metric
rough_far =
pd.DataFrame((df_cleaned['l_rough_far']+df_cleaned['r_rou
gh_far'])/2, columns=['rough_far'])
df_cleaned = pd.concat([df_cleaned,rough_far],axis = 1)
df_cleaned =
df_cleaned.drop(['l_rough_far','r_rough_far'],axis=1)

# Create average cca_ps metric
cca_ps =
pd.DataFrame((df_cleaned['l_cca_ps']+df_cleaned['r_cca_ps
'])/2, columns=['cca_ps'])
df_cleaned = pd.concat([df_cleaned,cca_ps],axis = 1)
df_cleaned =
df_cleaned.drop(['l_cca_ps','r_cca_ps'],axis=1)

# Create average cca_ed metric
cca_ed =
pd.DataFrame((df_cleaned['l_cca_ed']+df_cleaned['r_cca_ed
'])/2, columns=['cca_ed'])
df_cleaned = pd.concat([df_cleaned,cca_ed],axis = 1)
df_cleaned =
df_cleaned.drop(['l_cca_ed','r_cca_ed'],axis=1)

# Create average ica_ps metric
```

```
ica_ps =
pd.DataFrame((df_cleaned['l_ica_ps']+df_cleaned['r_ica_ps
'])/2, columns=['ica_ps'])
df_cleaned = pd.concat([df_cleaned,ica_ps],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ica_ps','r_ica_ps'],axis=1)

# Create average ica_ed metric
ica_ed =
pd.DataFrame((df_cleaned['l_ica_ed']+df_cleaned['r_ica_ed
'])/2, columns=['ica_ed'])
df_cleaned = pd.concat([df_cleaned,ica_ed],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ica_ed','r_ica_ed'],axis=1)

# Create average cswa metric
cswa =
pd.DataFrame((df_cleaned['lcswa']+df_cleaned['rcswa'])/2,
columns=['cswa'])
df_cleaned = pd.concat([df_cleaned,cswa],axis = 1)
df_cleaned = df_cleaned.drop(['lcswa','rcswa'],axis=1)

# Create average acimt metric
acimt =
pd.DataFrame((df_cleaned['l_acimt']+df_cleaned['r_acimt']
)/2, columns=['acimt'])
df_cleaned = pd.concat([df_cleaned,acimt],axis = 1)
df_cleaned =
df_cleaned.drop(['l_acimt','r_acimt'],axis=1)

# Create average delta_ld metric
delta_ld =
pd.DataFrame((df_cleaned['l_delta_ld']+df_cleaned['r_delt
a_ld'])/2, columns=['delta_ld'])
df_cleaned = pd.concat([df_cleaned,delta_ld],axis = 1)
df_cleaned =
df_cleaned.drop(['l_delta_ld','r_delta_ld'],axis=1)

# Create average strain metric
strain =
pd.DataFrame((df_cleaned['l_strain']+df_cleaned['r_strain
'])/2, columns=['strain'])
```

```
df_cleaned = pd.concat([df_cleaned,strain],axis = 1)
df_cleaned =
df_cleaned.drop(['l_strain','r_strain'],axis=1)

# Create average distensibility metric
distensibility =
pd.DataFrame((df_cleaned['l_distensibility']+df_cleaned['
r_distensibility'])/2, columns=['distensibility'])
df_cleaned = pd.concat([df_cleaned,distensibility],axis =
1)
df_cleaned =
df_cleaned.drop(['l_distensibility','r_distensibility'],a
xis=1)

# Create average compliance metric
compliance =
pd.DataFrame((df_cleaned['l_compliance']+df_cleaned['r_co
mpliance'])/2, columns=['compliance'])
df_cleaned = pd.concat([df_cleaned,compliance],axis = 1)
df_cleaned =
df_cleaned.drop(['l_compliance','r_compliance'],axis=1)

# Create average pem metric
bsi =
pd.DataFrame((df_cleaned['l_bsi']+df_cleaned['r_bsi'])/2,
columns=['bsi'])
df_cleaned = pd.concat([df_cleaned,bsi],axis = 1)
df_cleaned = df_cleaned.drop(['l_bsi','r_bsi'],axis=1)

# Create average pem metric
pem =
pd.DataFrame((df_cleaned['l_pem']+df_cleaned['r_pem'])/2,
columns=['pem'])
df_cleaned = pd.concat([df_cleaned,pem],axis = 1)
df_cleaned = df_cleaned.drop(['l_pem','r_pem'],axis=1)

# Create average pem metric
yem =
pd.DataFrame((df_cleaned['l_yem']+df_cleaned['r_yem'])/2,
columns=['yem'])
df_cleaned = pd.concat([df_cleaned,yem],axis = 1)
df_cleaned = df_cleaned.drop(['l_yem','r_yem'],axis=1)
```

```
# Clean patient data
df_cleaned['diabetes'] =
df_cleaned['diabetes'].replace(2, np.nan)
df_cleaned['cholesterol'] =
df_cleaned['cholesterol'].replace(2, np.nan)
df_cleaned['heart_disease'] =
df_cleaned['heart_disease'].replace(2, np.nan)
df_cleaned['mother_hypertension'] =
df_cleaned['mother_hypertension'].replace(2, np.nan)
df_cleaned['father_hypertension'] =
df_cleaned['father_hypertension'].replace(2, np.nan)
df_cleaned['mother_cholesterol'] =
df_cleaned['mother_cholesterol'].replace(2, np.nan)
df_cleaned['father_cholestrol'] =
df_cleaned['father_cholestrol'].replace(2, np.nan)
df['mother_heart_disease'] =
df['mother_heart_disease'].replace(1, np.nan)
df['mother_heart_disease'] =
df['mother_heart_disease'].replace([2,3,4], 1)
df['father_heart_disease'] =
df['father_heart_disease'].replace(1, np.nan)
df['father_heart_disease'] =
df['father_heart_disease'].replace([2,3,4], 1)
df['mother_stroke'] = df['mother_stroke'].replace(1,
np.nan)
df['mother_stroke'] =
df['mother_stroke'].replace([2,3,4], 1)
df['father_stroke'] = df['father_stroke'].replace(1,
np.nan)
df['father_stroke'] =
df['father_stroke'].replace([2,3,4], 1)
df['mother_diabetes'] = df['mother_diabetes'].replace(1,
np.nan)
df['mother_diabetes'] =
df['mother_diabetes'].replace([2,3,4], 1)
df['father_diabetes'] = df['father_diabetes'].replace(1,
np.nan)
df['father_diabetes'] =
df['father_diabetes'].replace([2,3,4], 1)

# Create prehypertensive metric
```

```
df_cleaned['prehypertensive'] = np.nan
df_cleaned.loc[(df_cleaned['sbp'] <= 119) &
(df_cleaned['dbp'] <= 79), 'prehypertensive'] = 0
df_cleaned.loc[(df_cleaned['sbp'] < 140) &
(df_cleaned['sbp'] > 119) | (df_cleaned['dbp'] < 90) &
(df_cleaned['dbp'] > 79), 'prehypertensive'] = 1

# Create obese metric
df_cleaned['obese'] = np.nan
df_cleaned.loc[(df_cleaned['bmi'] <= 25), 'obese'] = 0
df_cleaned.loc[(df_cleaned['bmi'] > 25), 'obese'] = 1

# Create left ventrical hypertrophy metric
df_cleaned['lvh'] = np.nan
df_cleaned.loc[((df_cleaned['ilvm_cube'] <= 115) &
(df_cleaned['sex'] == 1)) | ((df_cleaned['ilvm_cube'] <=
95) & (df_cleaned['sex'] == 0)), 'lvh'] = 0
df_cleaned.loc[((df_cleaned['ilvm_cube'] > 115) &
(df_cleaned['sex'] == 1)) | ((df_cleaned['ilvm_cube'] >
95) & (df_cleaned['sex'] == 0)), 'lvh'] = 1

# Create sedentary metric
df_cleaned['sedentary'] = np.nan
df_cleaned.loc[df_cleaned['vig_work_met_min'] +
df_cleaned['moderate_work_met_minutes'] +
df_cleaned['travel_met_minutes'] +
df_cleaned['vig_exercise_met'] +
df_cleaned['moderate_exercise_met'] >= 600,'sedentary'] =
0
df_cleaned.loc[df_cleaned['vig_work_met_min'] +
df_cleaned['moderate_work_met_minutes'] +
df_cleaned['travel_met_minutes'] +
df_cleaned['vig_exercise_met'] +
df_cleaned['moderate_exercise_met'] < 600,'sedentary'] =
1

# Create smoker metric
df_cleaned['smoker'] = np.nan
df_cleaned.loc[(df_cleaned['cotinine'] < 11) |
(df_cleaned['smoke'] == 0),'smoker'] = 0
df_cleaned.loc[(df_cleaned['cotinine'] >= 11) &
(df_cleaned['smoke'] == 1),'smoker'] = 1
```

```
# Create excessive alcohol metric
df_cleaned['excessive_alcohol'] = np.nan
df_cleaned.loc[(df_cleaned['ggt'] < 49) |
(df_cleaned['alcohol'] == 0),'excessive_alcohol'] = 0
df_cleaned.loc[(df_cleaned['ggt'] >= 49) &
(df_cleaned['alcohol'] == 1),'excessive_alcohol'] = 1

# Create dyslipidemic metric
df_cleaned['dyslipidemic'] = np.nan
df_cleaned.loc[df_cleaned['ldl'] < 3.5,'dyslipidemic'] =
0
df_cleaned.loc[df_cleaned['ldl'] >= 3.5,'dyslipidemic'] =
1

# Select Features
numeric_features =
['age_b','bh','bw','bmi','bsa','rfm','wc','hc','waist_hip
_ratio','nc','sbp','dbp',

'rmr','aee','dit','tee','pal','activity','vig_work_minute
s','minutes_moderate_work_week','travel_minutes_week',

'minutes_vig_exerc','minutes_moderate_exerc','sedentary_m
inutes','vig_act_day','mod_act_day','mvpa','sedentary_mvp
a',

'hr','cpp','map','cmp','pp','ap','cai','imtn_mean','imtn_
min','imtn_max','imtn_std','imtn_length','imtf_mean',

'imtf_min','imtf_max','imtf_std','imtf_length','ld_mean',
'ld_min','ld_max','ld_std','ld_legth','ad_mean','ad_min',

'ad_max','ad_std','ad_length','rough_near','rough_far','c
ca_ps','cca_ed','ica_ps','ica_ed','cswa','acimt','delta_l
d',

'strain','distensibility','compliance','bsi','pem','yem',
'ivsd','ivss','lvidd','lvids','lvpwd','lvpws','rwt','edv'
,'esv',

'ef','sv','fs','lvm_cube','ilvm_cube','lvd_mass','lvd_mas
```

```
s_ase','lvs_mass_ase','lvs_mass','lad','aod','la_ao','mv_
e_vel',

'mv_dect','mv_dec_slope','mv_a_vel','mv_ea_ratio','m_e_fi
ll','e_e_prime','mv_vmax','mv_vmean','mv_maxpg','mv_meanp
g','mv_vti',

'av_vmax','av_vmean','av_maxpg','av_meanpg','av_env_ti','
av_vti','fractalkine','gm_csf','ifn_gamma','il_1beta','il
_2','il_4','il_5',

'il_6m','il_7','il_8','il_10','il_12','il_13','il_17a','i
l_21','il_23','itac','mip_1_alpha','mip_1_beta','mip_3_al
pha','tnf_alpha_m',

'ang_ii_pmol','ang_1_7_pmol','ang_1_9_pmol','ang_1_10_pmo
l','ang_iii_pmol','ang_3_7_pmol','ang_1_5_pmol','ang_iv_p
mol','ang_2_7_pmol',

'ang_2_10_pmol','ras_aldosterone_pmol','ras_ace','ras_ren
in','aa2_ratio','ggt','chol','hdl','ldl','trig','glu','cr
p','cotinine']

categoric_features =
['sex','ethnicity','s_meds_a','s_meds_b','s_meds_d','s_me
ds_g','s_meds_h','s_meds_j','s_meds_l','s_meds_m',

's_meds_n','s_meds_p','s_meds_r','s_meds_s','s_meds_v','s
_meds_c','s_meds_c_cardiac','s_meds_c_ht','s_meds_c_ht',

's_meds_c_diur','s_meds_c_bb','s_meds_c_bb','s_meds_c_ccb
','s_meds_c_ren','s_meds_c_lipid','s_meds_c_other',

'smoke_occasion','smoke_type___1','smoke_type___2','smoke
_type___3','smoke_type___4','smoke_type___5','smoke_type_
__6',

'smoke_type___7','smoke_type___8','smoke_type___9','alcoh
ol','type_alcohol___1','type_alcohol___2','type_alcohol__
_3',

'type_alcohol___4','type_alcohol___5','type_alcohol___6',
```

```
'type_alcohol___7','type_alcohol___8','strenuous_exercise
',

'moderate_exercise','mild_exercise','leisure_activity_tim
es','heart_attack','stroke','cancer','tuberculosis',

'diabetes','cholesterol','heart_disease','mother_hyperten
sion','father_hypertension','mother_cholesterol','father_
cholestrol',

'mother_heart_disease','father_heart_disease','mother_str
oke','father_stroke','mother_diabetes','father_diabetes',

'prehypertensive','obese','lvh','sedentary','smoker','exc
essive_alcohol','dyslipidemic','clinic_bp_status']

# Create Dataset
dataset = df_cleaned.copy()
dataset =
dataset[numeric_features+categoric_features+[label]]

# Drop Rows Missing Label Values
dataset.dropna(subset = 'abpm_overall_ht', inplace =
True)

# Drop Rows with Hypertensive Patients
dataset = dataset[dataset['clinic_bp_status'] != 1]

# Remove Columns Missing More Than 10% of Data
missing_percent_columns = (dataset.isnull().sum() /
len(dataset)) * 100
columns_to_drop =
missing_percent_columns[missing_percent_columns >
10].index.tolist()
dataset = dataset.drop(columns=columns_to_drop)

# Remove Rows Missing More Than 10% of Data
missing_percent_rows = (dataset.isnull().sum(axis=1) /
len(dataset.columns)) * 100
rows_to_drop = missing_percent_rows[missing_percent_rows
> 10].index.tolist()
dataset = dataset.drop(index=rows_to_drop)
```

```
# Remove Columns With Only 1 Value
unique_counts = dataset.nunique()
unique_columns = unique_counts[unique_counts ==
1].index.tolist()
dataset = dataset.drop(columns=unique_columns)

columns_to_drop = columns_to_drop + unique_columns
# Remove Column Names from Numeric and Categorical
Feature Lists
for column in columns_to_drop:
    if column in numeric_features:
        numeric_features.remove(column)
    elif column in categoric_features:
        categoric_features.remove(column)

# Split into Features and Labels
X = dataset.drop('abpm_overall_ht',axis=1)
y = dataset['abpm_overall_ht']

# Split into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = split_ratio, random_state = SEED, stratify =
y)

### Preprocess Features
print('Preprocessing Features')
X_train_numeric = X_train[numeric_features]
X_train_categoric = X_train[categoric_features]
X_test_numeric = X_test[numeric_features]
X_test_categoric = X_test[categoric_features]

# Impute Missing Values
def mice_imputation_numeric(train_numeric, test_numeric):
    iter_imp_numeric =
IterativeImputer(GradientBoostingRegressor(random_state=S
EED))
    imputed_train =
iter_imp_numeric.fit_transform(train_numeric)
    imputed_test =
iter_imp_numeric.transform(test_numeric)
```

```
    train_numeric_imp = pd.DataFrame(imputed_train,
columns = train_numeric.columns, index =
train_numeric.index)
    test_numeric_imp = pd.DataFrame(imputed_test, columns
= test_numeric.columns, index = test_numeric.index)
    return train_numeric_imp, test_numeric_imp

def mice_imputation_categoric(train_categoric,
test_categoric):
    iter_imp_categoric =
IterativeImputer(GradientBoostingClassifier(random_state=
SEED), max_iter = 5, initial_strategy='most_frequent')
    imputed_train =
iter_imp_categoric.fit_transform(train_categoric)
    imputed_test =
iter_imp_categoric.transform(test_categoric)
    train_categoric_imp = pd.DataFrame(imputed_train,
columns = train_categoric.columns, index =
train_categoric.index).astype(int)
    test_categoric_imp = pd.DataFrame(imputed_test,
columns = test_categoric.columns, index =
test_categoric.index).astype(int)
    return train_categoric_imp, test_categoric_imp

print('Performing MICE Imputation')
X_train_numeric_imp, X_test_numeric_imp =
mice_imputation_numeric(X_train_numeric,X_test_numeric)
X_train_categoric_imp, X_test_categoric_imp =
mice_imputation_categoric(X_train_categoric,X_test_catego
ric)

X_train_imp = pd.concat([X_train_numeric_imp,
X_train_categoric_imp], axis = 1)
X_test_imp = pd.concat([X_test_numeric_imp,
X_test_categoric_imp], axis = 1)

# Scale Numerical Features
print('Performing MinMax Scaling')

X_train_scaled = X_train_imp.copy()
X_test_scaled = X_test_imp.copy()
```

```
scaler = MinMaxScaler()
X_train_scaled[numeric_features] =
scaler.fit_transform(X_train_scaled[numeric_features])
X_test_scaled[numeric_features] =
scaler.transform(X_test_scaled[numeric_features])

# Perform No Feature Selection
X_train_final = X_train_scaled.copy()
X_test_final = X_test_scaled.copy()
y_train_final = y_train.copy()
y_test_final = y_test.copy()

lr_model = LogisticRegression(random_state=SEED)
params = {
        'C': [0.1, 1, 10],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear','saga'],
        'class_weight':[None,'balanced']
    }

lr_search = BayesSearchCV(lr_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
lr_search.fit(X_train_final, y_train_final)
no_fs_best_lr_model = lr_search.best_estimator_

lr_model_scores = cross_val_score(no_fs_best_lr_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Base LR Model Score: %f (%f)" %
(lr_model_scores.mean(), lr_model_scores.std())
print(msg)

lr_model_scores = cross_val_score(no_fs_best_lr_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Base LR Model Score: %f (%f)" %
(lr_model_scores.mean(), lr_model_scores.std())
print(msg)

rf_model = RandomForestClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
```

```
        'max_depth': [None, 5, 10, 15],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto','sqrt','log2'],
        'criterion': ['gini', 'entropy'],
        'class_weight':[None,'balanced'],
        }

rf_search = BayesSearchCV(rf_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
rf_search.fit(X_train_final, y_train_final)
no_fs_best_rf_model = rf_search.best_estimator_

rf_model_scores = cross_val_score(no_fs_best_rf_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Base RF Model Score: %f (%f)" %
(rf_model_scores.mean(), rf_model_scores.std())
print(msg)

rf_model_scores = cross_val_score(no_fs_best_rf_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Base RF Model Score: %f (%f)" %
(rf_model_scores.mean(), rf_model_scores.std())
print(msg)

xgb_model = XGBClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.1, 0.05, 0.01],
        'max_depth': [3, 5, 10],
        'subsample': [0.5, 1.0, 'uniform'],
        'gamma':[0, 5.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
        }

xgb_search = BayesSearchCV(xgb_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
xgb_search.fit(X_train_final, y_train_final)
no_fs_best_xgb_model = xgb_search.best_estimator_
```

```
xgb_model_scores = cross_val_score(no_fs_best_xgb_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Base XGB Model Score: %f (%f)" %
(xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

xgb_model_scores = cross_val_score(no_fs_best_xgb_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Base XGB Model Score: %f (%f)" %
(xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

mlpc_model =
MLPClassifier(random_state=SEED,max_iter=2000)
params = {
        #'hidden_layer_sizes':
[(50,),(100,),(50,50),(100,50,25)],
        'activation':['relu','tanh'],
        'solver':['sgd','adam'],
        'learning_rate':['constant','adaptive'],
        'alpha':[0.0001, 0.001, 0.01]
        }

mlpc_search = BayesSearchCV(mlpc_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
mlpc_search.fit(X_train_final, y_train_final)
no_fs_best_mlpc_model = mlpc_search.best_estimator_

mlpc_model_scores =
cross_val_score(no_fs_best_mlpc_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - Base ANN Model Score: %f (%f)" %
(mlpc_model_scores.mean(), mlpc_model_scores.std())
print(msg)

mlpc_model_scores =
cross_val_score(no_fs_best_mlpc_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - Base ANN Model Score: %f (%f)" %
(mlpc_model_scores.mean(), mlpc_model_scores.std())
```

```
print(msg)

base_learners = [
        ('LR', no_fs_best_lr_model),
        ('RF', no_fs_best_rf_model),
        ('ANN', no_fs_best_mlpc_model),
        ('XGB', no_fs_best_xgb_model),
        ]

no_fs_stk_model =
StackingClassifier(estimators=base_learners,
final_estimator = lr_model, cv = cv_metric)
no_fs_stk_model.fit(X_train_final, y_train_final)

stk_model_scores = cross_val_score(no_fs_stk_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Base Stacking Classifier Model
Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

stk_model_scores = cross_val_score(no_fs_stk_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Base Stacking Classifier Model Score:
%f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

# Perform Manual Feature Selection
manually_selected_features =
['sex','ethnicity','age_b','bmi','waist_hip_ratio','sbp',
'dbp',

'map','pp','hr','glu','trig','ldl','hdl','chol','cotinine
',

'ggt','smoker','excessive_alcohol','sedentary','prehypert
ensive',

'obese','stroke','diabetes','lvm_cube','lvh','imtn_mean']
```

```
X_train_final = X_train_scaled.copy()
X_test_final = X_test_scaled.copy()
y_train_final = y_train.copy()
y_test_final = y_test.copy()

X_train_final = X_train_final[manually_selected_features]
X_test_final = X_test_final[manually_selected_features]
y_train_final = y_train_final
y_test_final = y_test_final

lr_model = LogisticRegression(random_state=SEED)
params = {
        'C': [0.1, 1, 10],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear','saga'],
        'class_weight':[None,'balanced']
    }

lr_search = BayesSearchCV(lr_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
lr_search.fit(X_train_final, y_train_final)
manual_best_lr_model = lr_search.best_estimator_

lr_model_scores = cross_val_score(manual_best_lr_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Manually Tunned LR Model Score: %f
(%f)" % (lr_model_scores.mean(), lr_model_scores.std())
print(msg)

lr_model_scores = cross_val_score(manual_best_lr_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Manually Tunned LR Model Score: %f
(%f)" % (lr_model_scores.mean(), lr_model_scores.std())
print(msg)

rf_model = RandomForestClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 5, 10, 15],
```

```
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto','sqrt','log2'],
        'criterion': ['gini', 'entropy'],
        'class_weight':[None,'balanced'],
        }

rf_search = BayesSearchCV(rf_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
rf_search.fit(X_train_final, y_train_final)
manual_best_rf_model = rf_search.best_estimator_

rf_model_scores = cross_val_score(manual_best_rf_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Manually Tunned RF Model Score: %f
(%f)" % (rf_model_scores.mean(), rf_model_scores.std())
print(msg)

rf_model_scores = cross_val_score(manual_best_rf_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TESTING SET - Manually Tunned RF Model Score: %f
(%f)" % (rf_model_scores.mean(), rf_model_scores.std())
print(msg)

xgb_model = XGBClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.1, 0.05, 0.01],
        'max_depth': [3, 5, 10],
        'subsample': [0.5, 1.0, 'uniform'],
        'gamma':[0, 5.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
        }

xgb_search = BayesSearchCV(xgb_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
xgb_search.fit(X_train_final, y_train_final)
manual_best_xgb_model = xgb_search.best_estimator_
```

```
xgb_model_scores = cross_val_score(manual_best_xgb_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Manually Tunned XGB Model Score: %f
(%f)" % (xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

xgb_model_scores = cross_val_score(manual_best_xgb_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Manually Tunned XGB Model Score: %f
(%f)" % (xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

mlpc_model =
MLPClassifier(random_state=SEED,max_iter=2000)
params = {
        #'hidden_layer_sizes':
[(50,),(100,),(50,50),(100,50,25)],
        'activation':['relu','tanh'],
        'solver':['sgd','adam'],
        'learning_rate':['constant','adaptive'],
        'alpha':[0.0001, 0.001, 0.01]
        }

mlpc_search = BayesSearchCV(mlpc_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
mlpc_search.fit(X_train_final, y_train_final)
manual_best_mlpc_model = mlpc_search.best_estimator_

mlpc_model_scores =
cross_val_score(manual_best_mlpc_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - Manually Tunned ANN Model Score: %f
(%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

mlpc_model_scores =
cross_val_score(manual_best_mlpc_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
```

```
msg = "TEST SET - Manually Tunned ANN Model Score: %f
(%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

base_learners = [
        ('LR', manual_best_lr_model),
        ('RF', manual_best_rf_model),
        ('ANN', manual_best_mlpc_model),
        ('XGB', manual_best_xgb_model),
        ]

manual_stk_model =
StackingClassifier(estimators=base_learners,
final_estimator = lr_model, cv = cv_metric)
manual_stk_model.fit(X_train_final, y_train_final)

stk_model_scores = cross_val_score(manual_stk_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - Manually Tunned Stacking Classifier
Model Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

stk_model_scores = cross_val_score(manual_stk_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - Manually Tunned Stacking Classifier
Model Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

# Select Features using RFE-SVM
print('Performing Feature Selection with RFE-SVM')
estimator = SVC(random_state=SEED, probability=True,
kernel='linear')
selector = RFECV(estimator = estimator, cv = cv_metric,
scoring = scoring_metric)
selector.fit(X_train_scaled,y_train)
selector.support_
```

```
#print('Optimal number of features :',
selector.n_features_)
#print('Best RFECV-SVM features: ',
X.columns[selector.support_])
#print('Original features :', X.columns)
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score \n of number of
selected features")
plt.plot(range(1, len(selector.grid_scores_) + 1),
selector.grid_scores_)
plt.show()

rfe_svm_features =
X_train_scaled.columns[selector.support_]

X_train_final = X_train_scaled.copy()
X_test_final = X_test_scaled.copy()
y_train_final = y_train.copy()
y_test_final = y_test.copy()

X_train_final = X_train_final[rfe_svm_features]
X_test_final = X_test_final[rfe_svm_features]
y_train_final = y_train_final
y_test_final = y_test_final

lr_model = LogisticRegression(random_state=SEED)
params = {
        'C': [0.1, 1, 10],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear','saga'],
        'class_weight':[None,'balanced']
    }

lr_search = BayesSearchCV(lr_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
lr_search.fit(X_train_final, y_train_final)
rfe_svm_best_lr_model = lr_search.best_estimator_

lr_model_scores = cross_val_score(rfe_svm_best_lr_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
```

```
msg = "TRAINING SET - RFECV-SVM Tunned LR Model Score: %f
(%f)" % (lr_model_scores.mean(), lr_model_scores.std())
print(msg)

lr_model_scores = cross_val_score(rfe_svm_best_lr_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - RFECV-SVM Tunned LR Model Score: %f
(%f)" % (lr_model_scores.mean(), lr_model_scores.std())
print(msg)

rf_model = RandomForestClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 5, 10, 15],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto','sqrt','log2'],
        'criterion': ['gini', 'entropy'],
        'class_weight':[None,'balanced'],
        }

rf_search = BayesSearchCV(rf_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
rf_search.fit(X_train_final, y_train_final)
rfe_svm_best_rf_model = rf_search.best_estimator_

rf_model_scores = cross_val_score(rfe_svm_best_rf_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - RFECV-SVM Tunned RF Model Score: %f
(%f)" % (rf_model_scores.mean(), rf_model_scores.std())
print(msg)

rf_model_scores = cross_val_score(rfe_svm_best_rf_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - RFECV-SVM Tunned RF Model Score: %f
(%f)" % (rf_model_scores.mean(), rf_model_scores.std())
print(msg)

xgb_model = XGBClassifier(random_state=SEED)
```

```
params = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.1, 0.05, 0.01],
        'max_depth': [3, 5, 10],
        'subsample': [0.5, 1.0, 'uniform'],
        'gamma':[0, 5.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
        }

xgb_search = BayesSearchCV(xgb_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
xgb_search.fit(X_train_final, y_train_final)
rfe_svm_best_xgb_model = xgb_search.best_estimator_

xgb_model_scores =
cross_val_score(rfe_svm_best_xgb_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - RFECV-SVM Tunned XGB Model Score:
%f (%f)" % (xgb_model_scores.mean(),
xgb_model_scores.std())
print(msg)

xgb_model_scores =
cross_val_score(rfe_svm_best_xgb_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - RFECV-SVM Tunned XGB Model Score: %f
(%f)" % (xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

mlpc_model =
MLPClassifier(random_state=SEED,max_iter=2000)
params = {
        #'hidden_layer_sizes':
[(50,),(100,),(50,50),(100,50,25)],
        'activation':['relu','tanh'],
        'solver':['sgd','adam'],
        'learning_rate':['constant','adaptive'],
        'alpha':[0.0001, 0.001, 0.01]
        }

mlpc_search = BayesSearchCV(mlpc_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
```

```
mlpc_search.fit(X_train_final, y_train_final)
rfe_svm_best_mlpc_model = mlpc_search.best_estimator_

mlpc_model_scores =
cross_val_score(rfe_svm_best_mlpc_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - RFECV-SVM Tunned ANN Model Score:
%f (%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

mlpc_model_scores =
cross_val_score(rfe_svm_best_mlpc_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - RFECV-SVM Tunned ANN Model Score: %f
(%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

base_learners = [
        ('LR', rfe_svm_best_lr_model),
        ('RF', rfe_svm_best_rf_model),
        ('ANN', rfe_svm_best_mlpc_model),
        ('XGB', rfe_svm_best_xgb_model),
        ]

rfe_svm_stk_model =
StackingClassifier(estimators=base_learners,
final_estimator = lr_model, cv = cv_metric)
rfe_svm_stk_model.fit(X_train_final, y_train_final)

stk_model_scores = cross_val_score(rfe_svm_stk_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - RFECV-SVM Tunned Stacking
Classifier Model Score: %f (%f)" %
(stk_model_scores.mean(), stk_model_scores.std())
print(msg)

stk_model_scores = cross_val_score(rfe_svm_stk_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
```

```
msg = "TEST SET - RFECV-SVM Tunned Stacking Classifier
Model Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

# Select Features using BorutaSHAP
print('Performing Feature Selection with BorutaSHAP')
selector = BorutaShap(model =
RandomForestClassifier(random_state=SEED),
importance_measure = 'shap', classification = True)
selector.fit(X=X_train_scaled, y=y_train, n_trials = 100,
random_state=SEED)
selector.plot(which_features='all', figsize=(16,12))

features_to_remove = selector.features_to_remove
borutaSHAP_features = [column for column in
X_train.columns if column not in features_to_remove]
#print('Best BorutaSHAP features: ', borutaSHAP_features)

X_train_final = X_train_scaled.copy()
X_test_final = X_test_scaled.copy()
y_train_final = y_train.copy()
y_test_final = y_test.copy()

X_train_final = X_train_final[borutaSHAP_features]
X_test_final = X_test_final[borutaSHAP_features]
y_train_final = y_train_final
y_test_final = y_test_final

lr_model = LogisticRegression(random_state=SEED)
params = {
        'C': [0.1, 1, 10],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear','saga'],
        'class_weight':[None,'balanced']
    }

lr_search = BayesSearchCV(lr_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
lr_search.fit(X_train_final, y_train_final)
borutaSHAP_best_lr_model = lr_search.best_estimator_
```

```
lr_model_scores =
cross_val_score(borutaSHAP_best_lr_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - borutaSHAP Tunned LR Model Score:
%f (%f)" % (lr_model_scores.mean(),
lr_model_scores.std())
print(msg)

lr_model_scores =
cross_val_score(borutaSHAP_best_lr_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - borutaSHAP Tunned LR Model Score: %f
(%f)" % (lr_model_scores.mean(), lr_model_scores.std())
print(msg)

rf_model = RandomForestClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 5, 10, 15],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto','sqrt','log2'],
        'criterion': ['gini', 'entropy'],
        'class_weight':[None,'balanced'],
        }

rf_search = BayesSearchCV(rf_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
rf_search.fit(X_train_final, y_train_final)
borutaSHAP_best_rf_model = rf_search.best_estimator_

rf_model_scores =
cross_val_score(borutaSHAP_best_rf_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - borutaSHAP Tunned RF Model Score:
%f (%f)" % (rf_model_scores.mean(),
rf_model_scores.std())
print(msg)

rf_model_scores =
cross_val_score(borutaSHAP_best_rf_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
```

```
msg = "TEST SET - borutaSHAP Tunned RF Model Score: %f
(%f)" % (rf_model_scores.mean(), rf_model_scores.std())
print(msg)

xgb_model = XGBClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.1, 0.05, 0.01],
        'max_depth': [3, 5, 10],
        'subsample': [0.5, 1.0, 'uniform'],
        'gamma':[0, 5.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
        }

xgb_search = BayesSearchCV(xgb_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
xgb_search.fit(X_train_final, y_train_final)
borutaSHAP_best_xgb_model = xgb_search.best_estimator_

xgb_model_scores =
cross_val_score(borutaSHAP_best_xgb_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - borutaSHAP Tunned XGB Model Score:
%f (%f)" % (xgb_model_scores.mean(),
xgb_model_scores.std())
print(msg)

xgb_model_scores =
cross_val_score(borutaSHAP_best_xgb_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - borutaSHAP Tunned XGB Model Score: %f
(%f)" % (xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

mlpc_model =
MLPClassifier(random_state=SEED,max_iter=2000)
params = {
        #'hidden_layer_sizes':
[(50,),(100,),(50,50),(100,50,25)],
        'activation':['relu','tanh'],
        'solver':['sgd','adam'],
        'learning_rate':['constant','adaptive'],
```

```
        'alpha':[0.0001, 0.001, 0.01]
        }

mlpc_search = BayesSearchCV(mlpc_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
mlpc_search.fit(X_train_final, y_train_final)
borutaSHAP_best_mlpc_model = mlpc_search.best_estimator_

mlpc_model_scores =
cross_val_score(borutaSHAP_best_mlpc_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - borutaSHAP Tunned ANN Model Score:
%f (%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

mlpc_model_scores =
cross_val_score(borutaSHAP_best_mlpc_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - borutaSHAP Tunned ANN Model Score: %f
(%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

base_learners = [
        ('LR', borutaSHAP_best_lr_model),
        ('RF', borutaSHAP_best_rf_model),
        ('ANN', borutaSHAP_best_mlpc_model),
        ('XGB', borutaSHAP_best_xgb_model),
        ]

borutaSHAP_stk_model =
StackingClassifier(estimators=base_learners,
final_estimator = lr_model, cv = cv_metric)
borutaSHAP_stk_model.fit(X_train_final, y_train_final)

stk_model_scores = cross_val_score(borutaSHAP_stk_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
```

```
msg = "TRAINING SET - BorutaSHAP Tunned Stacking
Classifier Model Score: %f (%f)" %
(stk_model_scores.mean(), stk_model_scores.std())
print(msg)

stk_model_scores = cross_val_score(borutaSHAP_stk_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - BorutaSHAP Tunned Stacking Classifier
Model Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

# Select Features using LASSO
print('Performing Feature Selection with LASSO')
selector = LassoCV()
selector.fit(X_train_scaled,y_train)
coefficients = selector.coef_
importance = np.abs(coefficients)
lasso_features = np.array(X_train.columns)[importance >
0]
#print('Best LASSO features: ', lasso_features)

X_train_final = X_train_scaled.copy()
X_test_final = X_test_scaled.copy()
y_train_final = y_train.copy()
y_test_final = y_test.copy()

X_train_final = X_train_final[lasso_features]
X_test_final = X_test_final[lasso_features]
y_train_final = y_train_final
y_test_final = y_test_final

lr_model = LogisticRegression(random_state=SEED)
params = {
        'C': [0.1, 1, 10],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear','saga'],
        'class_weight':[None,'balanced']
    }
```

```
lr_search = BayesSearchCV(lr_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
lr_search.fit(X_train_final, y_train_final)
lasso_best_lr_model = lr_search.best_estimator_

lr_model_scores = cross_val_score(lasso_best_lr_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - LASSO Tunned LR Model Score: %f
(%f)" % (lr_model_scores.mean(), lr_model_scores.std())
print(msg)

lr_model_scores = cross_val_score(lasso_best_lr_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - LASSO Tunned LR Model Score: %f (%f)" %
(lr_model_scores.mean(), lr_model_scores.std())
print(msg)

rf_model = RandomForestClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 5, 10, 15],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto','sqrt','log2'],
        'criterion': ['gini', 'entropy'],
        'class_weight':[None,'balanced'],
        }

rf_search = BayesSearchCV(rf_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
rf_search.fit(X_train_final, y_train_final)
lasso_best_rf_model = rf_search.best_estimator_

rf_model_scores = cross_val_score(lasso_best_rf_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - LASSO Tunned RF Model Score: %f
(%f)" % (rf_model_scores.mean(), rf_model_scores.std())
print(msg)
```

```
rf_model_scores = cross_val_score(lasso_best_rf_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - LASSO Tunned RF Model Score: %f (%f)" %
(rf_model_scores.mean(), rf_model_scores.std())
print(msg)

xgb_model = XGBClassifier(random_state=SEED)
params = {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.1, 0.05, 0.01],
        'max_depth': [3, 5, 10],
        'subsample': [0.5, 1.0, 'uniform'],
        'gamma':[0, 5.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
        }

xgb_search = BayesSearchCV(xgb_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
xgb_search.fit(X_train_final, y_train_final)
lasso_best_xgb_model = xgb_search.best_estimator_

xgb_model_scores = cross_val_score(lasso_best_xgb_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TRAINING SET - LASSO Tunned XGB Model Score: %f
(%f)" % (xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

xgb_model_scores = cross_val_score(lasso_best_xgb_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - LASSO Tunned XGB Model Score: %f (%f)"
% (xgb_model_scores.mean(), xgb_model_scores.std())
print(msg)

mlpc_model =
MLPClassifier(random_state=SEED,max_iter=2000)
params = {
        #'hidden_layer_sizes':
[(50,),(100,),(50,50),(100,50,25)],
        'activation':['relu','tanh'],
```

```
        'solver':['sgd','adam'],
        'learning_rate':['constant','adaptive'],
        'alpha':[0.0001, 0.001, 0.01]
        }

mlpc_search = BayesSearchCV(mlpc_model, params, scoring =
scoring_metric, cv = cv_metric, random_state = SEED)
mlpc_search.fit(X_train_final, y_train_final)
lasso_best_mlpc_model = mlpc_search.best_estimator_

mlpc_model_scores =
cross_val_score(lasso_best_mlpc_model, X_train_final,
y_train_final, cv = cv_metric, scoring=scoring_metric)
msg = "TRAINING SET - LASSO Tunned ANN Model Score: %f
(%f)" % (mlpc_model_scores.mean(),
mlpc_model_scores.std())
print(msg)

mlpc_model_scores =
cross_val_score(lasso_best_mlpc_model, X_test_final,
y_test_final, cv = cv_metric, scoring=scoring_metric)
msg = "TEST SET - LASSO Tunned ANN Model Score: %f (%f)"
% (mlpc_model_scores.mean(), mlpc_model_scores.std())
print(msg)

base_learners = [
        ('LR', lasso_best_lr_model),
        ('RF', lasso_best_rf_model),
        ('ANN', lasso_best_mlpc_model),
        ('XGB', lasso_best_xgb_model),
        ]

LASSO_stk_model =
StackingClassifier(estimators=base_learners,
final_estimator = lr_model, cv = cv_metric)
LASSO_stk_model.fit(X_train_final, y_train_final)

stk_model_scores = cross_val_score(LASSO_stk_model,
X_train_final, y_train_final, cv = cv_metric,
scoring=scoring_metric)
```

```
msg = "TRAINING SET - LASSO Tunned Stacking Classifier
Model Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

stk_model_scores = cross_val_score(LASSO_stk_model,
X_test_final, y_test_final, cv = cv_metric,
scoring=scoring_metric)
msg = "TEST SET - LASSO Tunned Stacking Classifier Model
Score: %f (%f)" % (stk_model_scores.mean(),
stk_model_scores.std())
print(msg)

# Perform Threshold Tunning
# Tune Stacking Model Decision Threshold
y_probs_tune =
borutaSHAP_stk_model.predict_proba(X_train_scaled[borutaS
HAP_features].values)
y_probs_tune = y_probs_tune[:,1]

cv = StratifiedKFold(n_splits=10, shuffle=True,
random_state=SEED)
thresholds = np.arange(0.01,0.99,0.001)
thresholds_df = pd.DataFrame({'threshold':thresholds})
i = 1
for _, val_index in
cv.split(X_train_scaled[borutaSHAP_features],y_train):
    X_val_cv =
X_train_scaled[borutaSHAP_features].iloc[val_index]
    y_val_cv = y_train_final.iloc[val_index]

    y_probs_tune =
borutaSHAP_stk_model.predict_proba(X_val_cv.values)
    y_probs_tune = y_probs_tune[:,1]

    fscores = []
    for threshold in thresholds:
        y_pred = y_probs_tune > threshold
        fscore = roc_auc_score(y_val_cv, y_pred)
        fscores.append(fscore)
```

```
    thresholds_df =
pd.concat([thresholds_df,pd.DataFrame({'Fold'+str(i):fsco
res})],axis=1)
    i = i + 1
average_fscores =
pd.DataFrame({'average':thresholds_df.iloc[:,1:10+1].mean
(axis=1)})
pr_ix = np.nanargmax(average_fscores)

decision_threshold = thresholds[pr_ix]

print('Optimal decision threshold: '+
str(decision_threshold))

def binary_performances(y_true, y_prob, thresh=0.5,
labels=['Positives','Negatives'], title = ''):

    shape = y_prob.shape
    if len(shape) > 1:
        if shape[1] > 2:
            raise ValueError('A binary class problem is
required')
        else:
            y_prob = y_prob[:,1]

    plt.figure(figsize=[15,12])

    #1 -- Confusion matrix
    cm = confusion_matrix(y_true,
(y_prob>thresh).astype(int))

    plt.subplot(221)
    ax = sns.heatmap(cm, annot=True, cmap='Blues',
cbar=False,
                     annot_kws={"size": 14}, fmt='g')
    cmlabels = ['True Negatives', 'False Positives',
                'False Negatives', 'True Positives']
    for i,t in enumerate(ax.texts):
        t.set_text(t.get_text() + "\n" + cmlabels[i])
    plt.title('Confusion Matrix', size=15)
    plt.xlabel('Predicted Values', size=13)
    plt.ylabel('True Values', size=13)
```

```
    #2 -- Distributions of Predicted Probabilities of
both classes
    plt.subplot(222)
    plt.hist(y_prob[y_true==1], density=True, bins=25,
             alpha=.5, color='green',  label=labels[0])
    plt.hist(y_prob[y_true==0], density=True, bins=25,
             alpha=.5, color='red', label=labels[1])
    plt.axvline(thresh, color='blue', linestyle='--',
label='Decision Threshold')
    plt.xlim([0,1])
    plt.title('Distributions of Predictions', size=15)
    plt.xlabel('Positive Probability (predicted)',
size=13)
    plt.ylabel('Samples (normalized scale)', size=13)
    plt.legend(loc="upper right")

    #3 -- ROC curve with annotated decision point
    fp_rates, tp_rates, _ = roc_curve(y_true, y_prob)
    roc_auc = auc(fp_rates, tp_rates)
    plt.subplot(223)
    plt.plot(fp_rates, tp_rates, color='orange',
             lw=1, label='ROC curve (area = %0.3f)' %
roc_auc)
    plt.plot([0, 1], [0, 1], lw=1, linestyle='--',
color='grey')
    tn, fp, fn, tp = [i for i in cm.ravel()]
    plt.plot(fp/(fp+tn), tp/(tp+fn), 'bo', markersize=8,
label='Decision Point')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate', size=13)
    plt.ylabel('True Positive Rate', size=13)
    plt.title('ROC Curve', size=15)
    plt.legend(loc="lower right")
    plt.subplots_adjust(wspace=.3)

    #4 -- PR curve with annotated decision point
    precisions, recalls, _ =
precision_recall_curve(y_true, y_prob)
    pr_auc = auc(recalls, precisions)
    plt.subplot(224)
```

```
    plt.plot(recalls, precisions, color='orange',
             lw=1, label='PR curve (area = %0.3f)' %
pr_auc)
    no_skill_score = len(y_true[y_true==1]) / len(y_true)
    plt.plot([0, 1], [no_skill_score, no_skill_score],
lw=1, linestyle='--', color='grey')
    tn, fp, fn, tp = [i for i in cm.ravel()]
    plt.plot(tp/(tp+fn), tp/(tp+fp), 'bo', markersize=8,
label='Decision Point')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('Recall', size=13)
    plt.ylabel('Precision', size=13)
    plt.title('Precision-Recall Curve', size=15)
    plt.legend(loc="upper right")
    plt.subplots_adjust(wspace=.3)

    plt.suptitle(title)
    plt.show()

    tn, fp, fn, tp = [i for i in cm.ravel()]
    accuracy = (tp+tn)/(tp+fp+tn+fp)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
    F1 = 2*(precision * recall) / (precision + recall)

    results = {
        'Decision Threshold': thresh,
        "TN": tn, "FP": fp, "FN": fn, "TP": tp,
        "Accuracy": accuracy, "Precision": precision,
        "Recall": recall, "F1 Score": F1,
        "AUC": roc_auc, "PR AUC":pr_auc
    }

    prints = [f"{kpi}: {round(score, 3)}" for kpi,score
in results.items()]
    prints = ' | '.join(prints)
    print(prints)

    return results
```

```
y_no_model = X_test['sbp'].apply(lambda x: 1 if x > 120
else 0)
no_model_test_results = binary_performances(y_test,
y_no_model, title = 'No Model Evaluation: Testing Set')

y_probs =
borutaSHAP_stk_model.predict_proba(X_test_scaled[borutaSH
AP_features])
y_probs = y_probs[:,1]
stk_test_results = binary_performances(y_test,
y_probs,thresh = decision_threshold, title = 'Stacking
Classifier Model Evaluation: Testing Set')

### Explain Classifier Model
# SHAP Feature Importance Plot
explainer = shap.Explainer(borutaSHAP_stk_model.predict,
X_test_scaled[borutaSHAP_features].values)
shap_values =
explainer(X_test_scaled[borutaSHAP_features])
#shap_values = shap_values[:,:,1]
shap.plots.beeswarm(shap_values,max_display=X_test_scaled
[borutaSHAP_features].shape[1])

# Calculate the mean absolute SHAP values for each
feature
feature_importance =
np.abs(shap_values.values).mean(axis=0)

# Create a DataFrame to store the feature importance
scores
feature_importance_df = pd.DataFrame({
    'Feature':
X_test_scaled[borutaSHAP_features].columns,
    'Importance': feature_importance
})

# Sort the DataFrame by importance scores in descending
order
feature_importance_df =
feature_importance_df.sort_values('Importance',
ascending=False)
```

```
# Partial Dependency Plots (PDP)
for i in
range(len(X_test_scaled[borutaSHAP_features].columns)):

plot_partial_dependence(estimator=borutaSHAP_stk_model,X=
X_test_scaled[borutaSHAP_features],features=[i],feature_n
ames=X_test_scaled[borutaSHAP_features].columns)
```

C-1.2 African PREDICT MHT Data Analysis

```
# Import Libraries
import pandas as pd
import numpy as np
from scipy.stats import bartlett, kruskal, shapiro,
ttest_ind, chi2_contingency
from sklearn.model_selection import train_test_split
import sys

# Define SEED
SEED = 42

# Define Scoring Metric
scoring_metric = 'roc_auc'

# Define CV Metric
cv_metric = 7

# Define Split Ratio
split_ratio = 0.2

# Define Label
label = 'abpm_overall_ht'

# Load Data
file_path = 'C:/Users/brendym/Desktop/African PREDICT
dataset/PhD Files/'
file_name = 'Richardson_Oct 2021 Unprotected.xlsx'
df = pd.read_excel(file_path+file_name)
df_cleaned = df.copy()

sbp =
pd.DataFrame((df_cleaned['l_sbp_1']+df_cleaned['l_sbp_2']

+df_cleaned['r_sbp_1']+df_cleaned['r_sbp_2']

+df_cleaned['sphygmocor_sbp1']+df_cleaned['sphygmocor_sbp
2']

+df_cleaned['csbp_1']+df_cleaned['csbp_2']
```

```
+df_cleaned['l_osbp']+df_cleaned['r_osbp'])/10,
columns=['sbp'])
df_cleaned = pd.concat([df_cleaned,sbp],axis = 1)
df_cleaned =
df_cleaned.drop(['l_sbp_1','l_sbp_2','r_sbp_1','r_sbp_2',
'sphygmocor_sbp1',

'sphygmocor_sbp2','csbp_1','csbp_2','l_osbp','r_osbp'],ax
is=1)

dbp =
pd.DataFrame((df_cleaned['l_dbp_1']+df_cleaned['l_dbp_2']

+df_cleaned['r_dbp_1']+df_cleaned['r_dbp_2']

+df_cleaned['sphygmocor_dbp1']+df_cleaned['sphygmocor_dbp
2']

+df_cleaned['cdbp_1']+df_cleaned['cdbp_2']

+df_cleaned['l_odbp']+df_cleaned['r_odbp'])/10,
columns=['dbp'])
df_cleaned = pd.concat([df_cleaned,dbp],axis = 1)
df_cleaned =
df_cleaned.drop(['l_dbp_1','l_dbp_2','r_dbp_1','r_dbp_2',
'sphygmocor_dbp1',

'sphygmocor_dbp2','cdbp_1','cdbp_2','l_odbp','r_odbp'],ax
is=1)

# Create average hr metric
hr =
pd.DataFrame((df_cleaned['l_hr_1']+df_cleaned['l_hr_2']

+df_cleaned['r_hr_1']+df_cleaned['r_hr_2']

+df_cleaned['sphyg_hr1']+df_cleaned['sphyg_hr2'])/6,
columns=['hr'])
df_cleaned = pd.concat([df_cleaned,hr],axis = 1)
```

```
df_cleaned =
df_cleaned.drop(['l_hr_1','l_hr_2','r_hr_1','r_hr_2','sph
yg_hr1','sphyg_hr2'],axis=1)

# Create average cpp metric
cpp =
pd.DataFrame((df_cleaned['cpp1']+df_cleaned['cpp2'])/2,
columns=['cpp'])
df_cleaned = pd.concat([df_cleaned,cpp],axis = 1)
df_cleaned = df_cleaned.drop(['cpp1','cpp2'],axis=1)

# Create average map metric
map = df_cleaned['clinic_map'].rename('map')
df_cleaned = pd.concat([df_cleaned,map],axis = 1)
df_cleaned = df_cleaned.drop(['clinic_map'],axis=1)

# Create average cmp metric
cmp =
pd.DataFrame((df_cleaned['cmp_1']+df_cleaned['cmp_2'])/2,
columns=['cmp'])
df_cleaned = pd.concat([df_cleaned,cmp],axis = 1)
df_cleaned = df_cleaned.drop(['cmp_1','cmp_2'],axis=1)

# Create average pp metric
pp =
pd.DataFrame((df_cleaned['pulse_pressure']+df_cleaned['l_
pp']+df_cleaned['r_pp']/3),columns=['pp'])
df_cleaned = pd.concat([df_cleaned,pp],axis = 1)
df_cleaned =
df_cleaned.drop(['pulse_pressure','l_pp','r_pp'],axis=1)

# Create average ap metric
ap =
pd.DataFrame((df_cleaned['ap_1']+df_cleaned['ap_2'])/2,
columns=['ap'])
df_cleaned = pd.concat([df_cleaned,ap],axis = 1)
df_cleaned = df_cleaned.drop(['ap_1','ap_2'],axis=1)

# Create average cai metric
cai =
pd.DataFrame((df_cleaned['cai_1']+df_cleaned['cai_2'])/2,
columns=['cai'])
```

```
df_cleaned = pd.concat([df_cleaned,cai],axis = 1)
df_cleaned = df_cleaned.drop(['cai_1','cai_2'],axis=1)

# Create average imtn_mean metric
imtn_mean =
pd.DataFrame((df_cleaned['l_imtn_mean']+df_cleaned['r_imt
n_mean'])/2, columns=['imtn_mean'])
df_cleaned = pd.concat([df_cleaned,imtn_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_mean','r_imtn_mean'],axis=1)

# Create average imtn_min metric
imtn_min =
pd.DataFrame((df_cleaned['l_imtn_min']+df_cleaned['r_imtn
_min'])/2, columns=['imtn_min'])
df_cleaned = pd.concat([df_cleaned,imtn_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_min','r_imtn_min'],axis=1)

# Create average imtn_max metric
imtn_max =
pd.DataFrame((df_cleaned['l_imtn_max']+df_cleaned['r_imtn
_max'])/2, columns=['imtn_max'])
df_cleaned = pd.concat([df_cleaned,imtn_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_max','r_imtn_max'],axis=1)

# Create average imtn_std metric
imtn_std =
pd.DataFrame((df_cleaned['l_imtn_std']+df_cleaned['r_imtn
_std'])/2, columns=['imtn_std'])
df_cleaned = pd.concat([df_cleaned,imtn_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_std','r_imtn_std'],axis=1)

# Create average imtn_length metric
imtn_length =
pd.DataFrame((df_cleaned['l_imtn_length']+df_cleaned['r_i
mtn_length'])/2, columns=['imtn_length'])
df_cleaned = pd.concat([df_cleaned,imtn_length],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtn_length','r_imtn_length'],axis=1)
```

```
# Create average imtf_mean metric
imtf_mean =
pd.DataFrame((df_cleaned['l_imtf_mean']+df_cleaned['r_imt
f_mean'])/2, columns=['imtf_mean'])
df_cleaned = pd.concat([df_cleaned,imtf_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_mean','r_imtf_mean'],axis=1)

# Create average imtf_min metric
imtf_min =
pd.DataFrame((df_cleaned['l_imtf_min']+df_cleaned['r_imtf
_min'])/2, columns=['imtf_min'])
df_cleaned = pd.concat([df_cleaned,imtf_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_min','r_imtf_min'],axis=1)

# Create average imtf_max metric
imtf_max =
pd.DataFrame((df_cleaned['l_imtf_max']+df_cleaned['r_imtf
_max'])/2, columns=['imtf_max'])
df_cleaned = pd.concat([df_cleaned,imtf_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_max','r_imtf_max'],axis=1)

# Create average imtf_std metric
imtf_std =
pd.DataFrame((df_cleaned['l_imtf_std']+df_cleaned['r_imtf
_std'])/2, columns=['imtf_std'])
df_cleaned = pd.concat([df_cleaned,imtf_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_std','r_imtf_std'],axis=1)

# Create average imtf_length metric
imtf_length =
pd.DataFrame((df_cleaned['l_imtf_length']+df_cleaned['r_i
mtf_length'])/2, columns=['imtf_length'])
df_cleaned = pd.concat([df_cleaned,imtf_length],axis = 1)
df_cleaned =
df_cleaned.drop(['l_imtf_length','r_imtf_length'],axis=1)

# Create average ld_mean metric
```

```
ld_mean =
pd.DataFrame((df_cleaned['l_ld_mean']+df_cleaned['r_ld_me
an'])/2, columns=['ld_mean'])
df_cleaned = pd.concat([df_cleaned,ld_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_mean','r_ld_mean'],axis=1)

# Create average ld_min metric
ld_min =
pd.DataFrame((df_cleaned['l_ld_min']+df_cleaned['r_ld_min
'])/2, columns=['ld_min'])
df_cleaned = pd.concat([df_cleaned,ld_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_min','r_ld_min'],axis=1)

# Create average ld_max metric
ld_max =
pd.DataFrame((df_cleaned['l_ld_max']+df_cleaned['r_ld_max
'])/2, columns=['ld_max'])
df_cleaned = pd.concat([df_cleaned,ld_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_max','r_ld_max'],axis=1)

# Create average ld_std metric
ld_std =
pd.DataFrame((df_cleaned['l_ld_std']+df_cleaned['r_ld_std
'])/2, columns=['ld_std'])
df_cleaned = pd.concat([df_cleaned,ld_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_std','r_ld_std'],axis=1)

# Create average ld_legth metric
ld_legth =
pd.DataFrame((df_cleaned['l_ld_legth']+df_cleaned['r_ld_l
egth'])/2, columns=['ld_legth'])
df_cleaned = pd.concat([df_cleaned,ld_legth],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ld_legth','r_ld_legth'],axis=1)

# Create average ad_mean metric
```

```
ad_mean =
pd.DataFrame((df_cleaned['l_ad_mean']+df_cleaned['r_ad_me
an'])/2, columns=['ad_mean'])
df_cleaned = pd.concat([df_cleaned,ad_mean],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_mean','r_ad_mean'],axis=1)

# Create average ad_min metric
ad_min =
pd.DataFrame((df_cleaned['l_ad_min']+df_cleaned['r_ad_min
'])/2, columns=['ad_min'])
df_cleaned = pd.concat([df_cleaned,ad_min],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_min','r_ad_min'],axis=1)

# Create average ad_max metric
ad_max =
pd.DataFrame((df_cleaned['l_ad_max']+df_cleaned['r_ad_max
'])/2, columns=['ad_max'])
df_cleaned = pd.concat([df_cleaned,ad_max],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_max','r_ad_max'],axis=1)

# Create average ad_std metric
ad_std =
pd.DataFrame((df_cleaned['l_ad_std']+df_cleaned['r_ad_std
'])/2, columns=['ad_std'])
df_cleaned = pd.concat([df_cleaned,ad_std],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_std','r_ad_std'],axis=1)

# Create average ad_legth metric
ad_length =
pd.DataFrame((df_cleaned['l_ad_length']+df_cleaned['r_ad_
length'])/2, columns=['ad_length'])
df_cleaned = pd.concat([df_cleaned,ad_length],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ad_length','r_ad_length'],axis=1)

# Create average rough_near metric
```

```
rough_near =
pd.DataFrame((df_cleaned['l_rough_near']+df_cleaned['r_ro
ugh_near'])/2, columns=['rough_near'])
df_cleaned = pd.concat([df_cleaned,rough_near],axis = 1)
df_cleaned =
df_cleaned.drop(['l_rough_near','r_rough_near'],axis=1)

# Create average rough_far metric
rough_far =
pd.DataFrame((df_cleaned['l_rough_far']+df_cleaned['r_rou
gh_far'])/2, columns=['rough_far'])
df_cleaned = pd.concat([df_cleaned,rough_far],axis = 1)
df_cleaned =
df_cleaned.drop(['l_rough_far','r_rough_far'],axis=1)

# Create average cca_ps metric
cca_ps =
pd.DataFrame((df_cleaned['l_cca_ps']+df_cleaned['r_cca_ps
'])/2, columns=['cca_ps'])
df_cleaned = pd.concat([df_cleaned,cca_ps],axis = 1)
df_cleaned =
df_cleaned.drop(['l_cca_ps','r_cca_ps'],axis=1)

# Create average cca_ed metric
cca_ed =
pd.DataFrame((df_cleaned['l_cca_ed']+df_cleaned['r_cca_ed
'])/2, columns=['cca_ed'])
df_cleaned = pd.concat([df_cleaned,cca_ed],axis = 1)
df_cleaned =
df_cleaned.drop(['l_cca_ed','r_cca_ed'],axis=1)

# Create average ica_ps metric
ica_ps =
pd.DataFrame((df_cleaned['l_ica_ps']+df_cleaned['r_ica_ps
'])/2, columns=['ica_ps'])
df_cleaned = pd.concat([df_cleaned,ica_ps],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ica_ps','r_ica_ps'],axis=1)

# Create average ica_ed metric
```

```
ica_ed =
pd.DataFrame((df_cleaned['l_ica_ed']+df_cleaned['r_ica_ed
'])/2, columns=['ica_ed'])
df_cleaned = pd.concat([df_cleaned,ica_ed],axis = 1)
df_cleaned =
df_cleaned.drop(['l_ica_ed','r_ica_ed'],axis=1)

# Create average cswa metric
cswa =
pd.DataFrame((df_cleaned['lcswa']+df_cleaned['rcswa'])/2,
columns=['cswa'])
df_cleaned = pd.concat([df_cleaned,cswa],axis = 1)
df_cleaned = df_cleaned.drop(['lcswa','rcswa'],axis=1)

# Create average acimt metric
acimt =
pd.DataFrame((df_cleaned['l_acimt']+df_cleaned['r_acimt']
)/2, columns=['acimt'])
df_cleaned = pd.concat([df_cleaned,acimt],axis = 1)
df_cleaned =
df_cleaned.drop(['l_acimt','r_acimt'],axis=1)

# Create average delta_ld metric
delta_ld =
pd.DataFrame((df_cleaned['l_delta_ld']+df_cleaned['r_delt
a_ld'])/2, columns=['delta_ld'])
df_cleaned = pd.concat([df_cleaned,delta_ld],axis = 1)
df_cleaned =
df_cleaned.drop(['l_delta_ld','r_delta_ld'],axis=1)

# Create average strain metric
strain =
pd.DataFrame((df_cleaned['l_strain']+df_cleaned['r_strain
'])/2, columns=['strain'])
df_cleaned = pd.concat([df_cleaned,strain],axis = 1)
df_cleaned =
df_cleaned.drop(['l_strain','r_strain'],axis=1)

# Create average distensibility metric
distensibility =
pd.DataFrame((df_cleaned['l_distensibility']+df_cleaned['
r_distensibility'])/2, columns=['distensibility'])
```

```
df_cleaned = pd.concat([df_cleaned,distensibility],axis =
1)
df_cleaned =
df_cleaned.drop(['l_distensibility','r_distensibility'],a
xis=1)

# Create average compliance metric
compliance =
pd.DataFrame((df_cleaned['l_compliance']+df_cleaned['r_co
mpliance'])/2, columns=['compliance'])
df_cleaned = pd.concat([df_cleaned,compliance],axis = 1)
df_cleaned =
df_cleaned.drop(['l_compliance','r_compliance'],axis=1)

# Create average pem metric
bsi =
pd.DataFrame((df_cleaned['l_bsi']+df_cleaned['r_bsi'])/2,
columns=['bsi'])
df_cleaned = pd.concat([df_cleaned,bsi],axis = 1)
df_cleaned = df_cleaned.drop(['l_bsi','r_bsi'],axis=1)

# Create average pem metric
pem =
pd.DataFrame((df_cleaned['l_pem']+df_cleaned['r_pem'])/2,
columns=['pem'])
df_cleaned = pd.concat([df_cleaned,pem],axis = 1)
df_cleaned = df_cleaned.drop(['l_pem','r_pem'],axis=1)

# Create average pem metric
yem =
pd.DataFrame((df_cleaned['l_yem']+df_cleaned['r_yem'])/2,
columns=['yem'])
df_cleaned = pd.concat([df_cleaned,yem],axis = 1)
df_cleaned = df_cleaned.drop(['l_yem','r_yem'],axis=1)

# Clean patient data
df_cleaned['diabetes'] =
df_cleaned['diabetes'].replace(2, np.nan)
df_cleaned['cholesterol'] =
df_cleaned['cholesterol'].replace(2, np.nan)
df_cleaned['heart_disease'] =
df_cleaned['heart_disease'].replace(2, np.nan)
```

```
df_cleaned['mother_hypertension'] =
df_cleaned['mother_hypertension'].replace(2, np.nan)
df_cleaned['father_hypertension'] =
df_cleaned['father_hypertension'].replace(2, np.nan)
df_cleaned['mother_cholesterol'] =
df_cleaned['mother_cholesterol'].replace(2, np.nan)
df_cleaned['father_cholestrol'] =
df_cleaned['father_cholestrol'].replace(2, np.nan)
df['mother_heart_disease'] =
df['mother_heart_disease'].replace(1, np.nan)
df['mother_heart_disease'] =
df['mother_heart_disease'].replace([2,3,4], 1)
df['father_heart_disease'] =
df['father_heart_disease'].replace(1, np.nan)
df['father_heart_disease'] =
df['father_heart_disease'].replace([2,3,4], 1)
df['mother_stroke'] = df['mother_stroke'].replace(1,
np.nan)
df['mother_stroke'] =
df['mother_stroke'].replace([2,3,4], 1)
df['father_stroke'] = df['father_stroke'].replace(1,
np.nan)
df['father_stroke'] =
df['father_stroke'].replace([2,3,4], 1)
df['mother_diabetes'] = df['mother_diabetes'].replace(1,
np.nan)
df['mother_diabetes'] =
df['mother_diabetes'].replace([2,3,4], 1)
df['father_diabetes'] = df['father_diabetes'].replace(1,
np.nan)
df['father_diabetes'] =
df['father_diabetes'].replace([2,3,4], 1)

# Create prehypertensive metric
df_cleaned['prehypertensive'] = np.nan
df_cleaned.loc[(df_cleaned['sbp'] <= 119) &
(df_cleaned['dbp'] <= 79), 'prehypertensive'] = 0
df_cleaned.loc[(df_cleaned['sbp'] < 140) &
(df_cleaned['sbp'] > 119) | (df_cleaned['dbp'] < 90) &
(df_cleaned['dbp'] > 79), 'prehypertensive'] = 1

# Create obese metric
```

```
df_cleaned['obese'] = np.nan
df_cleaned.loc[(df_cleaned['bmi'] <= 25), 'obese'] = 0
df_cleaned.loc[(df_cleaned['bmi'] > 25), 'obese'] = 1

# Create left ventrical hypertrophy metric
df_cleaned['lvh'] = np.nan
df_cleaned.loc[((df_cleaned['lvm_cube'] <= 115) &
(df_cleaned['sex'] == 1)) | ((df_cleaned['lvm_cube'] <=
95) & (df_cleaned['sex'] == 0)), 'lvh'] = 0
df_cleaned.loc[((df_cleaned['lvm_cube'] > 115) &
(df_cleaned['sex'] == 1)) | ((df_cleaned['lvm_cube'] >
95) & (df_cleaned['sex'] == 0)), 'lvh'] = 1

# Create sedentary metric
df_cleaned['sedentary'] = np.nan
df_cleaned.loc[df_cleaned['vig_work_met_min'] +
df_cleaned['moderate_work_met_minutes'] +
df_cleaned['travel_met_minutes'] +
df_cleaned['vig_exercise_met'] +
df_cleaned['moderate_exercise_met'] >= 600,'sedentary'] =
0
df_cleaned.loc[df_cleaned['vig_work_met_min'] +
df_cleaned['moderate_work_met_minutes'] +
df_cleaned['travel_met_minutes'] +
df_cleaned['vig_exercise_met'] +
df_cleaned['moderate_exercise_met'] < 600,'sedentary'] =
1

# Create smoker metric
df_cleaned['smoker'] = np.nan
df_cleaned.loc[(df_cleaned['cotinine'] < 11) |
(df_cleaned['smoke'] == 0),'smoker'] = 0
df_cleaned.loc[(df_cleaned['cotinine'] >= 11) &
(df_cleaned['smoke'] == 1),'smoker'] = 1

# Create excessive alcohol metric
df_cleaned['excessive_alcohol'] = np.nan
df_cleaned.loc[(df_cleaned['ggt'] < 49) |
(df_cleaned['alcohol'] == 0),'excessive_alcohol'] = 0
df_cleaned.loc[(df_cleaned['ggt'] >= 49) &
(df_cleaned['alcohol'] == 1),'excessive_alcohol'] = 1
```

```
# Create dyslipidemic metric
df_cleaned['dyslipidemic'] = np.nan
df_cleaned.loc[df_cleaned['ldl'] < 3.5,'dyslipidemic'] =
0
df_cleaned.loc[df_cleaned['ldl'] >= 3.5,'dyslipidemic'] =
1

# Select Features
numeric_features =
['age_b','sbp','dbp','map','pp','bh','bw','wc','hc','nc',

'bmi','waist_hip_ratio','chol','trig','hdl','ldl','glu',
                      'cotinine','ggt','lvm_cube']

categoric_features =
['sex','ethnicity','heart_attack','mother_hypertension','
father_hypertension',

'mother_cholesterol','father_cholestrol','mother_heart_di
sease',

'father_heart_disease','mother_stroke','father_stroke','m
other_diabetes',

'father_diabetes','clinic_bp_status']

# Create Dataset
dataset = df_cleaned.copy()
dataset =
dataset[numeric_features+categoric_features+[label]]

# Drop Rows Missing Label Values
dataset.dropna(subset = 'abpm_overall_ht', inplace =
True)

# Drop Rows with Hypertensive Patients
dataset = dataset[dataset['clinic_bp_status'] != 1]

# Remove Columns Missing More Than 50% of Data
missing_percent_columns = (dataset.isnull().sum() /
len(dataset)) * 100
```

```
columns_to_drop =
missing_percent_columns[missing_percent_columns >
50].index.tolist()
dataset = dataset.drop(columns=columns_to_drop)

# Remove Rows Missing More Than 50% of Data
missing_percent_rows = (dataset.isnull().sum(axis=1) /
len(dataset.columns)) * 100
rows_to_drop = missing_percent_rows[missing_percent_rows
> 50].index.tolist()
dataset = dataset.drop(index=rows_to_drop)

# Remove Columns With Only 1 Value
unique_counts = dataset.nunique()
unique_columns = unique_counts[unique_counts ==
1].index.tolist()
dataset = dataset.drop(columns=unique_columns)

columns_to_drop = columns_to_drop + unique_columns
# Remove Column Names from Numeric and Categorical
Feature Lists
for column in columns_to_drop:
    if column in numeric_features:
        numeric_features.remove(column)
    elif column in categoric_features:
        categoric_features.remove(column)

normotensive_dataset = dataset[dataset['abpm_overall_ht']
== 0]
normotensive_dataset =
normotensive_dataset.drop('abpm_overall_ht',axis=1)
mht_dataset = dataset[dataset['abpm_overall_ht'] == 1]
mht_dataset = mht_dataset.drop('abpm_overall_ht',axis=1)

# Evaluate statistical differences between Normotensive
and MHT group
def
evaluate_groups(group1,group2,numeric_features,categoric_
features):
    feature_name = []
    feature_p_val = []
    for feature in group1.columns:
```

```
        if feature in numeric_features:
            stat, p_val = bartlett(
                group1[feature].dropna(),
group2[feature].dropna())
            if p_val < 0.05:
                stat, p_val = kruskal(
                    group1[feature].dropna(),
group2[feature].dropna())
            else:
                stat, p_val = shapiro(
                    pd.concat([group1[feature].dropna(),
group2[feature].dropna()], axis=0))
                if p_val < 0.05:
                    stat, p_val = kruskal(
                        group1[feature].dropna(),
group2[feature].dropna())
                else:
                    stat, p_val =
ttest_ind(group1[feature].dropna(),
group2[feature].dropna(), equal_var=False)
            feature_name.append(feature)
            feature_p_val.append(p_val)
        elif feature in categoric_features:
            contingency_table =
np.array([[(group1[feature] == 0).sum(), (group2[feature]
== 0).sum()],

[(group1[feature] == 1).sum(), (group2[feature] ==
1).sum()]])
            stat, p_val, dof, expected =
chi2_contingency(contingency_table)
            feature_name.append(feature)
            feature_p_val.append(p_val)
        else:
            print('Error: feature not recognized')
            sys.exit()
    return pd.concat([pd.DataFrame(feature_name),
pd.DataFrame(feature_p_val)], axis=1)

group1_comparison =
evaluate_groups(normotensive_dataset,mht_dataset,numeric_
features,categoric_features)
```

```
shapiro_name = []
shapiro_p_val = []
for feature in dataset.columns:
    if feature in numeric_features:
        stat, p_val = shapiro(dataset[feature])
        shapiro_name.append(feature)
        shapiro_p_val.append(p_val)
normality_evaluation =
pd.concat([pd.DataFrame(shapiro_name),
pd.DataFrame(shapiro_p_val)], axis=1)

# Split into Features and Labels
X = dataset.drop('abpm_overall_ht',axis=1)
y = dataset['abpm_overall_ht']

# Split into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = split_ratio, random_state = SEED, stratify =
y)

group2_comparison =
evaluate_groups(X_train,X_test,numeric_features,categoric
_features)

numeric_feature_names = []
numeric_feature1_mean = []
numeric_feature1_std = []
numeric_feature2_mean = []
numeric_feature2_std = []
categoric_feature_names = []
categoric_feature1_number = []
categoric_feature1_percentage = []
categoric_feature2_number = []
categoric_feature2_percentage = []
for column in normotensive_dataset.columns:
    if column in numeric_features:
        numeric_feature_names.append(column)
        normotensive_data = normotensive_dataset[column]
        normotensive_mean = normotensive_data.mean()
        normotensive_std = normotensive_data.std()
        numeric_feature1_mean.append(normotensive_mean)
```

```
        numeric_feature1_std.append(normotensive_std)
        mht_data = mht_dataset[column]
        mht_mean = mht_data.mean()
        mht_std = mht_data.std()
        numeric_feature2_mean.append(mht_mean)
        numeric_feature2_std.append(mht_std)
    if column in categoric_features:
        categoric_feature_names.append(column)
        normotensive_data = normotensive_dataset[column]
        mht_data = mht_dataset[column]
        number1 = normotensive_data.sum()
        percentage1 = number1/len(normotensive_data)
        categoric_feature1_number.append(number1)
        categoric_feature1_percentage.append(percentage1)
        number2 = mht_data.sum()
        percentage2 = number2/len(mht_data)
        categoric_feature2_number.append(number2)
        categoric_feature2_percentage.append(percentage2)
numeric_dataframe = pd.DataFrame(
    {'Feature': numeric_feature_names,
     'Normotensive_Mean': numeric_feature1_mean,
     'Normotensive_StDev': numeric_feature1_std,
     'MHT_Mean': numeric_feature2_mean,
     'MHT_StDev': numeric_feature2_std
    })
categoric_dataframe = pd.DataFrame(
    {'Feature': categoric_feature_names,
     'Normotensive_Number': categoric_feature1_number,
     'Normotensive_Percentage':
categoric_feature1_percentage,
     'MHT_Number': categoric_feature2_number,
     'MHT_Percentage':categoric_feature2_percentage
     })

numeric_feature_names = []
numeric_feature1_mean = []
numeric_feature1_std = []
numeric_feature2_mean = []
numeric_feature2_std = []
categoric_feature_names = []
categoric_feature1_number = []
categoric_feature1_percentage = []
```

```
categoric_feature2_number = []
categoric_feature2_percentage = []
for column in X_train.columns:
    if column in numeric_features:
        numeric_feature_names.append(column)
        X_train_data = X_train[column]
        X_train_mean = X_train_data.mean()
        X_train_std = X_train_data.std()
        numeric_feature1_mean.append(X_train_mean)
        numeric_feature1_std.append(X_train_std)
        X_test_data = X_test[column]
        X_test_mean = X_test_data.mean()
        X_test_std = X_test_data.std()
        numeric_feature2_mean.append(X_test_mean)
        numeric_feature2_std.append(X_test_std)
    if column in categoric_features:
        categoric_feature_names.append(column)
        X_train_data = X_train[column]
        X_test_data = X_test[column]
        number1 = X_train_data.sum()
        percentage1 = number1/len(X_train_data)
        categoric_feature1_number.append(number1)
        categoric_feature1_percentage.append(percentage1)
        number2 = X_test_data.sum()
        percentage2 = number2/len(X_test_data)
        categoric_feature2_number.append(number2)
        categoric_feature2_percentage.append(percentage2)
numeric2_dataframe = pd.DataFrame(
    {'Feature': numeric_feature_names,
     'Train_Mean': numeric_feature1_mean,
     'Train_StDev': numeric_feature1_std,
     'Test_Mean': numeric_feature2_mean,
     'Test_StDev': numeric_feature2_std
    })
categoric2_dataframe = pd.DataFrame(
    {'Feature': categoric_feature_names,
     'Train_Number': categoric_feature1_number,
     'Train_Percentage': categoric_feature1_percentage,
     'Test_Number': categoric_feature2_number,
     'Test_Percentage':categoric_feature2_percentage
     })
```