Linear Regression Feature Engineering in Classification Tree Learning

by

Jacob William Renn

A Dissertation Presented in Partial Fulfillment

of the Requirements for the Degree

PhD in Technology

CAPITOL TECHNOLOGY UNIVERSITY

April 2022

Linear Regression Feature Engineering in Classification Tree Learning

Approved:

Dr. Ian McAndrew, PhD FRAeS, Chair

Dr. Benjamin Harvey, DSc, Committee Member

Accepted and Signed:

Ian McAndrew 04/16/2022

Dr. Ian McAndrew Date

[illegible] 04/16/2022

Dr. Benjamin Harvey Date

Ian McAndrew 04/16/2022

Dr Ian R. McAndrew Date

Dean of Doctoral Programs

Capitol Technology University

ABSTRACT

Artificial intelligence has grown extensively, in both application and theory, in recent years. Much of the growth in application can be attributed to increased data generation and processing techniques, and much of the growth in theory can be attributed to advancements in deep learning techniques. As a result of these advancements, however, there is an increasing problem related to model interpretability and explainability of results; for many of the state-of-the-art techniques used today, it is extremely difficult to retrieve any useful information to explain why a model makes its decisions, meaning that many industries which require high levels of accountability, such as defense, health care, and other regulated industries such as insurance, cannot use many of the most powerful, modern modeling techniques. This study expands upon the body of work being developed in the field of explainable artificial intelligence, which focuses on building artificial intelligence with greater interpretability in mind. This study investigates how traditional classification trees, a prototypical interpretable machine learning algorithm, can be made more powerful but maintain much of their interpretability by identifying multivariate splits. The resulting algorithm, the Linear Regression Classification Tree, is then tested against many existing techniques, both interpretable and uninterpretable, to determine how its performance and explainability compares to other commonly used techniques.

Keywords: Artificial Intelligence, Explainable AI, Decision Trees

DEDICATION

This work is dedicated to my wife, Kelsey Renn, whose love, support, and honesty toward my (sometimes crazy) ideas keep me focused and motivated.

ACKNOWLEDGMENT

Throughout the writing of this dissertation, I have received a great deal of support and assistance. There are many individuals other than the ones I will list here who have helped me along the way.

I would first like to thank my advisor, Dr. Ian McAndrew, for guiding me throughout the entire process of completing the dissertation and helping me focus my efforts. I would also like to thank Dr. Benjamin Harvey, who has made an immeasurable amount of impact in my professional and academic careers, and Dr. Courtney Paulson, who supported some of my first independent research efforts when I was finishing my master's degree.

In addition, I would like to thank my wife, Kelsey Renn, without whom I would never be where I am today. Lastly, I want to thank my grandparents, John and Kathy Renn, who have shown me unending support throughout my lifetime and who have pushed me to be the best that I can be.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Background of Study

Machine learning (ML) and artificial intelligence (AI) have seen significant amounts of growth in recent years (Bianchini, Müller, & Pelletier, 2020). Advances in data processing speeds, data creation speeds, and ML algorithms have contributed to the transformation of AI from largely experimental and academic to operational and deployed (Bianchini, Müller, & Pelletier, 2020). However, some of the largest barriers to the widespread adoption of these technologies are user acceptance and policy requirements. In many enterprises, individuals who lack experience with ML often doubt its usefulness and do not trust its predictions in their specific domains (Zhu, Liapis, Risi, Bidarra, & Youngblood, 2018). Additionally, many organizations and industries face heavy legal and policy restrictions regarding the use of automated prediction techniques. Regulation and governance are also growing; data privacy concerns and inappropriate model biases have led to regulations such as the General Data Protection Regulation (GDPR) (European Commission, 2018) in the European Union. Regulations such as this represent a great victory for society at large, but they also represent a paradigm shift for those building AI applications: it is no longer enough to have models that provide accurate predictions, but rather also provide assurance that data is being used in a compliant manner with the rights and interests of the data collection subjects at the forefront.

Some of the most powerful predictive models, such as deep learning (DL) models, ensemble methods, and support vector machines (SVMs), are likewise some of the most difficult to explain and are logically unintuitive due to the complexity of the decision functions these models make (Wenzel, Galy-Fajou, Deutsch, & Kloft, 2017). When using many of these methods, especially deep neural networks, even those who create the models are largely unable to understand why a model makes the decisions it does, sometimes leading to unexpected biases in models (Kodiyan, 2019). With the rising privacy and data mishandling concerns discussed above, and especially for industries such as defense, healthcare, transportation, and insurance, this lack of interpretability can be a significant deterrent from adopting ML and AI techniques. While deep learning has advanced AI and ML tremendously in the past decade, for many industries there is a significant need to research and develop interpretable models further. If this research is not done, the risk is continued loss of adoption of ML and AI technologies in multiple critical industries, resulting in further risks to human health, safety, and possibly lives.

Problem Statement

With much of current ML research focusing on DL techniques, many industries and problem spaces which require high levels of interpretability and accountability when making decisions have suffered from lack of innovation in interpretable modeling technologies. Some of these industries include, but are not limited to, defense, health care, transportation, and insurance. Most recent innovations in interpretable modeling techniques have occurred years or decades ago. With interpretable ML technologies lagging their opaque counterparts, there is real risk that these industries will fall too far behind in the race to adopt and integrate ML and AI technologies, putting people at risk for a variety of reasons.

Purpose of Dissertation Study

While there has emerged out of the problem of interpretability a new field, namely "Explainable Artificial Intelligence" (XAI) (Barredo Arrieta, et al., 2019; Gunning & Aha, 2019), further research into this field reveals that most work has focused on adding interpretability to DL techniques and that most of the techniques developed in this field are not general in nature (Gilpin, et al., 2018). While techniques such as Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro, Singh, & Guestrin, 2016) have been shown to be useful in creating a level of explanation for any type of classification model, these methods still fall short in providing the highest levels of interpretability needed for some applications. This is due to the methods LIME uses to make its explanations; instead of explaining the model itself, LIME tries to describe how the model's predictions change on a single example by perturbing features randomly (Ribeiro, Singh, & Guestrin, 2016).

As an attempt to foster more widespread adoption of ML technologies in highly regulated industries and on problems that require more-complete understanding of a model's internal decision structure, this research will focus on developing and testing a new interpretable ML model, the Linear Regression Classification Tree (LRCT). The goal of this work will thus be to investigate whether LRCT models improve upon traditional classification trees by utilizing a new feature engineering technique which identifies multivariate linear and nonlinear splits. Part of the motivation behind this feature engineering technique is to partially mimic the internal state representations and complex relationships which neural networks learn (LeCun, Bottou, Bengio, & Haffner, 1998; Tarsala & Kasprzyk, 2021) and which improve their predictive capabilities.

If this research is successful, it will help improve the state of the art of interpretable ML. With the development of a successful LRCT algorithm, ML practitioners will have a more-powerful model type at their disposal when interpretability is required. By utilizing this new LRCT model, individuals and teams facing problems requiring interpretability will be able to meet strict regulatory requirements while also taking advantage of more recent innovations in the field and obtaining more accurate predictions.

Significance of the Study

In the long term, the success of this research will help improve the state of interpretable ML. With the development of a successful LRCT model, ML practitioners will have at their disposal a new, highly explainable model which could be deployed in mission-critical systems and application areas. Such a model would help bridge the gap between the most powerful predictive models and the most interpretable ones. It is also hoped that the divergence in approach of this work, specifically its focus on improving already-interpretable models rather than adding interpretability to powerful ones, will also attract interest in different areas of XAI.

Nature of Study

There is no question that the recent surge of ML and AI is one of the defining characteristics of today's world. Showing AI's prominence in everyday life, even YouTube has identified this trend and has made a short series dedicated to showcasing AI's feats, even hiring Robert Downey Jr. as host (The Age of AI, 2019). This attention is warranted; there have been rapid improvements in the state of the art for many difficult prediction tasks in the past decade. Not the least of these are the introduction of the convolutional neural network (Tarsala & Kasprzyk, 2021) and transformer architectures (Vaswani, et al., 2017), which have directly contributed to improvements in the state of the art in image recognition (Simonyan & Zisserman, 2014) and natural language processing (NLP), (Devlin, Chang, Lee, & Toutanova, 2018; Lan, et al., 2019; Liu, et al., 2019) respectively.

Researchers have, however, focused largely on DL techniques in recent times (Roscher, Bohn, Duarte, & Garcke, 2020). DL, for all its benefits, has a glaring drawback when it comes to interpretability. Large neural networks such as the now-famous BERT model (Devlin, Chang, Lee, & Toutanova, 2018) have millions of parameters, and though there is promising research that has been shown to drastically reduce the number of parameters needed to perform prediction tasks with similar performance levels (Frankle & Carbin, 2018), true DL model interpretability seems unlikely. What makes this problem even more pervasive is that artificial neural networks (ANNs) have been shown to learn undesired biases in data, such as implicit association of the word “doctor” more with “man” than with “woman” (Dev & Phillips, 2019). This problem of explainability has led to the rise of XAI, but most of the research in this field has been dedicated to improving ANN explanations, rather than improving predictive capabilities of already-interpretable models.

Despite current research efforts, ML is not synonymous with DL. There are many ML model types that have seen little-to-no innovation in years or even decades. One such model type, often considered one of the most interpretable due to its simple-to-understand decision structure, is the decision tree. While there have been a number of innovations, such as random forests (Breiman, Random Forests, 2001) or adaboost (Freund & Schapire, 1997), which use ensembles of decision trees, little progress has been made to improve the underlying learning algorithm of a single decision tree since their inception. This phenomenon will be explored in more depth in Chapter 2 of this work, as it will lay the groundwork for the research done in this study.

Hypotheses/Research Questions

To answer the problem at hand, there are several questions to be answered and assumptions to be made. Firstly, the primary question to be addressed in this research study is:

Can an inherently interpretable algorithm be developed with improved performance over traditional interpretable models?

To answer this question, this work will focus on devising and testing an improvement to classification trees. The classification tree was chosen for the base learning algorithm because of (a) its high interpretability levels and intuitive decision structure, (b) the prevalence of tree-based models in practice, and (c) the fruitful research in developing further iterations based on other decision tree models (Breiman, Random Forests, 2001; Freund & Schapire, 1997; Chen & Guestrin, 2016; Friedman, 2002). To help answer the primary research question (RQ), the following RQs have also been posed to be answered in this work:

- RQ1: What methods have been created to induce multivariate splits in classification trees, and what are their shortcomings?
- RQ2: Does applying linear regression as a feature engineering and selection technique in classification tree learning improve predictive performance over traditional classification trees?
- RQ3: Do procedures analyzed in RQ2 also improve predictive performance over three of the most common interpretable algorithms?
- RQ4: How does this new algorithm compare in predictive performance to methods found in RQ1?

To answer the RQs above, and ultimately to answer the primary RQ, the following research goals have been identified. Through successful completion of these goals, a rigorous argument can be made for the validity and impacts of this research. The following outlines the goals and their contributions toward answering the proposed RQs:

1. **Conduct literature review to explore previous work in multivariate classification tree learning.** To increase level of expertise and show the novelty of the algorithm developed through this research, a thorough literature review is conducted to identify previous methods for creating multivariate splits in classification trees. This review will provide an answer to RQ1, will provide a foundation to answering RQ4, and will assist in identifying the advancements made by this research.

2. **Develop a methodology for creating multivariate splits in classification trees.** After researching existing methods for creating multivariate splits in classification trees, research will focus on creating a new method for doing so. Specifically, the goal of this new method will be approximating optimal linear and polynomial splits utilizing a combination of binning techniques and linear regression. This work will expand on the method developed and show how it is able to create accurate estimates of the multivariate surface function which optimally divides the classes being learned. Completing this step assists in answering RQ2, RQ3, and RQ4.

3. **Apply multivariate split learning procedure to classification tree learning, formalizing the LRCT algorithm.** After the splitting procedure is created, the method will be applied to classification tree learning, resulting in the final algorithm this research aims to develop. This work will assist in answering RQ2, RQ3, and RQ4.

4. **Conduct a series of experiments utilizing artificial data to test algorithm performance to learn specific multivariate decision boundaries.** Next, a series of experiments will be

conducted to test the LRCT algorithm's ability to learn various decision boundaries, such as linear and polynomial splits across varying numbers of variables. In these experiments, various other kinds of models will be utilized as well, thus giving an estimate of comparative performance for LRCT against other model types. This step will assist in answering RQ2, RQ3, and RQ4.

5. **Conduct a series of experiments utilizing collected, real-world data.** After completing experiments on artificial data, experiments on "real-world" data will be conducted. Because the goal of this research is to develop an interpretable algorithm that improves predictive performance over traditional interpretable methods, the LRCT algorithm must be tested on true, collected data. Without such experiments, this study would be incomplete; with these experiments, a better estimate of true performance will be achieved. This step will assist in answering RQ2, RQ3, and RQ4.

6. **Analyze experiment data, make conclusions, and recommend direction of future work.** Once experimentation is complete, the results will be analyzed in depth. Model performances will be compared and analyzed to determine answers to research questions RQ2, RQ3, and RQ4. After these answers are provided, directions for future work will be recommended.

Definitions of Fundamental Terminology

Before proceeding, it is important to clearly define some important terms. These terms, while not all-encompassing in this research, will properly frame this work and provide clarity for the reader.

- **Artificial Intelligence** is defined as the broad, interdisciplinary field of study which seeks to endow computer systems with behaviors that mimic human intelligence. While this definition is rather simple, the field of artificial intelligence is extremely complex and utilizes findings from many fields, including computer science, mathematics, statistics, neuroscience, psychology, and others.

- **Machine Learning** is a sub-field of Artificial Intelligence which is focused on creating and using algorithms to create and derive insight using past data. This is done by creating a function which maps input data to some output, typically with the goal of minimizing a loss function, which serves as a measure for the overall error of the learned function.

- **Supervised Learning** is a subset of machine learning where training data comes of the form of input and output pairs. The learning algorithm uses these input data to learn a function which accurately predicts the output values from the input data. The defining characteristic of supervised learning is that it makes use of already-labeled data.

- **Unsupervised Learning** is a subset of machine learning where training data is not labeled. Instead of trying to predict specific outcomes, the goal of unsupervised learning is to discover underlying relationships within the data being modeled. This is typically done by identifying some form of underlying groups within the training data. This work will not focus on unsupervised learning techniques.

- **Regression** is a subset of supervised learning in which the value to be predicted lies on a continuum. Take for example the problem of using features of a home to predict the selling price of the home. For simplification purposes, the possible values for a home selling price may be taken to be any positive number, thus making this problem a regression problem. This work will not focus on regression techniques.

- **Classification** is a second subset of supervised learning in which the value to be predicted lies within a set of categories instead of on a continuum. For example, taking an image of a handwritten digit and identifying the digit that is drawn would be an example of a classification problem, as the label for the handwritten image could only be one of ten distinct values. Each of the images would belong to one distinct category which corresponds to the number represented in the image.

- **Features** are the inputs to a ML algorithm. As ML models are mathematical models, each of these features supplied to the algorithm must be numeric or somehow converted to a numeric value. Continuing with the previous two examples: the image classification regarding photos of handwritten digits and the product price regression problem, there are several features available for both scenarios. In the image classification problem, each of the pixels in the images is a feature. These pixels have intensity values, often ranging from 0 to 255 for each of the red, green, and blue color channels, if the images are in color, or simply a single value between 0 and 255 for greyscale images. In contrast, there may be many different features available in the price regression problem, including the number of bedrooms for the home, the number of bathrooms, the location, the lot size, and others.

- **Feature Engineering** is the process of deriving new features using features already available. This process can take many forms, from creating combinations of variables to complicated preprocessing steps. This stage is often one of the most important in the ML pipeline, as discovering information rich features - features that are highly correlated with the label - can lead to simpler, more-robust, and overall better models.

At its core, ML is a field focused on creating and using algorithms to allow a computer to automatically detect, identify, and exploit relationships in data for predictive purposes. This is done by “learning” a function that maps input data to some output, typically with a goal of minimizing a loss function - a way of measuring the error of the learned function. ML has existed long before the recent surge in deep learning techniques. This work will explore advances in interpretable methods, specifically methods based on classification trees. This work will then explore and present a new method for utilizing linear regression and impurity-based splitting techniques as an automatic feature engineering method in classification tree learning. The result is a new, explainable classification tree model which can make linear and nonlinear multivariate splits. This work highlights the novel capabilities this new model has and showcases its capabilities in multiple learning situations.

In addition to a generalized vocabulary, developing a unified terminology for XAI is essential for the success of the research to be conducted. While a single, unified terminology has not been universally agreed upon due to the field still being in its infancy, work has been done to develop this standard terminology; this research will rely heavily on the review conducted in (Barredo Arrieta, et al., 2019) for key definitions and vocabulary. A list of essential terms follows:

- **Understandability** is a model characteristic. This characteristic refers to the ability for a human to understand the model’s inner working without need for explaining its internal structure of how the model processes its input data (Montavon, Samek, & Muller, 2018).
- **Comprehensibility** is the ability of an algorithm to demonstrate its knowledge about its given task in a way that is understandable to humans (Ferndandez, Herrera, Cordon, Jose del Jesus, & Marcelloni, 2019; Gleicher, 2016; Odense & Garcez, 2020).

- **Interpretability** is another ability a model can possess. In contrast to *comprehensibility*, interpretability is the ability of a model to either explain or provide meaning in human understandable terms (Barredo Arrieta, et al., 2019).
- **Explainability**, in terms of ML, refers to the providing of an explanation using an *interface* between a human and the model (Guidotti, et al., 2018).
- **Transparency** refers to a characteristic of a model to by itself be understandable (Lipton, 2017). This family of models can be further broken down into three subgroups, *simulatable models*, *decomposable models*, and *algorithmically transparent models*. An exploration of this breakdown will be done in a later section exploring current theories in XAI.

Scope, Limitations, and Delimitations

It is important both for the reader and for research purposes to identify the scope and limitations of this research. Concretely defining these will assist in both focusing and framing this research, as well as help in identifying direction for future work.

This study will primarily focus on applied results, rather than purely theoretical ones. While the theory behind the LRCT algorithm defined here will be presented and discussed in depth in a further chapter, this work will focus on developing an understanding of how the algorithm performs in a variety of situations. The ideas in (Breiman, Statistical Modeling: The Two Cultures, 2001) are drawn upon heavily for theoretical justification in this regard. In that work, the author cites his experience to describe two cultures in the statistical modeling community. The first of these cultures is largely focused on modeling and understanding the way data is generated; the second is focused on modeling the outcomes without regard for the data generation model. The author argues that this second “culture” often leads to more-general classes of models, as no assumptions about the data are required. The research conducted in this current study follows with those assertions the author makes about the second culture; the model developed and presented here makes minimal assumption about the underlying relationships or structure within the data but instead learns in a way that attempts to generalize upon the data presented.

The main goal of this research will be to develop an interpretable model which outperforms existing interpretable models. This work is therefore scoped such that predictive performance is the primary goal. In Chapter 3, the theoretical aspects and design choices for the algorithm will be presented. This chapter is included to explain the model’s inner logic and provide a basis for *why* the algorithm can outperform traditional interpretable methods, but Chapter 4 will contain the empirical justification that the model does in fact outperform these other methods using experimental results. While the contents of Chapter 3 are vital to this research, the true focus is on the applicability and utility of the methods presented in this work.

Clearly, the limitation of such a design choice is that the results rely on the data to be modeled. It would be possible to create data that would favor one algorithm type over the other. The design of this research takes multiple measures to mitigate such factors. Firstly, in the experiments on artificial data, multiple experiments are run with various data generation techniques. While certain data will be expected to favor one model type over another, care will be taken to run multiple experiments with various properties to help understand generalized behavior. Additionally, multiple experiments will be run on well-known, real world collected datasets.

Chapters Overview

This work will consist of five chapters, the first of which is this introduction. The following chapters will each address an important component of the research, including a thorough literature review, formal definition of the LRCT algorithm and experimental methods, results, and conclusions. Further detail on these chapters can be found in the following list.

- Chapter 2 consists of a thorough literature review regarding many of the topics this work falls under. A historical perspective of machine learning is provided, along with a review of decision tree techniques and recent developments in decision tree technologies. Additionally, literature surrounding the seminal field of "Explainable AI" is reviewed.

- Chapter 3 is where the LRCT algorithm is defined and explained. This chapter enumerates the design choices made and presents a mathematical reasoning behind those design choices. Detailed overviews of the splitting and learning procedures are explained here. After defining the algorithm, Chapter 3 then outlines the experimental design and methods for future chapters.

- Chapter 4 reports on experiments testing LRCT against artificial and collected data. An integral part of this research's design, these experiments allow for testing the algorithm against specific data patterns and relationships under controlled circumstances as well as testing the algorithm on real-world data. This chapter provides the experimental evidence which is then used to draw conclusions to answer this work's research questions.
- Chapter 5 builds upon the work done in all previous chapters to draw conclusions and recommend future directions of work. In this chapter, the results of all previous experiments are examined in detail to determine the answers to each of the research

questions posed earlier in Chapter 1. Once each of the questions are addresses, the results of this work are examined in further detail to recommend further questions to be answered, providing a recommendation for future research in this area.

Chapter Summary

In this chapter, an overview of the background surrounding this dissertation study was provided. First, a brief overview of AI techniques was presented, followed by a discussion of the problem to be addressed by this research. Next, a more detailed description of the purpose and significance of the study to be done was completed, including a set of primary research questions to be answered and steps to be completed. Thirdly, an overview of fundamental terminology in AI was presented, and finally an outline of the future chapters in this work was presented. In the next chapter, a detailed overview of existing literature will be conducted to provide context and framing for this research.

CHAPTER 2: LITERATURE REVIEW

Despite the recent surge of interest in the public's eye, ML and AI are not "new" fields. Alan Turing, considered by many to be the father of modern computer science, famously posed the question "can machines think?" in a work published in 1950 (Turing, 1950), marking the birth of modern AI. This question has garnered much interest in the time since it was asked; perhaps the most thought-provoking of the responses came in 1980 in a paper titled "Minds, Brains, and Programs" in which the author presents what is now known as the *Chinese Room Argument* (Searle, 1980). The author argues that mimicking human capabilities – he uses the example of question answering as a proxy of true language understanding – does not imply true "thought," at least in a human sense. This analogy can be likened to a parrot, which can learn to replicate the sounds of human language but cannot understand the meaning of what it is saying.

Despite the arguments noted in the preceding paragraph regarding whether AI in its current state truly can be considered intelligent, research in and adoption of ML technologies, particularly DL technologies, have defined a large portion of computer science in the twenty first century. Having identified these trends, this work will seek to explore and expand upon some both current and past developments in the field. In particular, the seminal field of XAI will be examined through the lens of the much older technology of classification trees. Literature in both areas will be examined and analyzed to frame the research and provide a more thorough understanding of the current state of these technology areas. In this chapter, that review is conducted. Firstly, research into decision tree technologies will be conducted, followed by research into XAI. The decision tree review will focus heavily on classification tree technologies, as the research to be conducted will focus on developing a new type of classification tree. The review of XAI will highlight the field, provide definitions to critical terminologies, and highlight some of the gaps in the literature. Once these reviews are completed, their intersection will be examined to identify gaps in theory and provide a foundational setting for the research to be conducted.

Decision Trees

Decision trees, specifically classification trees, can be considered one of the prototypical interpretable ML algorithms (Brodley, Utgoff, & Kibler, 1995; Awaysheh, et al., 2019). The simple-to-follow decision structure these models exhibit lends itself perfectly for human interpretation (Brodley, Utgoff, & Kibler, 1995). Consider for example Figure 1, taken from (Breiman, Friedman, Olshen, & Stone, 1984), which contains a diagram of a tree trained to predict high and low risk heart attack patients. Notice the simplicity of the decision structure; only a maximum of three data points from a patient is needed to make a decision. Furthermore, the decision structure is easily understood in human terms, and the resulting model can be examined and validated by physicians to ensure that scientifically valid relationships are learned and utilized. In the following section, a brief history of decision tree technologies will be presented. Following this history will be a discussion of more recent theories and evolutions with regards to decision trees. Finally, alternative theories and gaps in the literature will be reviewed as well.

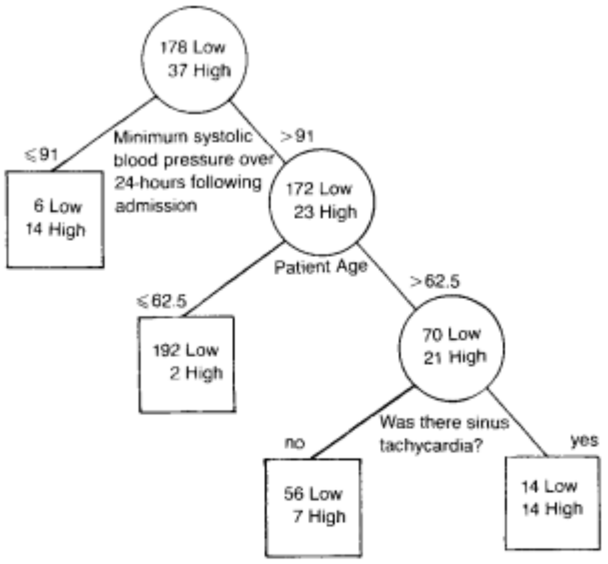

Figure 1. Heart Attack Risk Classification Decision Tree, taken from (Breiman, Friedman, Olshen, & Stone, 1984).

Brief History

Decision trees are perhaps most commonly associated with Classification And Regression Trees (CART) (Breiman, Friedman, Olshen, & Stone, 1984), but the methodology predates these authors' implementation by decades. The first regression tree algorithm, "Automatic Interaction Detection" (AID), was published in 1963 (Morgan & Sonquist, 1963). AID utilizes a greedy search to fit a piecewise-constant regression model on training data. Much like other regression tree algorithms, AID creates this decision function by splitting and reducing individual node impurities; specifically, the impurity function used in the original algorithm is the sum of squared errors, seen in Equation 1. In this equation, let n denote the individual node under consideration, y_i be the true value for instance $i,$ and $\overline{y_\iota}$ be the value predicted by the tree for the instance.

Nearly ten years later, in 1972, Messenger and Mandell created the first classification tree algorithm, "Theta Automatic Interaction Detection" (THAID) (Messenger & Mandell, 1972). This algorithm behaves very similarly to contemporary models, by creating recursive splits to attempt to maximize the number of instances in the modal category. This modal category is then taken to be the predicted class at that node.

In 1980, Kass created the Chi-squared Automatic Interaction Detection tree (CHAID), which furthered the theories behind decision trees by utilizing a merge and split procedure which applies a chi-squared test statistic (Kass, 1980). Despite its effectiveness, CHAID's major drawback is that it is only capable of splitting on categorical predictors (Wilkinson, 1992).

$$\Phi(n) = \sum_{i \in n} (y_i - \overline{y_i})^2$$

Equation 1. Sum of Squared Errors. Let n refer to the individual node under consideration, i refer to an instance located at that node, y_i be the actual target variable for that instance, and $\overline{y_i}$ be the predicted target variable for that instance.

It was not until 1984 when the now-famous *Classification and Regression Trees* (CART) book was released. The CART algorithm is widely used by ML practitioners applying decision tree analysis to this day, and it has many popular software implementations in programming languages commonly used for ML (Pedregosa, Varoquaux, Gramfort, & Michel, 2011; Therneau & Atkinson, 2019). The main innovation this algorithm possesses compared to previous implementations is its automatic pruning procedure. The procedure CART applies uses cross validation to automatically determine the optimal pruned size of the learned tree using hold out data as an estimate for global performance (Breiman, Friedman, Olshen, & Stone, 1984). While this is a standard technique used in ML modeling (James, Witten, Hastie, & Tibshirani, 2013), the automatic performance of this cross validation makes CART particularly powerful. For classification purposes, CART utilizes Gini impurity measure to determine optimal splits. Gini impurity is calculated using Equation 2, where *N* represents the number of classes to be predicted, and p_i be the proportion of instances at a given node of class *i*.

$$I_G = 1 - \sum_{i=1}^{N} p_i^2$$

Equation 2. Gini Impurity Measure. Let N be the number of classes to be predicted, i be the specific class under consideration, and p_i be the proportion of instances at the given node of class i.

After CART was proposed, Quinlan proposed the ID3 algorithm (1986) and an improved version, C4.5, was proposed in 1993 (Ross, 1993). These algorithms diverge from CART only slightly, having just two major differences. Firstly, an entropy-based impurity measure called the gain ratio is used as an impurity measure in classification. Secondly, categorical variables are first split into each of the individual classes and merged again until an optimal split is found.

What should be noted after presenting this historical review is the age of the work. While the following sections will highlight more current research and innovations in the field, the underlying methodologies for applying decision tree splitting procedures are decades old; most of the methods are nearly 30 years old at the time of this writing.

Recent Innovations: Ensemble Methods and Anomaly Detection

Despite being an older methodology, decision trees are still commonly used in practice. While decision trees are considered one of the most interpretable model types, recent innovations, such as ensemble methods, have become increasingly opaque. This will be investigated further in the following paragraphs, but it is important to note that the increased opacity directly conflicts with some of the basic characteristics of the original decision tree models.

In ML, an ensemble model is a form of aggregate model which consists of multiple individual models, usually of the same type. Ensembles have been shown to be more performant than individual models, but they also result in a loss of interpretability due to the resulting need to interpret many models instead of one (Breiman, Random Forests, 2001; Chen & Guestrin, 2016; Breiman, Statistical Modeling: The Two Cultures, 2001; Brodley, Utgoff, & Kibler, 1995; Masegosa, 2020; Lee, Ullah, & Wang, 2020). The following few paragraphs will explore a selection of some of the most prolific tree-based ensemble methods.

Bootstrap AGGregatING (bagging) is one of the first tree-based ensemble methods created (Lee, Ullah, & Wang, 2020). While not necessarily tied to a single model type, bagging is often used with classification trees, as is done in the original paper. Bagged models are constructed by randomly applying bootstrap sampling to the original training data to generate multiple training sets. Individual classifiers are then trained on each of these training sets, with the overall prediction for a new instance being the most predicted class among all model votes. In his paper, Breiman notes that bagging performs particularly well when a learning procedure is unstable, a noted characteristic of trees trained using the CART algorithm (Master, et al., 2017) that refers to the phenomenon where small changes in training data may cause large changes in the resulting model. Breiman further shows that the higher the model instability, the better performance a bagged model will perform; this is attributed to the bootstrapping method used to generate training sets (Lee, Ullah, & Wang, 2020).

Random forests are another ensemble method, but these models are tied specifically to decision trees. Like bagging, random forests also generate ensembles of trees based on bootstrap samples of the training data. However, there are two main differences between the techniques. The first of these is that each tree in a random forest is trained until no further splitting can be made on the data the specific tree is training on. Secondly, the features in the training data used to train each individual tree in a random forest are also randomly selected (Breiman, Random Forests, 2001). This randomized feature selection allows different trees to learn different feature relationships and ensures that each tree is not exploiting the same relationships. These combinations of features of random forests have been shown to improve the generalization of random forest models (Breiman, Random Forests, 2001).

In addition to the previously mentioned ensemble methods, which each take votes from each individual model equally, there are also ensemble methods referred to as boosting (Freund & Schapire, 1997; Chen & Guestrin, 2016; Friedman, 2002). As opposed to bagging methods, which create models in parallel to one another, boosting methods create individual models serially and place emphasis on misclassified entries. Two specific boosting methods, Adaboost and Stochastic Gradient Boosting, will be examined here.

ADAptive BOOSTing (adaboost) is a boosting method presented in 1997 by Freund and Schapire (Freund & Schapire, 1997). The method is not reliant on the use of decision trees, but decision trees are the most widely used base model when applying adaboost. To train an adaboost model using decision trees, the procedure begins by training a "decision stump," a decision tree with only one split. Once the first "stump" has been created, this model is used to make predictions on the entire training set. Weights are then assigned to each instance such that examples with higher error rates from the existing classifiers in the ensemble are weighted higher than those with lower error rates. The next stump is then trained respective to those weights. By using this training procedure, the ensemble is trained in such a way that it self-corrects for previous errors (Freund & Schapire, 1997).

Gradient boosting methods (Friedman, 2002; Chen & Guestrin, 2016) are another popular boosting method. As with adaboost, gradient boosting seeks to develop a set of weak learners that improve upon one another in an iterative manner. Unlike adaboost, this is done by iteratively reducing the residuals, a term referring to the difference between the predicted and actual output values. The residuals are the negative gradient of the mean squared error loss function, so by applying this method in training the model is optimizing over the input feature topology (Friedman, 2002).

Aside from ensemble methods, another area where decision tree technologies have shown tremendous performance is in anomaly detection. Anomaly detection is a broad field with applicability in many industries and problem spaces, and anomaly detection typically involves identifying data instances which vary significantly from expected. These techniques are particularly useful in problems such as identifying fraudulent bank transactions (Thudumu, Branch, Jin, & Singh, 2020).

Anomaly detection, as with other ML problems, does not always involve using a single model. It is commonplace to use other techniques, such as dimensionality reduction (Wold, Esbensen, & Geladi, 1987; Wold, Esbensen, & Geladi, 1987; McInnes, Healy, & Melville, 2018; van der Maaten & Hinton, 2008; Wattenberg, Viégas, & Johnson, 2016) to improve results

(Thudumu, Branch, Jin, & Singh, 2020). Decision trees have found application in this field using a technique called “isolation forests” (Liu, Ting, & Zhou, 2008; Sahand, Kind, & Brunner, 2019). In contrast to a traditional decision tree, isolation forests are used by performing random splits across one variable at a time until either each point is isolated or a predetermined depth is reached. This process is done repeatedly to create an ensemble of isolation trees, hence the term “forest.” The theory underlying isolation trees lies on the idea that anomalies are uncommon and lie further away – in topological space – from non-anomalous instances (Sahand, Kind, & Brunner, 2019). Using this assumption, the depth at which a new point reaches in a single tree is a measure of how anomalous the point is; the smaller the score, the more easily that point is separated from the rest of the data. In a forest, all trees’ scores are aggregated together, typically through arithmetic mean, to determine a point’s overall anomaly score. For example, a case study performed by researchers at RMIT University and CA Labs showed that isolation forests were able to effectively identify anomalous and malicious behavior within enterprise information systems. By analyzing data in log files and using a version of an isolation forest, the researchers were able to develop an anomaly score for different actions users performed. By further cross-referencing this information with items such as the number of anomalous behaviors for a user in each time frame, they were able to successfully identify many malicious events (Sun, Versteeg, Boztas, & Rao, 2016).

A critical review of the available literature identifies an important trend in decision tree research: there has been a lack of major improvements in decision tree methodologies in any modern capacity. The same training and inference procedures that were made popular by Breiman, Friedman, Olshen, and Stone in *Classification and Regression Trees* (1984) are still the ones widely used today. Even some of the newest innovations such as ensemble methods do not significantly alter these methods for creating splits. These methods are innovative instead through the use of multiple models. Isolation forests and other tree-based methods for anomaly detection are innovative in how the resulting models determine their splits, but they represent a form of unsupervised learning rather than a supervised approach, so their use is limited with respect to the scope of this research.

In short, decision trees are reliable, widely-used, and – with respect to technological innovation – stagnant. Little has been done in recent years to improve their underlying methodologies, and this is reflected in the ages of the literature reviewed to this point. The following section will review some alternative model types and theories that have been uncovered through literature review; the section following that will then review an area of innovation which this research will investigate heavily: multivariate splits in classification tree learning.

Alternative Model Types and Theories

There are two major theories in ML and statistical learning that diverge from the theories decision trees fall under. Namely, these two theories are *data modeling* and *uninterpretable models*.

The difference between what has been called the "data modeling culture" and the "algorithmic modeling culture" is perhaps best stated in Breiman's work *Statistical Modeling: The Two Cultures* (2001). In this work, the author elaborates on how his personal experience as an academic and as an industry professional highlighted the difference in theoretical mindsets between the more rigid statistical community and the more practical industry practitioners. As the primary creator of the CART algorithm (Breiman, Friedman, Olshen, & Stone, 1984), Breiman's word on this subject is particularly valuable.

The first of the factions Breiman mentions in his paper is the "data modeling culture," which is dominated largely by the theoretical statisticians (Breiman, Statistical Modeling: The Two Cultures, 2001). The term "data modeling" used here highlights how proponents of this ideology seek to create models which seek to recreate the way the data being modeled is generated. This paradigm is best understood when contrasted with the other ideology Breiman mentions, the "algorithmic modeling" paradigm, of which the author himself states he belongs to. In contrast to data modelers, algorithmic modelers seek to develop mathematically based algorithms which can "learn" simply to predict the outputs of the data creation mechanism. The key differentiator here is that algorithmic modelers seek to build flexible algorithms, which may be interpretable or uninterpretable, that are able to successfully model a wide variety of scenarios without necessarily uncovering the data generation methods.

To illustrate the differences of these two cultures, Breiman uses two process diagrams, presented in Figure 2 and Figure 3, which show how the data modeling culture seeks to replicate the data creation process and the algorithmic modeling culture seeks to circumvent this generation process and generate algorithms which are capable of modeling the outcome. Two examples of data modeling algorithms are linear regression (Stanton, 2001) and logistic regression (Cuartas, et al., 2021). These algorithms require making strict assumptions about the data generation process to be used to their full potential. In contrast, algorithmic modeling algorithms such as decision trees and neural networks require no such assumptions.

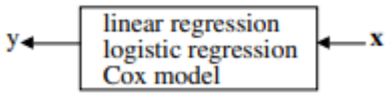

Figure 2. Data Modeling Culture Process Diagram, taken from (Breiman, Statistical Modeling: The Two Cultures, 2001)

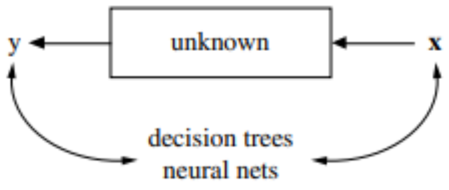

Figure 3. Algorithmic Modeling Culture Process Diagram, taken from (Breiman, Statistical Modeling: The Two Cultures, 2001)

While both decision trees and ANNs fall under the algorithmic modeling paradigm, they vary in a second modeling theory *interpretable* vs. *uninterpretable models*. Decision trees do not seek to replicate the association function mapping input data to output values, but the resulting model is easily understood and can be inspected. In contrast, ANNs create complex linear and nonlinear combinations of input variables; this makes it impossible for humans to reasonably understand the decision process (Barredo Arrieta, et al., 2019; Gilpin, et al., 2018; LeCun, Bottou, Bengio, & Haffner, 1998). This opaque, complicated decision process is usually overparametrized (Allen-Zhu, Li, & Liang, 2020; Du, Zhai, Poczos, & Singh) and can be surprisingly susceptible to adversarial attacks, particularly in the cases of computer vision (Moosavi-Dezfooli, Fawsi, & Frossard, 2017; Cao, et al., 2022). However, adversarial attacks have become increasingly potent in natural language processing (Jia & Liang, 2017) as well as others, highlighting the increasing importance of XAI.

Taking these theories; *data modeling*, *algorithmic modeling*, *interpretable modeling*, and *uninterpretable modeling*; into consideration, the unique placement of decision trees in this theoretical landscape becomes apparent. Decision trees fall under the algorithmic modeling paradigm, implying that these technologies are capable of modeling without making as many assumptions about the underlying data compared to data modeling algorithms such as linear regression. However, decision trees also fall under the paradigm of interpretable models, meaning these techniques provide a significant value by allowing the end users to understand the decision process for these models. This unique combination of characteristics makes decision trees uniquely capable algorithms suited for a wide array of tasks. They are powerful prediction algorithms capable of high levels of flexibility, yet they are highly transparent and understandable even to users with little experience in ML and AI.

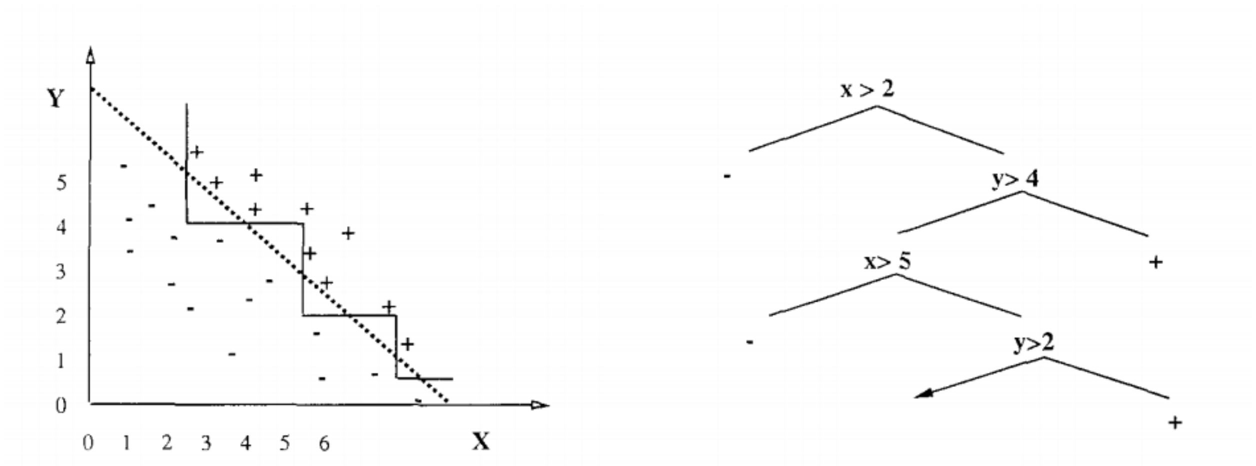

Figure 4. Simple Multivariate Decision Tree, taken from (Brodley, Utgoff, & Kibler, 1995). On the left is a visual representation of the data. The stepwise function illustrates the decision

function learned from a traditional decision tree, illustrated on the right; the dotted line illustrates the optimal multivariate decision function, a single linear split which perfectly separates the data.

Gaps in Splitting Theory: Multivariate Splitting Criteria

While some recent innovations in decision tree methodologies, such as those mentioned previously, have proven to be versatile and powerful prediction methods which have been used extensively by practitioners, there is a noticeable gap in decision tree splitting principles. Specifically, this problem is that traditional decision trees are only capable of making splits across one variable at a time, a term referred to as “axis-parallel” splitting. This limited capability can not only lead to sub-optimal performance, but it can also lead to overcomplicated decision functions in resulting trees which mask the true relationships being learned (Breiman, Friedman, Olshen, & Stone, 1984; Brodley, Utgoff, & Kibler, 1995). Consider Figure 4, which displays this phenomenon and highlights how multivariate splits in a tree can greatly reduce the number of splits needed and thus improve human understanding of the tree. In the figure, the illustration to the left shows the spatial distribution of data on a binary classification task. The solid, stepwise function indicates the decision boundary learned from traditional decision tree learning procedures. The resulting tree is illustrated to the right. By contrast a single multivariate split, represented by the dotted line, perfectly splits the data. This example indicates that a multivariate tree, if it could efficiently learn effective multivariate splits, would more concisely and accurately model the data.

Though there have been some decision trees developed which can make multivariate splits (Breiman, Friedman, Olshen, & Stone, 1984; Brodley, Utgoff, & Kibler, 1995; Kim & Loh, 2001; Fu, Guo, Lin, & Lu, 2010; Murthy, Kasif, & Salzberg, 1994), multivariate decision trees have seen little use outside of academic settings. For this reason, this research identifies multivariate decision trees as a gap in the theory behind the use of decision tree technologies for ML. Due also to the lack of multivariate trees in production scenarios, this work will only review a selection of some of the most innovative and historically relevant multivariate trees which have been presented over time.

In *Classification and Regression Trees*, the same work in which the CART algorithm was presented, the authors present three methods for creating multivariate CART trees in Chapter 5 (Breiman, Friedman, Olshen, & Stone, 1984). The first of these methods involves creating splits of the form found in Equation 3 at every node in the tree, subject to the condition $\sum_m a^2 = 1$. To maximize interpretability by removing some variables at each split, the authors propose a backwards-deletion algorithm to iteratively set some coefficient values to zero based on a variable importance factor they define in the work. While the authors also note that trees resulting from using this method will be more difficult to interpret than traditional decision trees, it should also be noted that the authors make no attempt for the algorithm to favor univariate splits in training using this procedure. By contrast, to make a univariate split using this algorithm, all other variables would have to be removed using the authors' backwards deletion algorithm (Breiman, Friedman, Olshen, & Stone, 1984).

$$\sum_m a_m x_m \leq c$$

Equation 3. Multivariate CART 1. Let m be the specific variable under consideration among M total variables, a_m be the coefficient for variable m, x_m be the mth variable, and c be a value that ranges across all possible values.

$$\bar{x}_{m1,m2} = \frac{1}{m_2 - m_1 + 1} \sum_{m=m_1}^{m_2} x_m \,, m_2 > m_1$$

Equation 4. Multivariate CART 3. Let m, m_1, and m_2 refer to specific variable numbers, x_m refer to the variable corresponding to m, and $\bar{x}_{m1,m2}$ refer to the averaged value of the values from x_{m1} and x_{m2}

The second multivariate split method presented in *Classification and Regression Trees* is one that makes splits based on Boolean combinations of variables. The authors cite this algorithm as uniquely capable in a medical setting, where certain combinations of variables, such as patient symptoms, are highly indicative of disease. Once again, the authors present a stepwise optimization process to approximate the best combination of variables to use in splitting (Breiman, Friedman, Olshen, & Stone, 1984). After further review, this method was not found to be used in other literature sources.

The third method presented by those same authors utilizes an averaging method to perform feature engineering. Additional features are made according to Equation 4. This method adds what the authors call an “intuitive appraisal” of variables, in that new features can be split in ways that indicate patterns across multiple variables at a time (Breiman, Friedman, Olshen, & Stone, 1984).

Outside of the work in *Classification and Regression Trees*, another relatively popular and powerful multivariate classification tree algorithm is Oblique Classifier 1, or OC1 for short (Murthy, Kasif, & Salzberg, 1994). OC1 creates multivariate splits by attempting to optimize a multivariate hyperplane using a combination of deterministic and heuristic strategies. Specifically, the algorithm uses a standard optimization procedure on the current hyperplane to find the local minimum of the loss function; once that optimization is complete, the hyperplane is randomly perturbed, and optimization follows once more. This procedure enables greater coverage over the overall parameter space, ultimately ensuring a better probability of reaching the true optimal hyperplane for the current split (Murthy, Kasif, & Salzberg, 1994). Like the methods presented in *Classification and Regression Trees*, however, this decision tree methodology does not place emphasis on interpretability, as splits are considered across all variables at each point. While it is true the optimization procedure may set the coefficients of individual variables in the hyperplane equal to zero, ultimately this is unlikely.

While the approaches mentioned to this point all involve creating decision-tree-like splits across multiple variables, there are also hybridized models which utilize a combination of decision tree techniques coupled with non-decision-tree techniques, such as one which utilizes support vector machines (SVMs), called the Decision Tree SVM (DTSVM) (Fu, Guo, Lin, & Lu, 2010). To train a DTSVM, first, either a CART or C4.5 decision tree is trained on the input data until stoppage criteria are met. Once this training is complete, SVM models are trained on the data at the leaf nodes of the resulting tree. The authors cite (Vapnik, 1999) as a primary driver for their decision to utilize SVMs in this process, as they note SVMs have a unique capability to solve pattern classification problems in a variety of domains (Fu, Guo, Lin, & Lu, 2010). While the decision tree portion of the DTSVM training procedure does not differ from a traditional decision tree, the method is noted in this review because of the SVM portion of the methodology, since SVMs do perform a form of multivariate "splitting" by creating a multivariate planar boundary across the training features (Cortes & Vapnik, 1995). Further investigation has revealed additional methods which utilize decision trees and SVMs in conjunction with one another across a variety of use cases (Sun, Zou, Fu, Chen, & Wang, 2019).

There have been multiple methods for creating multivariate decision trees explored here, but it is important to note that these methods exist almost solely in academic arenas and have seen extremely low usage outside of these circles. Primarily for this reason, multivariate trees have been identified as an underexplored area of research in decision tree technologies. By combining this knowledge with information and theories from the seminal field of XAI, which will be explored in the following section, the research to be conducted in this study will explore this underutilized area and develop a new type of classification tree capable of making multivariate splits without sacrificing interpretability.

Explainable Artificial Intelligence

Unlike decision trees, XAI is a very new field. This newness is due in part because the current wave of AI models is marked by the use of increasingly opaque decision models (Barredo Arrieta, et al., 2019). ANNs are particularly opaque (Castelvecchi, 2016); for example, the now-famous BERT model developed for natural language processing tasks has over 340M parameters (Devlin, Chang, Lee, & Toutanova, 2018) and uses the extremely mathematically complicated transformer architecture (Vaswani, et al., 2017). While model architecture and training procedure optimization techniques (Liu, et al., 2019) have resulted in similar performance levels from models with as few as 18M parameters (Lan, et al., 2019) and other research has shown that as much as 90% of network parameters may be removable without harming overall performance (Frankle & Carbin, 2018), interpretability of even the most optimized models seems impossible given these methods (Gunning & Aha, 2019).

There are several reasons why interpretable models are needed. Firstly, there is the simple reason that some problem spaces are not suited for opaque systems. These problem spaces require a full understanding of the decision structure of ML models, something that cannot be directly determined when using opaque models (Gunning & Aha, 2019). Secondly, XAI is expected to help with the adoption of AI technologies. In general, humans have been found to be hesitant to adopt technologies that are uninterpretable (Zhu, Liapis, Risi, Bidarra, & Youngblood, 2018), so developing explanatory interfaces or building explanations directly into the models themselves has been identified as a key to fostering more widespread adoption of these technologies (Barredo Arrieta, et al., 2019).

A third reason for creating explainable models would be to ensure the impartiality of decision making (Barredo Arrieta, et al., 2019). Unfortunately, biases in data can be learned and ingrained in the models that train from that data, as with a hiring algorithm developed by Amazon. This model, which has since been decommissioned, was shown to weigh the resumes of women negatively due to biases in the training data – resumes from existing, mostly-male, employees (Kodiyan, 2019). Explainable models may also be robust against adversarial attacks and perturbations (Barredo Arrieta, et al., 2019). Succinctly, there are many reasons for adopting explainable models, ranging from explicit needs for explainability to desires to gain buy-in from stakeholders.

As a result of current modeling paradigms and industry needs, the Defense Advanced Research Projects Agency (DARPA) almost singlehandedly brought together all these needs and created what is now called XAI. DARPA identified a need for a “suite of technologies” which produce more explainable models and enable humans to easily understand, trust, and manage AI systems (Gunning & Aha, 2019) . As stated before, XAI is a new field, and as such the terminology has not fully converged. The next section will provide an overview of important terminology definitions used in this research.

Current Research and Theories

In this section, a deeper review of the theories and methods within the field of XAI is examined, with a focus on understanding current theories and technologies used for explainability.

One of the rising theories in XAI is the idea of *audience* at the center of importance for the explainability of a model (Barredo Arrieta, et al., 2019). Put simply, this theory states that the reason for creating explanations and the level of explanation required for a model are tied to the consumers of the model. Figure 5 shows the importance of audience in XAI, as it highlights both a wide selection of groups who utilize ML models and why each of those groups cares about interpretability and explainability in those models.

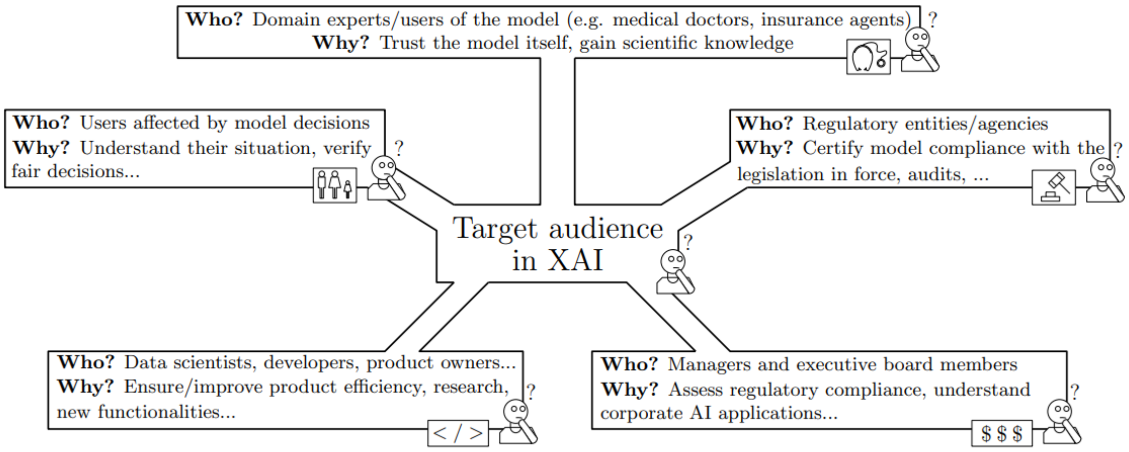
Who? Domain experts/users of the model (e.g. medical doctors, insurance agents)
Why? Trust the model itself, gain scientific knowledge
Who? Users affected by model decisions
Why? Understand their situation, verify fair decisions...
Who? Regulatory entities/agencies
Why? Certify model compliance with the legislation in force, audits, ...
Target audience in XAI
Who? Data scientists, developers, product owners...
Why? Ensure/improve product efficiency, research, new functionalities...
Who? Managers and executive board members
Why? Assess regulatory compliance, understand corporate AI applications...
</>
$ $ $

Figure 5. Showing the importance of audience in XAI. Taken from (Barredo Arrieta, et al., 2019).

This theory of audience at the center of XAI may appear to be a way to prevaricate the issue of developing a complete, unified set of requirements for explainability and interpretability in AI. However, as AI’s prominence grows and becomes more widespread, it reaches an ever-increasing number of people. These people have a variety of backgrounds – and thus requirements for explainability. Referencing Figure 5 once more, the diversity of audience backgrounds and requirements is highlighted well in that illustration, with the reasons for explainability ranging from understanding the implications of using the model to truly understanding the relationships in the data that the model is exploiting.

Further examination of the needs of specific stakeholders and AI users has revealed a further set of specific underlying reasons why explainability are required. These reasons are Trustworthiness, Causality, Transferability, Informativeness, Confidence, Fairness, Accessibility, Interactivity, and Privacy Awareness. The following paragraphs highlight each of these needs.

Establishing trustworthiness is perhaps the primary goal of XAI (Ribeiro, Singh, & Guestrin, 2016). However, trustworthiness of a model is often very hard to quantify, or even define. Clearly, an explainable model would be more trustworthy than one that achieves the same results without any explanations, but a trustworthy model may not require itself to be explainable (Barredo Arrieta, et al., 2019). Perhaps then a model’s trustworthiness is linked to its reliability in the eyes of the audience; one is more likely to deem a model trustworthy if one understands which factors influence when the model is correct and when the model fails. This definition is clearly linked to explainability, as an explanation of a model’s choices provides the audience with information of the model’s inner decision process. However, what the audience is truly looking for is an understanding of the model’s behaviors in terms of output, indicating more on the lines of reliability than explainability. An analogous situation would be a user of an automobile developing an understanding of how the vehicle performs in different weather conditions. Developing operator trust in this situation does not rely on the user having an intimate understanding of the technologies built into the vehicle, but rather experience using the vehicle and experimenting with its capabilities.

Identifying causality and potentially causal relationships is another goal of XAI. Finding relationships among variable which may be indicative of universal truths, rather than correlational relationships, is particularly present in fields such as medicine. While current ML technologies are only capable of finding correlational relationships between variables, and there is the popular adage that correlation does not imply causation, another viewpoint is that "causation involves correlation" (Barredo Arrieta, et al., 2019). Furthermore, research has been done to identify causal relationships in statistics (Pearl, 2022; Liang, 2016)

The third identified reason for XAI, transferability, relates to the applicability of a model on unseen data. Transferability is like trustworthiness in the regard that explainability helps the audience understand when a model can be transferable, but transferability does not necessarily require explainability. For example, transferability is tested when using a standard train-validate-test approach in building a ML model (Vapnik, 1999; James, Witten, Hastie, & Tibshirani, 2013). However, not understanding how a model makes its decisions can lead to incorrect assumptions and undesired results when applying a model to real inference scenarios (Kodiyan, 2019).

The fourth reason for adopting XAI techniques, informativeness, refers simply to the idea that an explainable model should reveal some information about the problem (Barredo Arrieta, et al., 2019). This reason is perhaps the simplest of the reasons addressed here, but it is an important one, particularly when one considers that ML models address simplified versions of the problems they are employed to solve.

Confidence, in terms of a reason for requiring XAI, refers not necessarily to how confident the human is in the model, but how likely the model believes its prediction is true. A simple approach to showing a model's confidence in classification is to output the predicted probabilities of each class, rather than only a single number. Clearly, using the definitions described previously, this is a form of model explanation.

Another reason for XAI, Fairness, refers mainly to building an understanding of the biases that the model itself possesses (Chouldechova, 2017). Fairness can be particularly important when the AI models affect humans and their lives. By building explainability into these models, AI practitioners can be surer that what they are doing does not negatively affect people.

Accessibility is another reason for employing explainability in AI. Accessibility in this context refers to allowing more of the audience, such as end users, to become more involved in the modeling process and even developing models themselves (Chander, Srinivasan, Chelian, Wang, & Uchino, 2018). Clearly, providing users with more-explainable models to use in this context would make it easier for these users to understand and trust the outputs these models provide.

Interactivity is another factor influencing XAI. This one is again tied mostly to the end users of the algorithm and giving them the freedom to look deeper into a model and possibly alter it (Langley, Meadows, Sridharan, & Choi, 2017).

The last reason for employing XAI, privacy awareness, is rarely considered in the literature (Barredo Arrieta, et al., 2019). As seen previously, ML models may learn undesired biases that propagate to decision systems (Kodiyan, 2019) and can be considered a privacy breach. However, privacy breaches can come from the other direction as well, such as if a third party is able to discover the relationships the model learned through explainability techniques. This second method could potentially put the privacy of individuals whose data was used to build the model at risk.

An interesting trend in XAI is the focus on post-hoc explainability techniques. These are techniques which add explainability to opaque models. Some of these techniques are universal and can be applied to any model type, while others are model specific (Barredo Arrieta, et al., 2019). The review presented here gives a brief overview of some of the most popular techniques.

One of the most famous techniques for post-hoc explainability is LIME, or Local Interpretable Model-Agnostic Explanations (Ribeiro, Singh, & Guestrin, 2016). LIME acts by randomly perturbing the inputs to the model and observing how the outputs change. The method then uses this information to develop linear approximations to the model's decision function in the region surrounding the instance under question. It is important to note here that among all literature reviewed, LIME was mentioned quite often and as one of the first algorithms presented in each work – a testament to its importance and its ubiquity in the field of XAI.

Though model agnostic techniques such as LIME exist, many XAI techniques and technologies are model-specific or not wholly generalized. The following paragraphs will explore some techniques which have been developed for specific use cases and model types. This discussion will begin with previously explored techniques such as ensemble methods and will then explore other ML models such as different deep learning architectures.

As explored previously, ensemble methods – particularly tree-based ensemble methods – utilize multiple "weak" models in aggregate to perform prediction on a new data instance. While this technique has been shown to improve predictive capabilities, the added complexity that prediction from tens or hundreds of individual models adds makes true interpretability difficult.

Due to the widespread use of and impressive predictive capabilities of tree ensembles, a variety of techniques have been developed to improve their explainability and make them more suitable for some of the industries and problem spaces previously discussed (Barredo Arrieta, et al., 2019).

One such technique for imposing explainability onto ensemble techniques has been to train a single, simpler model to make the same predictions as the ensemble. One such method involves the creation of a single, simple model taken using sample of data taken from the training data along with the predictions from a trained ensemble method. The simple model, typically a single decision tree, is then trained on that sample and those predictions with hopes to mimic the overall decision structure of the ensemble (Rosenfeld & Richardson, 2019).

Though there are some methods, such as the one mentioned in the preceding paragraph, which seek to simulate the ensemble’s decision process through the use of smaller, more explainable models, many explanation techniques for ensembles rely on feature relevance and feature importance methods. Breiman was among the first to explore the use of these techniques by utilizing a mean decrease in accuracy (MDA) measure which is determined by randomly permuting a single variable (Breiman, Classification and regression trees, 2017). A following study showed that this technique was accurate when applied to real-world problems (Tolomei, Silvestri, Haines, & Lalmas, 2017).

While there are methods for introducing higher levels of explainability into ensemble methods, much of the detected literature focuses on bagging ensembles as opposed to boosting ones, and little has been published in recent years in this area in general (Barredo Arrieta, et al., 2019). It appears instead that much of the literature since the creation of the term “XAI” has instead focused on explainability techniques applied to deep learning, an area which will be discussed in the following paragraphs.

As mentioned previously, ANNs are some of the most powerful ML algorithms that exist today, but that predictive power is offset heavily by their lack of interpretability and explainability. It is perhaps understandable then why much of the field of XAI has been devoted to adding explanatory powers to these models. The review conducted here provides an overview of explanation techniques to two of the most widely used neural network architectures: the multi-layer perceptron and the convolutional neural network.

The multi-layer perceptron is perhaps the prototypical neural network architecture. This architecture includes stacking layers of artificial neurons which perform linear and nonlinear transformations on their respective inputs to solve the requested task. This chain of transformations is what makes interpretation difficult; the relationships learned by the model are often so complex that no human-understandable information can be retrieved from the resulting model.

In response to the growing need of explainability, several techniques have been developed to improve end user understanding of multi-layer perceptrons. One such technique is the DeepRED algorithm, which uses decision trees and rule extraction techniques to build a simplification of the ANN's decision structure (Zilke, Mencia, & Janssen, 2016). Another model simplification technique is presented in (Che, Purushotham, Khemani, & Liu, 2016), where the authors propose a method for creating an interpretable model which mimics the neural network's decision process using gradient boosted trees.

While the techniques mentioned in the previous paragraph add explainability to multi-layered perceptron by using another, simpler model, oftentimes with deeper – and thus more complicated – models, feature importance techniques are used. One such technique is presented in (Montavon, Lapuschkin, Binder, Samek, & Muller, 2017), in which the authors identify a procedure to decompose a network by treating each neuron as an individual object. This results in a deep Taylor decomposition for the network and a greater understanding of the relevance of each feature (Montavon, Lapuschkin, Binder, Samek, & Muller, 2017). Another method for performing feature relevance is DeepLIFT, which measures the differences between an artificial neuron's activation and a reference activation level and uses that difference in calculations to determine the relevance score (Shrikumar, Greenside, Shcherbina, & Kundaje, 2016). However, it should be noted that multiple studies have shown that many of the feature relevance techniques used are not theoretically rigorous and can result in explanations which are not wholly correct (Sundararajan, Taly, & Yan, 2017; Kindermans, et al., 2017).

Convolutional Neural Networks (CNNs), as opposed to other types of ANNs, are perhaps easier to apply explainability techniques to because they are used commonly with computer vision tasks. The actual relationships these kinds of networks learn is extremely complex; multiple convolutional layers are stacked upon one another to learn higher level features at each step. For example, for a facial recognition task, the first layer may serve as edge detection. Subsequent layers will then be able to use this information to identify facial features such as the nose, lips, or eyes. Due to the use of CNNs on primarily visual tasks, many explanation techniques leverage the visual nature of these problems to make human-centric explanations easier (Barredo Arrieta, et al., 2019).

Though previously discussed, LIME (Ribeiro, Singh, & Guestrin, 2016) is a technique which has shown to be particularly effective technique for explaining CNNs for computer vision purposes. By perturbing input pixels and recording how the resulting class probability predictions change, LIME identifies regions of the image which contribute highly toward the predictions the model makes. These regions are then returned to the user as a form of explanation.

Separate from LIME, many CNN explainability techniques involve making alterations to the network architecture. One such method was presented in (Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016), in which the authors add an average pooling layer between the network regions containing CNN layers and fully connected layers. This average pooling layer is used to make predictions on the image, whereby identifying and localizing areas of the input image which contribute highly toward the predicted values. Another similar method is Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju, et al., 2017). For this method, the authors use the gradients of the target values with respect to the final convolutional layer to supply the values for the heatmap used to identify regions of the input image which contribute highly toward the prediction value. Interestingly, the authors show in their work that Grad-CAM produces similar results to other explainability and visualization methods for CNNs, such as occlusion sensitivity, while also simultaneously being much cheaper to compute (Selvaraju, et al., 2017).

In summary, current XAI theories and ideas exist on a wide spectrum ranging from the theoretical aspects of what explainability is to how to apply explanation techniques toward various opaque model types. Still a relatively new field, XAI is reaping the benefits of the pervasiveness of DL and other uninterpretable model types in today's AI landscape. However, as will be discussed in the following section, XAI has largely existed with a one-directional focus. That is, the focus of XAI research has largely been to impose explainability onto uninterpretable models. This research examines another, virtually unexplored area of XAI: increasing performance of already-explainable ML model types without sacrificing high levels of interpretability.

Gaps in Theory and Applicability toward this Study

One notable gap in the field of XAI involves improving inherently interpretable ML techniques and technologies. None of the previous methods described are designed to give greater predictive power to interpretable methods. Instead, either interpretable models are built to help explain more complicated ones or entirely different techniques are used to impose some explainability on complex models such as ensembles or ANNs.

That is not to say that this area of XAI has been completely abandoned, however. There are a few notable algorithms which have been created more recently that focus on bringing more power to interpretable model types. Among these are Additive Trees (Luna, et al., 2019), and Explainable Boosting Machines (Nori, Jenkins, Koch, & Caruana, 2019), the latter of which are based on the GA^{2}M algorithm (Lou, Caruana, Hooker, & Gehrke, 2013).

Additive trees, as the name implies, do utilize tree-based learning procedures to create a powerful interpretable model. They do this by applying gradient boosting techniques at each of the nodes in the tree during training across all the training data. The result is a decision tree system which is more powerful than traditional CART while also retaining interpretability (Luna, et al., 2019). However, this algorithm will not be tested in this study because these models still solely perform univariate splits.

Explainable Boosting Machines, comparatively, use boosting to identify a single decision function for the entire dataset which has a strikingly similar appearance to a linear regression decision function (Lou, Caruana, Hooker, & Gehrke, 2013). Furthermore, interaction terms of up to two terms are also included in this algorithm. While this algorithm does use some of the learning methods of decision trees in training, the decision function learned is very different from a traditional decision tree in that it is a single function instead of a stepwise one. For this reason, this algorithm will not be compared in this study.

There are a few models, typically ones that fall under the category of transparent models, which require very little – often no more than visualization – to be understood by a wide variety of audiences (Barredo Arrieta, et al., 2019). While there are some mathematical concepts which the intended audience should understand at a basic level for all model types, particularly with logistic regression (Mood, 2010), in general models such as linear regression, logistic regression, decision trees, and K Nearest Neighbors (KNN) are among those which are easiest to understand. However, no studies were found which involved adding improved predictive capabilities to any of these model types.

It is at this region of XAI theory that this research is to take place. As noted in the previous review of decision tree literature, an area which has experienced minimal research in decision tree theory is the addition of multivariate splits. Similarly, a review of XAI literature reveals that no significant progress has been made toward developing more powerful interpretable algorithms. Instead of continuing along with current trends of making opaque models more explainable, this research study will examine the use of a new type of multivariate classification tree, built with explainability in mind, to address issues in both sub-fields in AI.

Chapter Summary

In this chapter, literature relevant to the study to be conducted was analyzed. This review was divided into two sections: decision tree theory and XAI theory. In the first section, a history of decision tree methodologies and technologies was presented, followed by an overview of more recent innovations in these technologies, and ending with a narrative concerning the lack of innovation in the area of multivariate decision trees was made. For the analysis in XAI theory, a brief history and overview of the field was presented, followed by an analysis of recent innovations and how the theory has adapted over time as it has become more mature. Lastly, a gap in XAI theory, specifically the lack of research into developing more powerful inherently explainable algorithms, is examined.

The research to be conducted in this work places itself at the intersection of these two fields, namely in the areas identified as gaps in the existing theories. This work seeks to develop a more powerful, inherently explainable decision tree ML algorithm by utilizing multivariate splits in training. By designing, implementing, and testing this new algorithm, a step forward in both decision tree and XAI theory can be made.

CHAPTER 3: METHOD

As shown in previous chapters[1], there has been a significant rise in need for explainability in AI. However, given the focus of developing post-hoc explainability techniques aimed largely at explaining neural networks, there is a remarkable lack of research into developing more powerful inherently explainable models, such as decision trees. After examining decision tree literature, it has become apparent that there has been little innovation in this field for years.

The primary goal of this research is to examine the intersection between XAI and decision tree theories, as this area represents a unique opportunity to develop more-powerful, inherently explainable algorithms. By developing and testing a new form of multivariate classification tree with explainability as a main design consideration, it is hoped that this research will result in the development of a new technology which can be immediately deployed and used in ML situations requiring high levels of interpretability. This technology, named the Linear Regression Classification Tree (LRCT), utilizes a technique for developing multivariate linear and polynomial splits in classification tree training – a capability not seen in any detected literature. An in-depth explanation of this splitting procedure is contained in the following section, followed by an elucidation of the experiments to be conducted to answer the research questions presented in Chapter 1 of this work.

[1] Ibid. p. 36-39

LRCT Splitting Procedure

As indicated by its name, the LRCT splitting procedure utilizes linear regression as a means for identifying multivariate splits. This is done by using one of the feature variables as a dependent variable and fitting the regression model to the best split in that dependent variable across a selection of other variables as they are broken into a grid-like structure.

The general splitting procedure is defined by the following steps:

1. Let n, j, and p, determined by the user, be positive integers referring to the number of bins per variable, the number of variables to be used to create each linear regression model, and the maximum power to raise each of the j variables to, respectively.
2. Find the best single-variable split.
3. For every combination of $j + 1$ variables, do the following:
    a. Let one of the variables, defined further as x_i, be used as an independent variable.
    b. Across each of the other j variables, determine m_j and M_j, which correspond to the minimum and maximum values for that variable for which there are all classes present, respectively.
    c. Across the j variables, divide each of those variables into n evenly sized bins ranging from m_j to M_j, creating a j-dimensional grid.
    d. For each of the grid boxes identified in the step above, calculate the optimal split across x_i for the data in that grid box.
    e. Use the value found in the previous step as a dependent variable value located at the midpoint of the individual grid box.

   f. Using the points created through this process, fit a linear regression model with target values in x_i and independent values across grid point values for the other j variables, up to and including power p.
   g. Use the fit linear model to create a new multivariate feature by utilizing the coefficients from the linear model and subtracting x_i
   h. Determine the best split across this new variable.
4. Repeat steps 3(a)-3(g) on every combination of variables, recording the best split, measured by greatest decrease in weighted impurity score.
5. Choose the best split between the univariate split and the multivariate split.

There were several design considerations when creating the LRCT splitting procedure. First, the procedure must result in effective multivariate splits. For this research to be successful, the LRCT training procedure must be capable of identifying multivariate splits which provide greater informational value than univariate ones. It is expected that the binning technique described above will assist in efficiently identifying multivariate splits by optimizing the split's hyperplane surface.

A second design consideration was the level of explainability in the learned splits. Unlike traditional decision trees which have simple-to-understand univariate splits, multivariate splits can be much more difficult for a human to understand. As explainability is another of the primary concerns for this research, heavy consideration was given to developing a splitting procedure which could be customized to the user's needs. This condition was met by allowing the user to choose both the number of variables to use in creating multivariate splits and the highest polynomial degree to make these splits at. Splits with fewer variables are easier to understand; this is especially true when only two or three variables are involved, as then the split can be visualized. By allowing the user to control this hyperparameter, the concept of *audience* in XAI is utilized in such a way that allows for different levels of explainability for different scenarios.

The third design concept for the LRCT training procedure, which is related to the concept of explainability, is simplicity. If a simpler, univariate split divides the data in a way that results in equal or better performance than multivariate splits, it is logical that the univariate split be preferred. For this reason, the LRCT procedure includes finding the optimal univariate split as well as multivariate. This also effectively means that an LRCT tree can result in only univariate splits, and multivariate splits are only used if they provide performance benefits over their traditional counterpart.

Now that the LRCT algorithm has been described, the experimental design of this research will be explored fully. This research will contain a mixture of experiments on artificial and collected data. The focus on artificial data experiments will be to highlight the exact capabilities of the LRCT training procedure, while the experiments on collected data will provide evidence for the procedure's ability to create practically useful decision trees.

Research Method and Design

This research study is designed to focus on empirical results, rather than focusing solely on the mathematical theory behind the modeling procedures. Following this fundamental basis, this study will include two sets of experiments. The first of these will focus on understanding how the presented LRCT algorithm performs on artificial data. The data in these experiments will be designed to test specific capabilities, particularly ones which the algorithm is specifically designed to be well-suited for. Additionally, in these experiments, a traditional decision tree splitting procedure and an OC1 (Murthy, Kasif, & Salzberg, 1994) splitting procedure will be used as control cases to show how LRCT's performance compares to other decision tree models.

The first experiment in this set will test the simplest possible scenario that LRCT is designed for: performing a single linear split across two feature variables. In this experiment, a dataset will be created with two input features and a binary classification task. A single linear split across these two features will be devised such that the classes are perfectly divided along that boundary. For the LRCT and traditional decision tree splitting procedures, the resulting models will be trained to perform a single split. The OC1 tree will be trained as in the original paper. The performance of each of the models will be investigated in detail.

The second experiment will extend the first experiment by identifying whether the LRCT algorithm can efficiently perform automatic feature selection. To test this, three additional features with no informational value will be added to the dataset from the previous experiment. If the LRCT algorithm behaves as expected, the split this model learns will be identical to the split learned in the previous experiment. Once again, the two control models under consideration will also be trained on the same data, and the performance of each of the models will be analyzed.

The third and fourth experiments conducted will match closely to the previous two, but the split across variables will be a polynomial one, rather than a linear split. The ability for LRCT to make polynomial splits across variables is one of its defining features, as it is something that has been unprecedented in any of the detected literature. The performance analysis in these experiments relative to each of the three models used will be the same as the previous experiments.

Each of the experiments conducted to this point in the research will have tested some of the core capabilities of the LRCT algorithm, but the training and testing data in these experiments is simple in that the expected splits will perfectly classify the data. This phenomenon is rarely – if ever – the norm in "real world" ML modeling scenarios. To further test the LRCT procedure and provide a better estimate of how the procedure performs on less clean data, the previous experiments will be replicated with added noise in the form of data instances of the unexpected class on either side of the optimal decision boundary. These experiments will see noise levels ranging from 10% to 30% in increments of 10%, where a noise level of 20% indicates that 20% of the data will belong to the "incorrect" class respective of the decision boundary. These noise values will only be present in the training data; test data will remain unaltered to measure performance equally among the original experiments and the altered experiments with noise present. The hypothesis is that so long as the general data structure is upheld, LRCT will be robust to this kind of noise.

The last set of experiments on artificial data will test LRCT's abilities on complicated learning tasks. The first of these experiments will involve a binary classification task with the data distributed in a "checkerboard" configuration. This data distribution will test the procedure's ability to identify a series of splits which result in optimal, or a close approximation of optimal, classification. Following this experiment, experiments with generalized, automatically created data structures will be conducted. This last set of experiments, where data generation is not done by hand, are expected to yield the most accurate results short of testing on true, "real-world" data. These experiments will be conducted on a range of model types that extends beyond the additional two model types mentioned previously; these experiments will include KNN, logistic regression, and feedforward neural networks as well as LRCT, OC1, and a traditional decision tree.

After the previous experiments have been completed, the final set of experiments will be conducted on “real world,” collected datasets. These experiments are essential to this study, as they give the best indication as to the practical predictive capabilities of the LRCT training procedure in deployed, production scenarios. These experiments are designed to be the most complete and rigorous experiments conducted on LRCT. By analyzing the results of these experiments, a detailed understanding of the predictive capabilities and limitations of LRCT will be formed.

The data for these experiments will be taken from common, well-known sources. The first dataset will be the ubiquitous *iris dataset* (Fisher, 1936; Anderson, 1935), a dataset containing 50 samples of four feature variables for each of three different species of iris flowers. The goal in this analysis is to identify the species of flower each sample belongs to, given the information collected. Further evaluation and explanation of the dataset will be done in the following chapter.

The second dataset used in this set of experiments will be a medical dataset with the prediction task to identify whether each sample corresponds to a patient with breast cancer (Dua & Graff, 2017). Compared to the other data used in these experiments, this dataset contains a far greater number of features (30), indicating that this learning problem will identify whether LRCT is able to accurately and efficiently perform automatic feature engineering and selection as designed. Further evaluation and explanation of the dataset will be done in the following chapter.

The final dataset to be used in these experiments will be the well-known wine classification dataset (Dua & Graff, 2017). This dataset, which contains three classes of wines to be predicted and 13 predictive features about each wine, will further the empirical knowledge of LRCT's predictive skills gained through these experiments. Further evaluation and explanation of the dataset will be done in the following chapter.

Model Performance and Evaluation Metrics

Model performance and evaluation is a critical piece for this research study. It is imperative that a comprehensive set of performance metrics be collected for each model in each experiment. By properly collecting performance metrics, conclusions made from this research can be validated with minimal disputation; the following list highlights information related to how each model will be trained and evaluated:

1. For each experiment, the total dataset will be divided randomly into three sets. The first set, comprised of 50% of the total data, will be used in training the model. The second set, comprised of 20% of the total data, will be used as a validation set. The remaining 30% of the data will be used as a test set to estimate performance on unseen data.
2. For each model, overall accuracy, confusion matrices, and per-class and weighted average F1 scores will be presented. These numbers will reflect performance on test data only.

By adhering to the data splitting characteristics defined above, it is ensured that all models in each experiment is trained equally. By training on the training set, selecting the most performant model using the validation set, and measuring overall performance using the hold-out test set, each model is held to the same standards without being purposefully trained to perform well on the test set. By utilizing this method, it is ensured that the metrics recorded present an accurate depiction of the generalized performance of the resulting model, rather than overfitting.

Furthermore, each of the metrics presented in the experiments represent specific generalizations about how the models perform. Accuracy metrics present an overview of model performance, but the amount of information presented through accuracy alone is incomplete. By providing confusion matrices and F1 scores as well, a greater picture of model performance can be deduced. Specifically, confusion matrices help show how predictions in binary or multiclass classification problems can be broken down by class. F1 scores help identify the balance between precision and recall for a model. In addition to these metrics, presenting an AUC score in the binary classification tasks identifies how certain each model is when predicting outcomes from new data.

Research Design Effectiveness

By designing the experiments conducted for this dissertation study in the way described above, it is expected that a comprehensive and thorough study of the LRCT algorithm will result. The initial experimentation on contrived, artificial data will provide the foundational information for the study by showcasing how LRCT learns specific kinds of decision boundaries relative to existing models. The second set of experiments on collected data will provide further information into LRCT's algorithmic viability by measuring how the algorithm performs a variety of different tasks which simulate production scenarios.

One shortcoming to this experimental design is the focus on empirical results, rather than theoretical ones. Due to this focus, it is possible that the results of the experiments conducted through this research could misrepresent actual performance. However, by conducting multiple artificial data experiments with varying relationships and by selecting a wide range of collected datasets to use in this study, it is unlikely that the results of these experiments will be unrepresentative. This research design is also chosen specifically because of the focus on utility of the algorithm presented, as it falls under the algorithmic modeling paradigm (Breiman, Statistical Modeling: The Two Cultures, 2001).

Chapter Summary

In this chapter, an overview of the LRCT algorithm and the experiments concerning its performance was presented. By dividing training data into a grid-like structure on a per-variable basis at training time, the LRCT algorithm creates a simplification of the optimal multivariate decision boundary across sets of input features. The algorithm then utilizes linear regression techniques to approximate this true decision boundary, which is hypothesized to provide improved prediction capabilities without hindering model transparency.

Testing will include experimentation using both artificial and collected datasets. By creating a variety of situations in the artificial data experiments, the performance of LRCT compared to a selection of other model types can be carefully measured across a variety of relationships within the input data. Furthermore, by utilizing collected, “real-world” data in multiple experiments as well, a greater understanding can be developed with respect to LRCT’s performance in deployed scenarios.

CHAPTER 4: RESULTS

This chapter presents the results of the experiments conducted as part of this study and as outlined in the previous chapter. For each of the experiments conducted, an overview of the data will first be presented. After the data has been presented, each of the models used in the experiment and any hyperparameters specific to each model will be discussed. Finally, a presentation of the results of each model in the experiment will be presented. Conclusions and discussions related to the experimental results will be withheld until the following chapter.

Experiment 1: Two Variables, Single Linear Split

The first experiment conducted in this study focused on the simplest possible scenario for which the LRCT model was developed to outperform traditional decision tree models: a single linear split across two variables which perfectly classifies a binary classification task. For this task, one thousand instances of two features were randomly selected across the uniform distribution in the interval $(0,1)$. These features were then expanded via multiplication such that the first variable could have a maximum value of 14 and the second variable could have a maximum value of 8. The binary target was then created via the decision function in Equation 5, where i refers to each individual instance and X_0 and X_1 correspond to the first and second features, respectively.

$$Y_i = \begin{cases} 1\ if X_{1,i} > 10 - X_{0,i} \\ 0\ otherwise \end{cases}$$

Equation 5. Experiment 1 Decision Function. Let i refer to an individual instance, X_0 and X_1 refer to the first and second feature variables, respectively, and Y_i refer to the target class for the instance under consideration.

After data generation, 50% of the data was reserved for training, 20% of the data was reserved for validation, and the remaining 30% of the data was used as test data. Visualizing the training data yielded in Figure 6, which clearly shows the underlying relationship between the two classes.

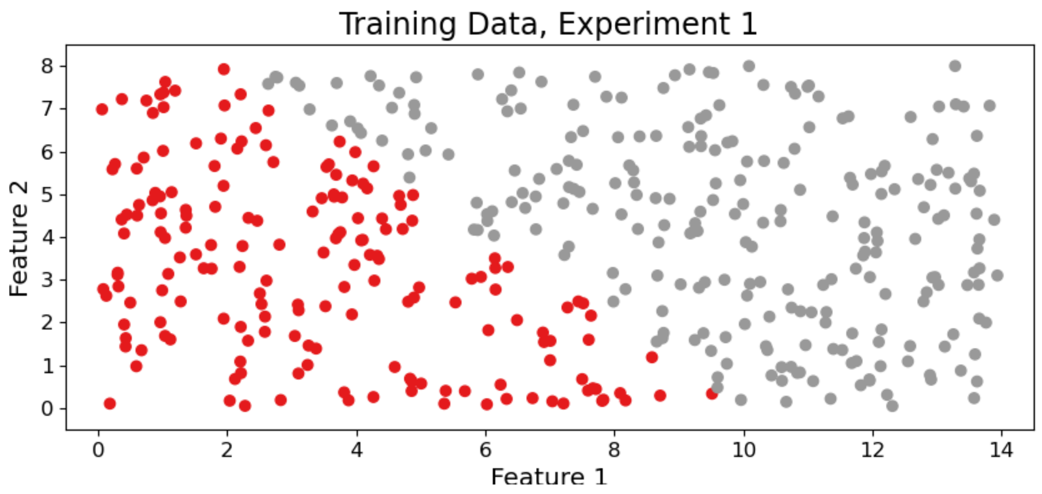

Figure 6. Experiment 1 Training Data. Red dots represent instances belonging to class 0, and grey dots represent instances belonging to class 1.

For this experiment, due to the known underlying structure of the data and the desire to test LRCT's ability to learn such relationships succinctly, both the LRCT model and the standard decision tree model were trained only to perform a single split. Furthermore, LRCT was only allowed to learn linear decision boundaries; the OC1 tree was allowed to train as in the original paper. The results of this experiment are present in Table 1, and the decision boundary learned by the LRCT tree is presented in Figure 7.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
LRCT	0.997	126	1	173	0	1.0	1.0	1.0
CART	0.823	122	49	125	4	0.82	0.83	0.82
OC1	1.0	126	0	174	0	1.0	1.0	1.0

Table 1. Experiment 1 Results

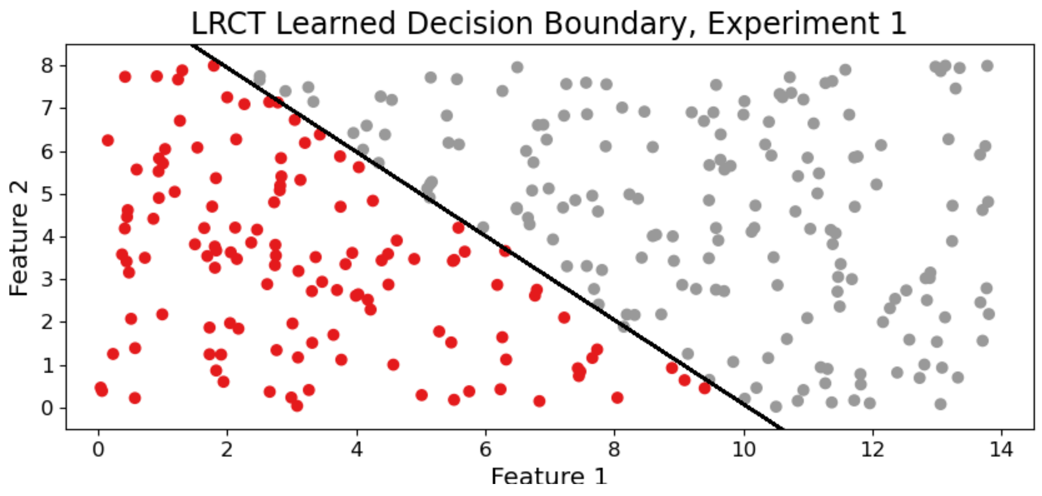

Figure 7. Experiment 1 LRCT Learned Decision Boundary. Red dots refer to instances belonging to class 0, and grey dots refer to instances belonging to class 1. The black line indicates the decision boundary learned by the LRCT tree.

Clearly, both LRCT and OC1 drastically outperform the CART decision tree with a single linear split. However, it is also clear that the decision boundary learned by the LRCT algorithm very closely matches the true function defined in Equation 5. This result gives an initial indication that the LRCT, under extremely simple circumstances, performs as expected and can identify multidimensional splits efficiently.

Experiment 2: Five Variables, Single Linear Split

The second experiment conducted in this study expended the test conducted in the first experiment by adding three additional variables to the dataset, each with no informational value. Once again, one thousand data instances were created, each now with five feature variables. The first two features once again were expanded to exist within the (0, 14) and (0, 8), respectively, while the third through fifth variables were expanded to belong to the ranges (-6, 0), (0, 3), and (-5, 0), respectively. The true decision function was determined again by Equation 5. The same splitting percentages for training, validation, and testing data as previously were observed. The training data for this experiment can be viewed in Figure 8; the careful observer will notice a slight difference between the exact location of the points in this figure and in Figure 6. This difference is due to the random data generation function used.

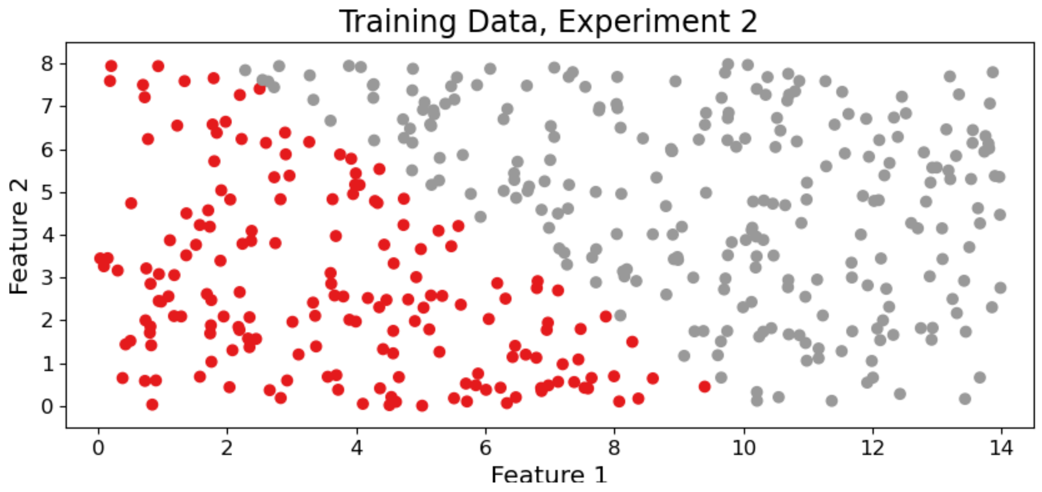

Figure 8. Experiment 2 Training Data. Red dots represent instances belonging to class 0, and grey dots represent instances belonging to class 1.

Again, due to the known underlying structure of the data, for this experiment the LRCT model and the traditional classification tree model were allowed only to train to a depth of one. The LRCT tree was allowed to only create linear splits across two variables, as before. OC1 was trained as in the original paper. The results of the experiment are presented in Table 2, and the decision boundary learned by the LRCT tree is presented in Table 2.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
LRCT	0.98	125	6	169	0	0.98	0.98	0.98
CART	0.823	111	39	136	14	0.81	0.84	0.82
OC1	0.876	109	21	154	16	0.85	0.89	0.88

Table 2. Experiment 2 Results

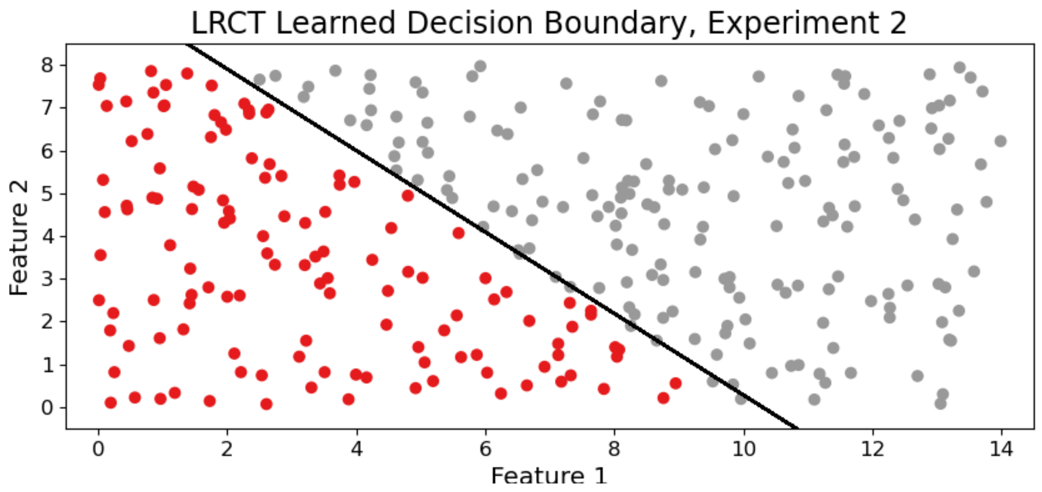

Figure 9. Experiment 2 LRCT Learned Decision Boundary. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1. The black line indicates the decision boundary learned by the LRCT tree.

The results of this experiment highlight a unique capability of the LRCT algorithm: the automatic feature selection capabilities it has. By searching through pairs of input variables instead of all input variables at once, the LRCT algorithm was able to efficiently identify an extremely similar decision boundary as in Experiment 1. Note the small decline in performance in this experiment as opposed to the prior one is due to the nature of random data generation – there were simply more data points generated near the decision boundary for this experiment than in the previous one.

Compared to the LRCT algorithm, OC1 sees a marked decrease in performance compared to the previous experiment. This decrease in performance is likely due to OC1's inherent use of all variables at each split instead of selecting a subset of variables to perform splitting on. CART has virtually the same performance in this experiment when compared to the previous one due to its univariate splitting characteristic.

Experiment 3: Two Variables, Single Polynomial Split

The third experiment run in this study follows the logic of the first experiment, but this time instead of a linear split across two variables, the optimal decision boundary is a polynomial split across two variables. For this experiment, five thousand instances of two variables were randomly created using a uniform distribution in the interval (0, 1). These features were then expanded via multiplication so that the first feature's interval was changed to (0, 5) and the second feature's interval was changed to (0, 20). The binary target was then created via the decision function in Equation 6, where i refers to each individual instance and X_0 and X_1 correspond to the first and second features, respectively. Once again, 50% of the data was reserved for training data, 20% was reserved for validation, and the final 30% was reserved for testing. A visualization of the training data for this experiment can be found in Figure 10.

$$Y_i = \begin{cases} 1 \; if \; X_{1,i} > 2 + X_{0,i} - X_{0,i}^2 + \dfrac{X_{0,i}^3}{2} \\ 0 \; otherwise \end{cases}$$

Equation 6. Experiment 3 Decision Function. Let i refer to an individual instance, X_0 and X_1 refer to the first and second feature variables, respectively, and Y_i refer to the target class for the instance under consideration.

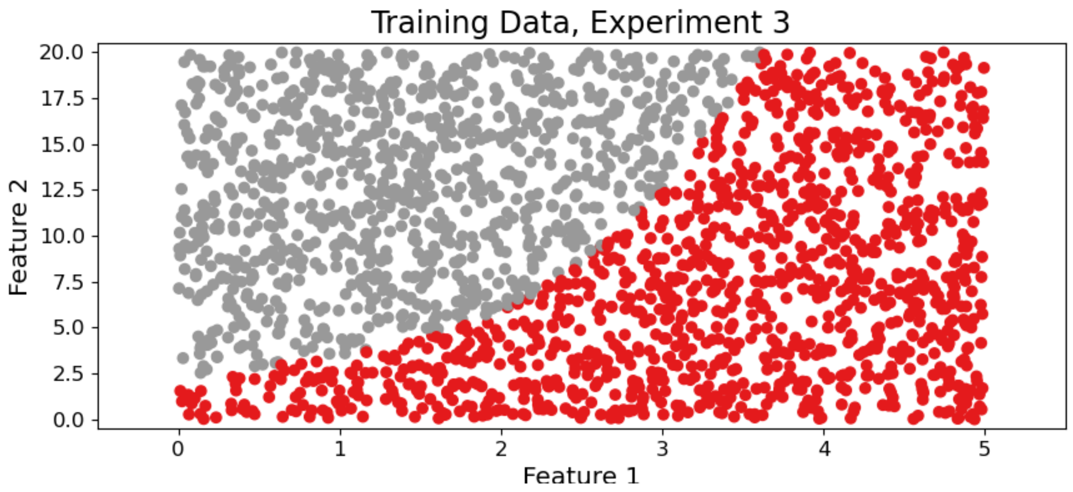

Figure 10. Experiment 3 Training Data. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1.

As with the previous two experiments, in this experiment both the LRCT algorithm and the traditional decision tree algorithm were allowed to make one split. To test whether the LRCT algorithm could learn the polynomial decision boundary present in this scenario, however, the LRCT algorithm was allowed to learn up to third-degree polynomial relationships between considered variables. OC1 was trained as in the original paper. The results of this experiment can be found in Table 3, and the decision boundary learned by the LRCT and presented on the test data can be found in Figure 11.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
LRCT	0.986	810	2	669	19	0.99	0.98	0.99
CART	0.805	576	39	632	253	0.80	0.80	0.80
OC1	0.994	824	4	667	5	0.99	0.99	0.99

Table 3. Experiment 3 Results

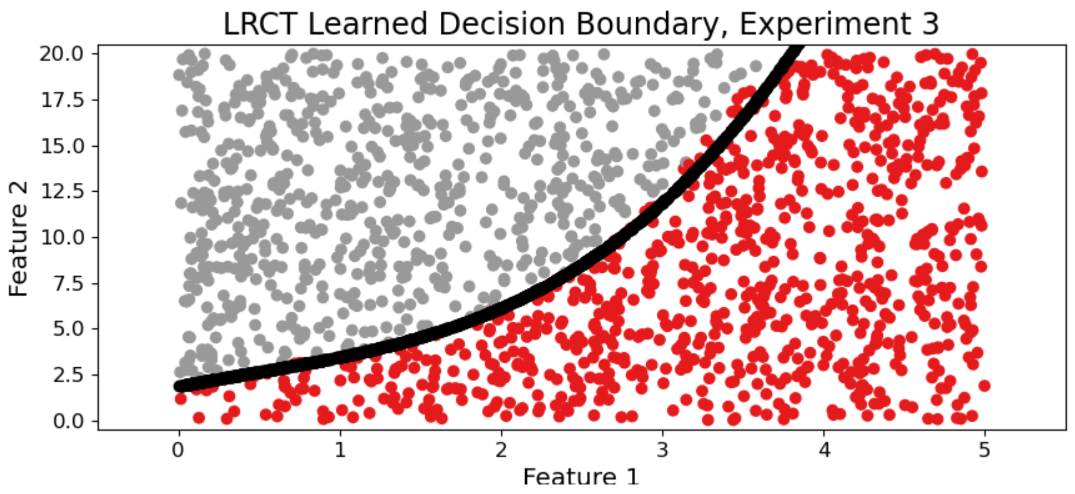

Figure 11. Experiment 3 LRCT Learned Decision Boundary. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1. The black line indicates the decision boundary learned by the LRCT tree.

Once again, with only two variables under consideration, similar performance between the LRCT algorithm and the OC1 algorithm is seen. However, once again the LRCT algorithm was only allowed to make a single polynomial split, whereas the OC1 algorithm was trained until the exit criteria in the original paper were met. This indicates that the LRCT algorithm was able to quickly and efficiently identify the multivariate decision boundary present in this dataset, thus enhancing its explainability and interpretability.

One important factor to note from this experiment is that the decision boundary learned by the LRCT algorithm is not as steep as the true decision boundary, as visible toward the right-hand side of the black line in Figure 11. It is easy to see that the black decision curve falls under the true decision boundary as "Feature 1" increases. This is likely due to the mathematical nature of LRCT's splitting criteria and how the multivariate regression line is created. As the number of bins and the amount of data increases, it is expected that LRCT's polynomial splits will converge to the true decision boundary.

Experiment 4: Five Variables, Single Polynomial Split

Just as the second experiment in this study tested the LRCT algorithm's ability to automatically identify features and multivariate splits of importance, the fourth experiment in this study tests the same characteristic; the only difference is that this experiment mirrored the polynomial decision boundary present in the third experiment. For this experiment, once again five thousand instances were created, with the first and second of these variables being expanded to fill the intervals (0, 5) and (0, 20), respectively. The remaining three variables were manipulated to exist in the intervals (-3, 0), (0, 17), and (-8, 0), respectively. The decision function was the same as the previous experiment as well, recorded in Equation 6. The data partitioning proportions and model hyperparameters were equivalent to the ones used in

Experiment 3. The results of each model in the experiment are presented in Table 4, and the decision boundary learned by the LRCT model is presented in Figure 12.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
LRCT	0.998	808	1	689	2	1.0	1.0	1.0
CART	0.815	583	50	640	227	0.81	0.82	0.81
OC1	0.941	759	37	653	51	0.95	0.94	0.94

Table 4. Experiment 4 Results

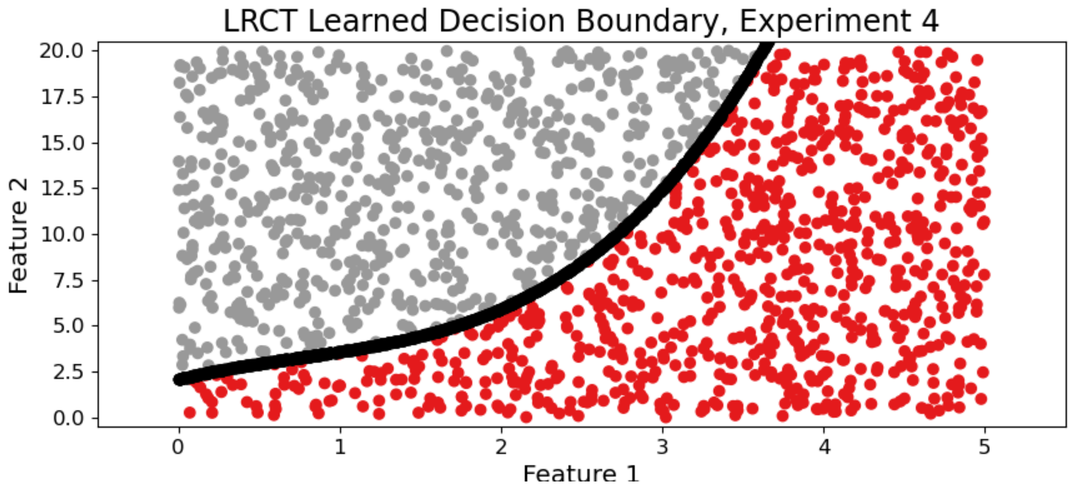

Figure 12. Experiment 4 LRCT Learned Decision Boundary. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1. The black line indicates the decision boundary learned by the LRCT tree.

In this experiment, a similar phenomenon to that present in Experiment 2 is seen. Namely, the LRCT and CART algorithms perform almost identically as in the two-variable experiment (Experiment 3), but the OC1 model sees a decrease in performance compared to that experiment. Once again, it is believed that these performance characteristics are due to LRCT's ability, and OC1's inability, to identify a subset of variables for each split.

Experiment 5: Two Variables, Single Linear Split, Noise

Experiment 5 is the first experiment designed to identify whether LRCT can efficiently identify multivariate boundaries with "noise" present. In all previous experiments, the optimal decision boundary perfectly separated all instances in all datasets – the training data, the validation data, and the test data. However, this is rarely the case in practice when utilizing ML. To better understand LRCT's abilities in more realistic scenarios, duplicates of the previous four experiments were run with various proportions of randomly selected instances labeled "incorrectly."

For Experiment 5, the data generation and optimal decision boundary were both created to be identical to Experiment 1; note that the decision boundary is presented in Equation 5. However, for this experiment, three separate runs were conducted. In the first run, 10% of the training data[2] is selected at random to have its class label switched. The following two runs increase this percentage to 20% and 30%, respectively. A visualization presenting the training data with a noise level of 0.2[3] is presented in Figure 13.

[2] This 10% indicates a noise level of 0.1.
[3] A noise level of 0.2 indicates that 20% of the training data has its label switched.

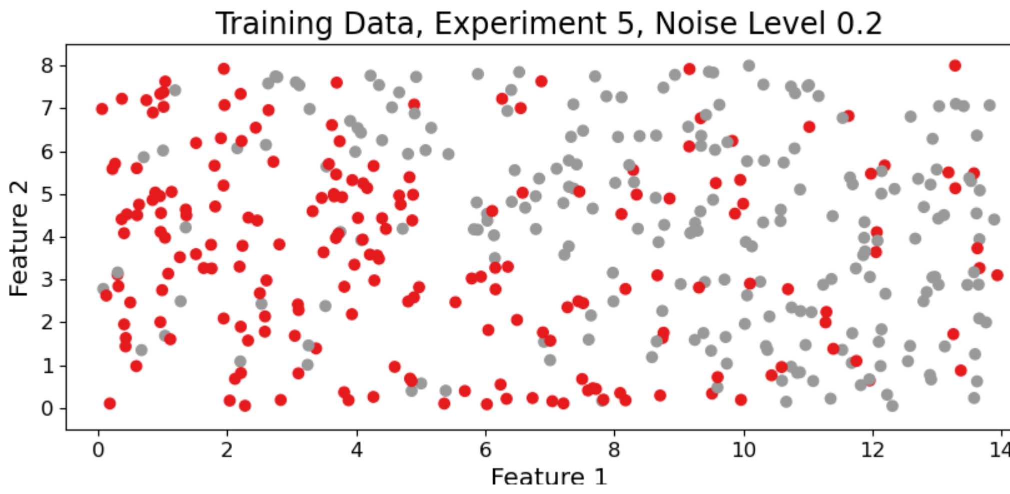

Figure 13. Training data for Experiment 5 with noise level 0.2. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1.

In contrast to the random noise applied to the training data, however, the validation and test data were not altered in these experiments. These data were unaltered to ensure an even comparison between this experiment and Experiment 1. Results for this experiment can be found in Table 5.

Noise Level	*Model Type*	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
0.1	LRCT	0.867	108	22	152	18	0.84	0.88	0.87
	CART	0.867	108	22	152	18	0.84	0.88	0.87
	OC1	0.9	119	23	151	7	0.89	0.91	0.9
0.2	LRCT	0.87	99	12	162	27	0.84	0.89	0.87
	CART	0.87	99	12	162	27	0.84	0.89	0.87
	OC1	0.773	97	39	135	29	0.74	0.8	0.77
0.3	LRCT	0.783	123	62	112	3	0.79	0.78	0.78
	CART	0.783	123	62	112	3	0.79	0.78	0.78
	OC1	0.67	92	65	109	34	0.65	0.69	0.67

Table 5. Experiment 5 Results

The results of this experiment are interesting in that they reveal two characteristics of the LRCT algorithm. The first of these is that if a multivariate split cannot be found such that the performance of that split exceeds the performance of the "best" univariate split, the univariate split is used. Put more simply, LRCT reverts to using CART splits if no better multivariate split can be found. Note that in each of the sub experiments presented above, LRCT's performance is equal to that of the CART decision tree; this equivalence of performance indicates that due to the noise present in this experiment, LRCT could not identify the optimal multivariate decision boundary. Despite LRCT being unable to identify the optimal multivariate decision boundary in this experiment, it appears that the OC1 algorithm had troubles with this as well.

Experiment 6: Five Variables, Single Linear Split, Noise

Just as Experiment 5 replicated the scenario in Experiment 1 with added noise, Experiment 6 replicated the scenario in Experiment 2 with added noise. As such, the data generation techniques used in this experiment were the same as in Experiment 2, including the class labels denoted by Equation 5. Once again, noise levels of 0.1, 0.2, and 0.3 were imposed on the training data. Results of this experiment are presented in Table 6. Experiment 6 Results

Noise Level	*Model Type*	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
0.1	LRCT	0.84	91	14	161	34	0.79	0.87	0.84
	CART	0.84	91	14	161	34	0.79	0.87	0.84
	OC1	0.81	100	32	143	25	0.78	0.83	0.81
0.2	LRCT	0.737	80	34	141	45	0.67	0.78	0.73
	CART	0.823	111	39	136	14	0.81	0.84	0.82
	OC1	0.747	93	44	131	32	0.71	0.78	0.75
0.3	LRCT	0.85	86	6	169	39	0.79	0.88	0.85
	CART	0.85	86	6	169	39	0.79	0.88	0.85
	OC1	0.653	76	55	120	49	0.59	0.7	0.65

Table 6. Experiment 6 Results

In this experiment, the same phenomenon present in Experiment 5 is seen for noise levels of 0.1 and 0.3. Namely, the LRCT algorithm was unable to discover the underlying multivariate decision boundary, and thus it reverted to the same univariate boundaries discovered by the

CART tree with those noise levels. Interestingly, however, the LRCT algorithm did find a multivariate split for noise level 0.2. While that split may have led to better performance on the training and validation data, on the test data this split led the model to perform worse than both the CART and OC1 models, indicating there is a possibility for the LRCT algorithm to overfit and draw incorrect conclusions in training. Additionally, unlike the previous experiment, the OC1 model did not outperform both the LRCT and CART models for any of the noise levels in this experiment, indicating that the algorithm has the same difficulty with performing feature selection as in Experiment 2.

Experiment 7: Two Variables, Single Polynomial Split, Noise

Experiment 7 continues the trend of introducing noise to previous experiments. This time, noise was added to the procedures done in Experiment 3. As in Experiment 3, five thousand instances of two variables were generated, each drawn from the uniform distribution in the interval (0, 1). Just as previously, the first variable was expanded via multiplication to reside in the interval (0, 5), and the second was expanded similarly to reside in the interval (0, 20). The binary class label was created by using Equation 6. This time, however, three different levels of noise were added to the training data: 0.1, 0.2, and 0.3. These noise levels are the same as the ones used in the previous experiments containing noise. Once the training data was manipulated, an LRCT model trained to a depth of one, a traditional decision tree model trained to a depth of one, and an OC1 tree trained as in the original paper were all trained using the training and validation data. Metrics on test data, which was not imbued with noise, are presented in Table 7.

Noise Level	*Model Type*	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
0.1	LRCT	0.805	576	39	632	253	0.8	0.81	0.8
	CART	0.805	576	39	632	253	0.8	0.81	0.8
	OC1	0.89	756	92	579	73	0.9	0.88	0.89
0.2	LRCT	0.823	649	85	586	180	0.83	0.82	0.82
	CART	0.823	649	85	586	180	0.83	0.82	0.82
	OC1	0.81	679	135	536	150	0.83	0.79	0.81
0.3	LRCT	0.806	577	39	632	252	0.80	0.81	0.80
	CART	0.806	577	39	632	252	0.80	0.81	0.80
	OC1	0.664	564	239	432	265	0.69	0.63	0.66

Table 7. Experiment 7 Results

The results of this experiment show similar results compared to those in Experiment 5 with the exception that for noise level 0.1 the OC1 algorithm was able to learn the contrived decision boundary very well. However, OC1's performance degrades somewhat when the noise level is raised to 0.2, and significantly when the noise level is raised to 0.3. Additionally, the LRCT algorithm reverted to univariate splits for all noise levels in this experiment, which is why its performance is equal to that of the CART models for each noise level.

Experiment 8: Five Variables, Single Polynomial Split, Noise

Experiment 8 was the final experiment which replicated previous experiments but included noise present in the training data. The data generation for this experiment was completed replicated from Experiment 4, where five variables were created, and a polynomial split consistent with Equation 6 was used to define the decision boundaries between classes. For each of the separate three sub-experiments within this experiment, noise was added to the training data. As with the previous experiments, noise levels of 0.1, 0.2, and 0.3 were used. LRCT, CART, and OC1 models were then trained on the training data as done previously. The results of these models on test data, which contained no noise, is present in Table 8.

Noise Level	*Model Type*	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
0.1	LRCT	0.815	583	50	640	227	0.81	0.82	0.81
	CART	0.815	583	50	640	227	0.81	0.82	0.81
	OC1	0.839	717	148	542	93	0.86	0.82	0.84
0.2	LRCT	0.972	806	38	652	4	0.97	0.97	0.97
	CART	0.815	583	50	640	227	0.81	0.82	0.81
	OC1	0.747	631	200	490	179	0.77	0.72	0.75
0.3	LRCT	0.962	783	30	660	27	0.96	0.96	0.96
	CART	0.821	646	104	586	164	0.83	0.81	0.82
	OC1	0.675	567	244	446	243	0.70	0.64	0.68

Table 8. Experiment 8 Results

Contrary to some of the previous experiments, in this experiment the LRCT algorithm was able to effectively learn the created decision boundary. With the noise level of 0.1, clearly the LRCT and CART trees identified the same split, as their performances were identical. However, for noise levels of 0.2 and 0.3, the LRCT trees were able to effectively identify the multivariate relationships present in the data, obtaining over 95% accuracy and weighted F1 scores of greater than 0.95 on both test datasets. Similar again to previous experiments, the OC1 tree's performance decreased drastically as noise levels were increased, while the CART and LRCT trees were able to maintain similar levels of performance across the noise levels.

Experiment 9: Checkerboard Configuration

Compared to the previous eight experiments, Experiment 9 differs remarkably in terms of modeling methodology. Instead of training only LRCT, OC1, and CART models in this experiment, several additional model types were also trained. These additional models were KNN, logistic regression, and a fully connected neural network, meaning a total of 6 model types were tested, each with varying levels of explainability and expected predictive power.

Additionally, compared to the previous experiments, Experiment 9 featured a much more complicated decision boundary. Instead of a binary task with a decision boundary defined by a single function of polynomial degree, the target variable in this experiment segments off squares of side length in two-dimensional space, leaving the "checkerboard" configuration seen in Figure 14. The data generation procedure required generating two random variables, each independently sampled from the uniform distribution in the interval (0, 10). The target variable was then created according to the configuration mentioned previously. Once more, 30% of the data was reserved for test data, and 40% of the remaining data was reserved for validation data.

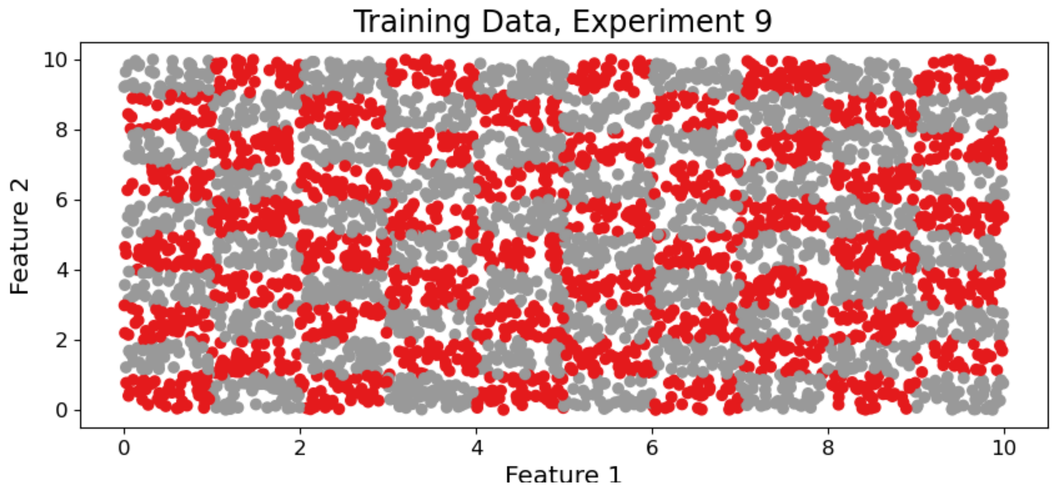

Figure 14. Experiment 9 Training Data. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1.

This experiment also utilized performed a grid search across various hyperparameters for the CART, KNN, logistic regression, and LRCT models. The neural network was trained for 1,000 epochs using validation data accuracy as the performance measure to indicate better or worse performance; only the best model weights according to this metric were used in the end. The results on the test data for this experiment are presented in Table 9.

The results of this experiment show the relative difficulty for the various kinds of models to identify the checkerboard decision boundary present in this data. As KNN predicts based on proximity, it is understandable why KNN achieves high performance on this data. Similarly, LR implicitly requires certain assumptions about the data's distribution be met. These assumptions are not met in this data, thus the extremely poor performance of the LR model results. The CART, neural network, and OC1 models perform similarly on this problem, with the LRCT model's performance lagging further behind the rest of the models except for the LR model. A possible implication of this experiment as it relates to understanding the performance of the

LRCT algorithm is that LRCT likely performs better in learning scenarios where there are smaller numbers of clusters of each class. As this experiment contains multiple clusters for each of the two classes, the underlying logic behind the LRCT algorithm cannot handle these different clusters appropriately, as it is implicitly designed to handle all training data of a single class as a single group.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
OC1	0.797	1215	315	1175	295	0.80	0.79	0.80
CART	0.785	921	57	1433	589	0.74	0.82	0.78
KNN	0.932	1409	102	1388	101	0.93	0.93	0.93
Logistic Regression	0.497	0	0	1490	1510	0.00	0.66	0.33
Neural Network	0.817	1232	270	1220	278	0.82	0.82	0.82
LRCT	0.699	1214	606	884	296	0.73	0.66	0.70

Table 9. Experiment 9 Results

Experiment 10: Random Binary Classification

Experiment 10 follows a similar experimental style as Experiment 9[4] in that it is designed to more closely mimic the training experimental style ML practitioners use when building models. However, with this experiment, the data generation procedure was completely automated, with only a few parameters chosen to guide the way the data was generated.

For this experiment, the scikit-learn make_classification function (Pedregosa, Varoquaux, Gramfort, & Michel, 2011) was used to create a dataset containing 10,000 instances of two features for a binary classification task. Each of the two features were sampled from the standard normal distribution with standard deviation 1, and two clusters were created for each class. Data was divided into training, validation, and testing data as with previous experiments. A visualization of the training data for this experiment is presented in Figure 15.

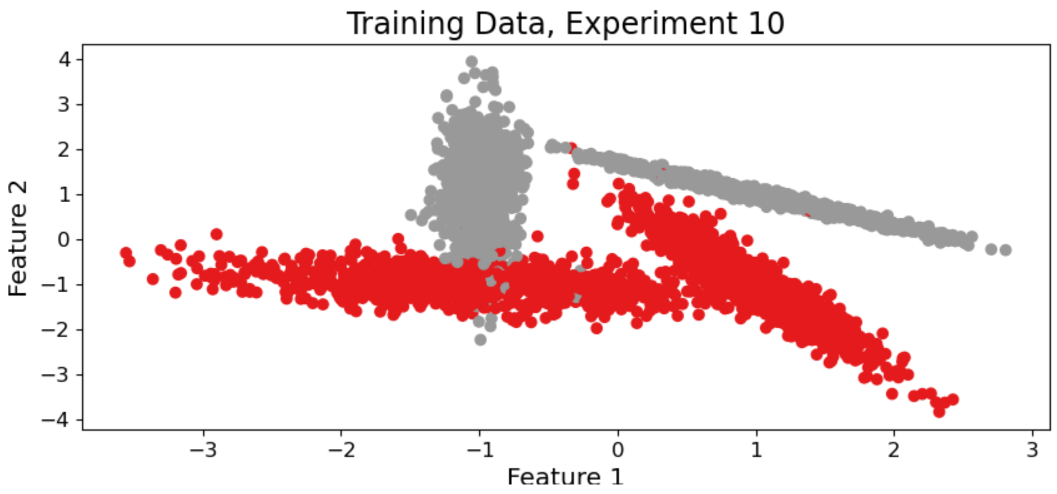

Figure 15. Experiment 10 Training Data. Red dots correspond to instances belonging to class 0, and grey dots correspond to instances belonging to class 1.

[4]All remaining experiments follow this style.

The same model types were used as with the previous experiment, and a hyperparameter grid search procedure was utilized for the CART, KNN, Logistic Regression, and LRCT models. The neural network model was once more trained for 1,000 epochs with validation data accuracy as the measure for model performance. The results for Experiment 10 are presented in Table 10.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
OC1	0.970	1445	44	1466	45	0.97	0.97	0.97
CART	0.983	1469	31	1479	21	0.98	0.98	0.98
KNN	0.981	1470	38	1472	20	0.98	0.98	0.98
Logistic Regression	0.938	1390	85	1425	100	0.94	0.94	0.94
Neural Network	0.982	1470	33	1477	20	0.98	0.98	0.98
LRCT	0.979	1467	39	1471	23	0.98	0.98	0.98

Table 10. Experiment 10 Results

Clearly, for this experiment, all the models under consideration were able to learn an efficient representation of the decision function for this problem. In this regard, these results show that the LRCT algorithm can achieve comparable performance on realistic modeling scenarios compared to other model types. Interestingly, however, the LRCT model did not perform as well as the CART model in this experiment, indicating that the LRCT model identified multivariate splits in the learning process which did not generalize to the test data.

The results of this experiment further support the conclusions made after analyzing the previous experiment's results in that multiple clusters of a single class are not as easily learned by the LRCT algorithm as single clusters. As the LRCT algorithm inherently assumes that the data belonging to each class can be treated as a universal group instead of multiple clusters or groups, this feature is likely an underlying reason why for this experiment and the past experiment the LRCT algorithm's performance have been worse than some of the other models under consideration. It is important to note here, however, that for this experiment LRCT does not suffer much in terms of performance due to the closeness of each of the clusters present in this data.

Experiment 11: Random Multiclass Classification

Experiment 11 is similar to Experiment 10 in that the data generation procedure was once again completely automated. For this experiment, the same scikit-learn make_classification function (Pedregosa, Varoquaux, Gramfort, & Michel, 2011) was utilized to create the data, but for this experiment 10,000 total instances of 12 features were created. To increase the complexity of this experiment, only six of those features were created to have informational value, four of the features were created to be linear combinations of the six informative features, and two additional features were created to be redundant copies of two informative features. This learning problem featured four classes instead of a binary classification problem, and each of the classes were grouped into three clusters.

Once more, data was split into training, validation, and test sets as all previous experiments. Each of the models were trained as previously, including grid searches for finding most performant parameters and number of epochs for the neural network model. Performance metrics for each of the models is presented in the following tables.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*	*Class 3*
Class 0	573	55	49	36
Class 1	61	549	81	70
Class 2	40	79	580	45
Class 3	50	64	73	595

Table 11. OC1 Confusion Matrix, Experiment 11.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*	*Class 3*
Class 0	571	72	29	41
Class 1	54	561	53	93
Class 2	64	76	529	75
Class 3	49	77	53	603

Table 12. CART Confusion Matrix, Experiment 11.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*	*Class 3*
Class 0	622	32	36	23
Class 1	43	622	39	57
Class 2	34	66	599	45
Class 3	32	45	27	678

Table 13. KNN Confusion Matrix, Experiment 11.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*	*Class 3*
Class 0	466	135	86	26
Class 1	124	322	128	187
Class 2	135	71	483	55
Class 3	69	192	53	468

Table 14. LR Confusion Matrix, Experiment 11.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*	*Class 3*
Class 0	640	25	27	21
Class 1	44	648	39	30
Class 2	30	36	659	19
Class 3	26	31	32	693

Table 15. Neural Network Confusion Matrix, Experiment 11.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*	*Class 3*
Class 0	590	70	39	14
Class 1	56	621	32	52
Class 2	41	72	581	50
Class 3	37	77	31	637

Table 16. LRCT Confusion Matrix, Experiment 11.

Model Type	*Overall Accuracy*	*Class 0 F1*	*Class 1 F1*	*Class 2 F1*	*Class 3 F1*	*Overall F1*
OC1	0.766	0.80	0.73	0.76	0.78	0.77
CART	0.755	0.79	0.73	0.75	0.76	0.75
KNN	0.840	0.86	0.82	0.83	0.86	0.84
LR	0.580	0.62	0.43	0.65	0.62	0.58
NN	0.88	0.88	0.86	0.88	0.90	0.88
LRCT	0.810	0.82	0.78	0.81	0.83	0.81

Table 17. Experiment 11 Aggregated Results.

For this experiment, the performance benefit of the LRCT algorithm compared to the other tree-based algorithms can clearly be seen. There is a noticeable difference in performance for both the OC1 and CART models compared to the LRCT model, indicating that the LRCT algorithm was able to learn the decision boundaries between classes more generally than those model types. Interestingly, however, the LRCT model's performance still lagged that of both KNN and the neural network.

Experiment 12: Fisher's Iris Dataset Classification

Experiment 12 sees the use of Fisher's famous iris classification dataset (Fisher, 1936). This dataset contains 150 samples of four features sampled from 50 of each of three different varieties of iris flowers. The features collected were the sepal length, the sepal width, the petal length, and the petal width, all measured in centimeters, for setosa, versicolor, and virginica varieties of iris flowers.

In this experiment, each of the length and width measurements were utilized as input features to predict the variety of iris the measurements correspond to. As before, the data was split into training, validation, and test sets; each of the models trained were trained as in the previous few experiments as well. Performance metrics for each of the models is presented in the following tables.

Actual/Predicted	*Setosa*	*Versicolor*	*Virginica*
Setosa	15	0	0
Versicolor	1	12	3
Virginica	0	0	14

Table 18. OC1 Confusion Matrix, Experiment 12

Actual/Predicted	*Setosa*	*Versicolor*	*Virginica*
Setosa	15	0	0
Versicolor	0	13	3
Virginica	0	1	13

Table 19. CART Confusion Matrix, Experiment 12

Actual/Predicted	*Setosa*	*Versicolor*	*Virginica*
Setosa	15	0	0
Versicolor	0	14	2
Virginica	0	2	12

Table 20. KNN Confusion Matrix, Experiment 12

Actual/Predicted	*Setosa*	*Versicolor*	*Virginica*
Setosa	15	0	0
Versicolor	0	13	3
Virginica	0	0	14

Table 21. LR Confusion Matrix, Experiment 12

Actual/Predicted	*Setosa*	*Versicolor*	*Virginica*
Setosa	15	0	0
Versicolor	0	15	1
Virginica	0	1	13

Table 22. Neural Network Confusion Matrix, Experiment 12

Actual/Predicted	*Setosa*	*Versicolor*	*Virginica*
Setosa	15	0	0
Versicolor	0	14	2
Virginica	0	1	13

Table 23. LRCT Confusion Matrix, Experiment 12

Model Type	*Overall Accuracy*	*Setosa F1*	*Versicolor F1*	*Virginica F1*	*Overall F1*
OC1	0.911	0.97	0.86	0.90	0.91
CART	0.911	1.0	0.87	0.87	0.91
KNN	0.911	1.0	0.88	0.86	0.91
LR	0.933	1.0	0.90	0.90	0.93
NN	0.956	1.0	0.94	0.93	0.96
LRCT	0.933	1.0	0.90	0.90	0.93

Table 24. Experiment 12 Aggregated Results

Each of the models in this experiment was able to identify each of the classes of iris flowers relatively efficiently, as indicated by the high accuracy and F1 scores of each of the models. However, the LRCT model was able to identify multivariate relationships within the data which generalized to improve performance on test data compared to the other explainable algorithms.

Experiment 13: Wisconsin Breast Cancer Dataset

Experiment 13 utilizes the diagnostic breast cancer dataset created by Dr. William Wolberg, Nick Street, and Olvi Mangasarian and accessed through the UCI Machine Learning Repository (Dua & Graff, 2017). This dataset contains over 500 instances of 30 variables taken from fine needle aspirates of breast masses. The goal of this problem is to correctly classify those samples into benign and malignant cases using the features available. This problem provides a unique test for the LRCT algorithm, as this dataset has a much higher dimensionality than the other problems faced. This higher dimensionality stresses the combinatorial complexity of the LRCT algorithm.

For this experiment, all model training procedures, data splitting, and preprocessing techniques were identical to the previous few experiments run. Results for this experiment are presented in Table 25.

Model Type	*Accuracy*	*True Negatives*	*False Negatives*	*True Positives*	*False Positives*	*Class 0 F1*	*Class 1 F1*	*Overall F1*
OC1	0.923	52	8	107	4	0.90	0.95	0.93
CART	0.959	52	3	112	14	0.94	0.97	0.96
KNN	0.942	50	4	111	6	0.91	0.96	0.94
Logistic Regression	0.959	51	2	113	5	0.94	0.97	0.96
Neural Network	0.959	50	1	114	6	0.93	0.97	0.96
LRCT	0.959	51	2	113	5	0.94	0.97	0.96

Table 25. Experiment 13 Results

All models in this experiment performed well on the classification task at hand. However, due to the important nature of this problem and identifying potentially life-threatening health issues in patients, there is a particular interest in minimizing the rate at which false negatives occur. Based on this metric, the neural network outperforms all other models with only one false negative. The LRCT and Logistic regression models are tied for second with regards to this performance metric, with only two false negatives present. Additionally, both the linear regression and LRCT algorithms resulted in only five false positives, whereas the neural network had six false positives. As the LRCT and linear regression models are also inherently explainable, whereas the neural network is opaque, these models may be more useful from a scientific standpoint to identify and analyze which specific relationships in the data are being used for predictive purposes.

Experiment 14: Wine Classification

The final experiment in this study contains a multiclass classification problem which contains 178 instances of 13 features of three classes of wines. Each class has at least 50 individual instances. Once again, this dataset was retrieved through the UCI Machine Learning Repository (Dua & Graff, 2017). This experiment followed the procedures of the previous experiments with regards to data splitting and model training procedures as the previous experiments. The results for this experiment are presented in the following figures.

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*
Class 0	19	1	1
Class 1	0	17	4
Class 2	0	0	12

Table 26. OC1 Confusion Matrix, Experiment 14

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*
Class 0	19	2	0
Class 1	1	16	4
Class 2	0	0	12

Table 27. CART Confusion Matrix, Experiment 14

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*
Class 0	20	0	1
Class 1	0	15	6
Class 2	0	0	12

Table 28. KNN Confusion Matrix, Experiment 14

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*
Class 0	20	1	0
Class 1	1	18	2
Class 2	0	0	12

Table 29. Logistic Regression Confusion Matrix, Experiment 14

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*
Class 0	18	3	0
Class 1	0	20	1
Class 2	0	0	12

Table 30. Neural Network Confusion Matrix, Experiment 14

Actual/Predicted	*Class 0*	*Class 1*	*Class 2*
Class 0	20	1	0
Class 1	2	17	2
Class 2	0	0	12

Table 31. LRCT Confusion Matrix, Experiment 14

Model Type	*Overall Accuracy*	*Class 0 F1*	*Class 1 F1*	*Class 2 F1*	*Overall F1*
OC1	0.889	0.95	0.87	0.83	0.89
CART	0.870	0.93	0.82	0.86	0.87
KNN	0.778	0.91	0.79	0.54	0.78
LR	0.926	0.95	0.90	0.92	0.93
NN	0.926	0.92	0.91	0.96	0.93
LRCT	0.907	0.93	0.87	0.92	0.91

Table 32. Experiment 14 Aggregated Results

In this final experiment, many of the same general performance trends as present in the previous experiments can be seen. The LRCT model outperforms many of the other explainable models in this experiment, especially the other tree-based models, and the neural network outperforms all other models. In general, these results serve as additional evidence to support the level of performance typically seen from the LRCT algorithm on both contrived and collected data and learning scenarios.

Comprehensive Summary

In this chapter, the results of the experiments conducted in the study are presented. In total, 14 experiments were completed across a range of datasets. The initial experiments conducted on artificial data were designed specifically to test LRCT's ability to learn a specific set of decision boundaries compared to other tree-based methods. Following those experiments, a series of additional experiments were conducted across a larger range of model types. These experiments contained binary and multiclass classification tasks across a range of both artificially generated and collected datasets. In general, the LRCT algorithm performed better than the other tree-based algorithms on most experiments, indicating that the research questions posed at the beginning of this work are answered through these experiments.

The contents of this chapter are a culmination of the research efforts of this study. At the beginning of this work, a series of research questions was posed related to whether an inherently interpretable machine learning algorithm can be created which improves upon the performance of other interpretable algorithms. To better answer these questions, a thorough search of the literature was conducted across different subfields within the machine learning community.

The first of the subfields investigated was related to decision trees. Built as one of the prototypical explainable machine learning algorithms, decision trees make a series of splits on input data to make predictions. Interestingly, however, decision tree research has focused primarily on improving model performance through ensemble methods, rather than improving model splitting methods. Only a small portion of the detected literature considered utilizing multivariate splits in decision trees, which could potentially improve predictive performance while simultaneously maintaining model interpretability.

The second subfield which was researched through the literature is the newer field of XAI. It was found that most of the work done in this field to date has focused on improving the explainability or interpretability of black-box models, such as neural networks, rather than improving the performance of interpretable models. While there were a few examples of algorithms which were designed for the latter purpose, the ones identified through the literature review do not directly relate to the research questions posed for this work.

After the literature review was completed, this study presents the LRCT algorithm. This algorithm, designed to efficiently identify multivariate splits across a range of variables in decision tree learning, resulting in an explainable model which is expected to achieve greater performance on complicated tasks than other explainable model types, thus allowing it to be used in scenarios where high levels of regulation are present, such as in defense, health care, and insurance. After the LRCT algorithm was presented, an overview of the experiments conducted in this chapter is made.

The following chapter, the final one of this work, will investigate the results of the experiments further, provide a more in-depth analysis of the overall performance of the LRCT algorithm compared to other algorithms, identify any shortcomings of this research, and provide a set of recommendations for future work. This analysis will review the initial goals of this study as well and identify how the results of the experiments contribute to these goals.

CHAPTER 5: FINDINGS AND RECOMMENDATIONS

In the beginning of this work, a series of Research Questions (RQs) was posed, with the main question being:

Can an inherently interpretable algorithm be developed with improved performance over traditional interpretable models?

The rest of this work, including the literature review, the formulation of the LRCT algorithm and its learning procedures, and the experiments conducted in the previous chapter, have all served to assist in answering this question. The focus of this final chapter will be to analyze the entirety of that completed work and formulate an answer to the primary RQ. This will be done by first analyzing the results of the experiments contained in Chapter 4 in greater detail, followed by identifying any limitations of the study, and making final conclusions about the answer this work provides for the primary RQ. Finally, recommendations for future work are presented to facilitate any additional research to be conducted in this area.

Experiment Results

As the results of each individual experiment have been discussed in greater detail in the previous chapter, the analysis that follows will focus on identifying aggregate trends and making conclusions about the overall performance of the LRCT algorithm and how it addresses the primary RQ of this work. The experiments conducted in this study ranged across a variety of classification tasks, including binary and multiclass classification on artificial and collected data. An aggregation of the experiment results is presented in Table 33.

Experiment	*Results*
Experiment 1: Two Variables, Single Linear Split	LRCT makes a near perfect classification with a single linear split, providing evidence that it can identify multivariate decision boundaries efficiently in simple situations.
Experiment 2: Five Variables, Single Linear Split	LRCT sees similar performance compared to the previous experiment, indicating that it can identify information-rich combinations of variables and perform feature selection.
Experiment 3: Two Variables, Single Polynomial Split	LRCT once again identifies an efficient decision boundary and achieves high performance with a single split. Additionally, LRCT's decision boundary is not as steep as the true decision curve, a feature of how the algorithm identifies splits.
Experiment 4: Five Variables, Single Polynomial Split	LRCT once again efficiently identifies which variables to perform splitting on, something which it appears the OC1 algorithm does not do.
Experiment 5: Two Variables, Single Linear Split, Noise	In every case, LRCT reverts to using traditional univariate splits instead of multivariate splits, a feature included in the algorithm to help ensure its performance is at least that of a traditional decision tree.
Experiment 6: Five Variables, Single Linear Split, Noise	The LRCT model identified a multivariate boundary only with noise level of 0.2, and this multivariate boundary led to worse

	performance than both the CART and OC1 models. This indicates that it is indeed possible for LRCT to overfit and draw incorrect conclusions.
Experiment 7: Two Variables, Single Polynomial Split, Noise	LRCT reverted to univariate splits for each of the cases in this experiment, once more indicating that LRCT is susceptible to noise. Additionally, the OC1 model was able to efficiently identify the decision boundary in this case with a noise level of 0.1, but OC1 performance degraded significantly as the noise level increased.
Experiment 8: Five Variables, Single Polynomial Split, Noise	With a noise level of 0.1, LRCT reverted to a univariate split. However, with noise levels of 0.2 and 0.3, LRCT was able to efficiently identify the decision boundaries, achieving over 96% accuracy on test data in both cases. Once more, OC1's performance degraded significantly as noise levels increased.
Experiment 9: Checkerboard Configuration	In this first experiment with more than tree-based models, the only model which performed exceptionally well was the KNN model. The logistic regression model performed exceptionally poorly, and the LRCT model performed poorly as well. The results of this experiment indicate that LRCT is not adept at identifying the kinds of relationships present in this data.
Experiment 10: Random Binary Classification	All models perform well in this experiment, achieving over 93% accuracy on the test data. However, virtually all models except for OC1 and logistic regression achieved approximately

	98% accuracy and F1 score, indicating that LRCT performed nearly as well as or better than all other models under consideration.
Experiment 11: Random Multiclass Classification	In this experiment, LRCT outperformed all other explainable model types except for the KNN model.
Experiment 12: Fisher's Iris Dataset Classification	As in the previous experiment, the LRCT algorithm was only outperformed by the neural network model in this experiment. The only explainable model the LRCT algorithm was matched in performance with was the linear regression model.
Experiment 13: Wisconsin Breast Cancer Dataset	Four models achieved the same accuracy on test data in this experiment, with the LRCT model being one of them. However, given the additional goal of minimizing false negatives in this situation, LRCT was tied among the explainable models in this regard and was only outperformed by the neural network model.
Experiment 14: Wine Classification	LRCT outperforms many of the other explainable models in this experiment, especially the other tree-based models. However, the logistic regression model outperformed all other explainable models and equaled the accuracy of the neural network.

Table 33. Experiment Results Summary.

The first four experiments were designed to test LRCT's ability to identify multivariate linear and polynomial splits in simple circumstances where splits across two variables perfectly classify the data. In these experiments, the LRCT algorithm performed exceptionally well, as it was able to identify the multivariate decision boundary nearly perfectly with a single split in all cases, even the ones requiring feature selection. LRCT performed much better than CART in these situations, due to the multivariate splitting characteristic of LRCT compared to the purely univariate splits made by CART. Additionally, OC1 performed similarly well to LRCT in these experiments, but OC1 was trained as in the original paper (Murthy, Kasif, & Salzberg, 1994), meaning the OC1 models which resulted were more complex than the LRCT models through their use of multiple splits.

The results of these experiments provide evidence for a positive answer to the primary RQ. When considering both the predictive performance and interpretability of results, LRCT outperforms both CART and OC1 in these experiments. One drawback to the LRCT algorithm which has been raised previously, however, is that in polynomial split experiment the decision function learned by the LRCT tree does not have as much curvature as the true decision boundary. This is due to the methods used to identify multivariate splits by the LRCT algorithm. Splits can be improved by increasing the number of bins created; however more data is required to ensure meaningful, statistically significant bins are used.

The following four experiments repeated the initial four, but this time noise was added to the training data in the form of a proportion of randomly incorrectly labeled instances. The results of these experiments are particularly interesting because they largely result in the trained LRCT trees creating only univariate splits. Multivariate splits were only created in Experiment 6 with a noise level of 0.2 and Experiment 8 with noise levels of 0.2 and 0.3, and only in

Experiment 8 did the splits result in better performance than the CART model. These results indicate that the LRCT algorithm is susceptible to noise, which was unexpected. Furthermore, the underperforming split identified by the LRCT algorithm in Experiment 6 indicates that LRCT can be susceptible to overfitting in some circumstances.

When comparing LRCT's performance in these four experiments to the performance of OC1, it becomes clear how important it is that LRCT can revert to univariate splits if needs be. In many of these cases, OC1 degrades significantly in performance as noise levels increase while CART and LRCT performance does not degrade at all or at a much slower rate. This stability can be attributed to the phenomenon that LRCT reverts to more-stable univariate splits if a multivariate split which outperforms that univariate split cannot be found.

Experiment 9, where the binary data is configured in a checkerboard configuration, is the first experiment with all model types under consideration in this study utilized. Additionally, this experiment was the first one that utilized a grid search methodology to select hyperparameters for each model except the neural network, which was trained using traditional methods and had the best weights preserved and used for testing. For this experiment, the LRCT algorithm was trained to a maximum depth of up to 21 and a highest polynomial degree of 2.

Unfortunately, LRCT did not perform as well as anticipated in this experiment. It was hypothesized that LRCT's use of binning while splitting would allow the algorithm to identify when clusters of points are present in the way they are in this experiment. However, LRCT performed nearly the worst out of all model types in this experiment. Additionally, LRCT took an extremely long time to train in this experiment, due to the characteristic that for every possible split considered there are possibly multiple linear regression models trained. The large number of computations required to thus train the models past a shallow depth makes LRCT inefficient. This feature will be discussed later in this chapter as an area of improvement for future research.

Experiments 10 and 11 contained the full set of algorithms as well, and all algorithms were trained the same way they were in the previous experiment. For these experiments, however, the data generation techniques were automated. In Experiment 10, a binary classification task was created with two variables and two clusters per class created for each of the two classes. For Experiment 11, four classes were created across 12 variables, with six of those variables being informative, four redundant, two repeated, and three clusters created for each individual class. Based on these experiments, a few important insights about the LRCT algorithm and how it answers the primary RQ can be made.

Firstly, it appears that multiple clusters for each of the classes can be problematic for the LRCT algorithm. This is due to the mathematical structure of how LRCT decides to make splits and how the algorithm implicitly treats each class as existing within a single cluster. Additionally, for multiclass classification, the LRCT algorithm was trained using a global approach rather than a one-versus-rest or alternative approach. As a result, the multivariate splits which occur in Experiment 11 are the result of averaging the regression models which are created on a per-class basis. Taking a one-versus-rest approach, in which individual models are created specifically to identify each of the classes to be identified, may yield better results.

One of the key takeaways from these experiments is that in both experiments the LRCT model performs nearly identically to or better than each of the other explainable and/or interpretable models. This result provides further evidence for a positive answer to the primary RQ. While the LRCT model did not perform as well as the KNN model in Experiment 11, LRCT also does provide more explicit information about the decision function used to determine the classification decisions. Furthermore, LRCT also performs automatic feature engineering and selection during this process, meaning it is better able to provide the end user with more specific information regarding the relationships learned within the data.

The final three experiments are designed to replicate the "real world" performance of LRCT and the other models under consideration as closely as possible. This goal is achieved by utilizing collected data across a variety of domains and problems.

The first of these problems is the multiclass iris classification dataset (Fisher, 1936). In this experiment, all the models were able to classify the test data with over 90% accuracy, but only three models were able to achieve greater performance. These three models were the neural network (95.6%), the logistic regression model (93.3%), and the LRCT model (93.3%). This finding is very informative as it relates to the primary RQ, since the LRCT model performed equally as well or better than all the other inherently interpretable model types.

In the second of these problems, which identified whether patients had breast cancer based on clinical findings the experiment once more yielded positive results with regards to the primary RQ; the LRCT model was able to efficiently identify a single split across three variables of degree 2 which achieves performance equal to or better than all the other models under consideration, including the neural network. Not only does the LRCT model in this case provide better predictive performance than all other models when also considering the goal of minimizing false negatives, but it also does so in an efficient, explainable manner which can be studied and validated by experts.

The last experiment in this study, where the task was to identify different classes of wine, presents the final pieces of information available through this study into the performance of the proposed LRCT algorithm. For this problem, there are three classes of wines to be identified which are distributed slightly unevenly in the training data. There are 59 instances of class 0, 71 instances of class 1, and 48 instances of class 2. Once again, all models perform relatively well in this experiment, with the least performant one being the KNN model which achieved 77.8% accuracy on the test data. The OC1 model slightly outperformed the CART model, but the LRCT algorithm outperformed all except the neural network and linear regression models, which perform nearly identically and achieve the same accuracy on the test data. While the LRCT model does not outperform every other explainable method in this experiment, it does once again outperform the other tree methods studied. Furthermore, the parameters which resulted in this performance for the LRCT model was a maximum depth of three and splits across a maximum of two variables. Similarly, the CART model was also trained to a depth of three, indicating that the LRCT model was able to identify efficient multivariate splits which provided additional predictive performance compared to the univariate splits created by the CART algorithm. The results of this experiment further support a positive answer to the primary RQ, as LRCT outperforms many of the other explainable algorithms.

Limitations

There are a few limitations to this study. The first of these is that it would be impossible to test all model types for this study. Random forest and other tree-based ensemble learners were left out of consideration. The original intent for this omission was to focus on testing the LRCT algorithm against other interpretable non-ensemble algorithms. However, future research should incorporate ensemble methods in its studies as well.

Another limitation of this study was the time required to train each LRCT model, particularly when hyperparameter grid search was applied. Due to the way LRCT is implemented for this study, both in terms of algorithmically and in the software used, training each individual LRCT model was a timely process as maximum depth increased. To save some time during the training process, parameter searches were restricted to regions expected to contain the optimal solutions. Despite this limitation, the results of this study do show that the proposed LRCT algorithm can perform better than many of the other models under consideration in this study, particularly when explainability is also a factor.

A third limitation to this study is the number of individual experiments conducted. The goal of this study is to identify whether the LRCT algorithm proposed herein can serve as a legitimate, powerful replacement to many existing ML algorithms, all while remaining interpretable. As such, this study heavily relies on the evidence provided through the experiments conducted. However, regardless of the total number of experiments conducted in this study, such research will always be inherently incomplete; there is always more data which can be tested. There is perhaps no way to mathematically *prove* that the LRCT algorithm will be better than another, alternative algorithm. Despite this, the study was conducted in such a way that a multitude of classification scenarios were tested, each one providing information into the relative performance levels of each of the algorithms under question. Similarly, by running many experiments on a variety of different datasets, a more accurate depiction of overall performance compared to alternative algorithms can also be determined.

Conclusions and Original Contribution

Despite the shortcomings mentioned in the previous section, this study does provide answers to the RQs posed at the beginning of this work. The literature review presented in Chapter 2 provides an answer to RQ1, where existing multivariate decision tree algorithms are identified. OC1 is identified as a leading multivariate decision tree algorithm, and it was utilized in the experimental portion of the study to provide an answer to RQ4.

The experimental setup of this study was also intended to provide answers to RQ2 and RQ3, which asked whether applying linear regression as a feature engineering technique in classification tree learning improves performance over traditional classification trees and other common interpretable algorithms, respectively. After analyzing the experimental results of the study, LRCT performed equally as well as or better than CART in 12 of the 14 experiments, except for experiment 6 with a noise level of 0.2. LRCT also performed as well as, better than, or within a margin of error of OC1 in 12 experiments as well, except for experiment 5 with noise level 0.1, experiment 7 with noise level 0.1, and experiment 8 with noise level of 0.1. Of the five experiments where additional interpretable models are considered, LRCT obtained the best or tied for best performance in two of them and performed second best in an additional two. LRCT performed equally as well as the neural network in only one experiment.

The results of the experimental portion of this study do provide favorable evidence for the RQs presented in the first chapter of this work. By performing better than both CART and OC1 in most of the experiments presented in this study, it appears that LRCT will outperform these algorithms on average. This evidence provides support for a favorable answer to RQ2 and RQ4.

The evidence this study provides for RQ3 is more nuanced than the relatively straightforward answer to the other RQs. Compared to the other interpretable methods tested in the five experiments with all models considered, LRCT performed among the top two in four of these experiments and performed the best in two of these. While these performance metrics do provide evidence that LRCT may consistently perform among the top interpretable algorithms, it is interesting that it did not consistently outperform all other models in these experiments.

The LRCT model was typically outperformed by the KNN models in these experiments. This trend indicates that perhaps LRCT will consistently be outperformed by KNN, but due to the small sample size of experiments conducted in this study such a conclusion cannot yet be made. One point of interest, however, is the level of interpretability available via both LRCT and KNN. Referring to the definitions made in Chapter 1 of this work, it is clear that both model types are Transparent and Understandable, but the direct Interpretability of KNN is more difficult. The entire basis of trust and understanding for KNN models comes from the idea that the closest K points in the training data have corresponding labels which result in the prediction made on new data. By contrast, LRCT models can be examined in such a way to understand the characteristics of the model more rigorously. Due to this more explanatory interface between the researcher and LRCT, it would perhaps be more useful in certain circumstances for LRCT to be applied in place of KNN.

In total, several conclusions can be drawn from the results of this study. Firstly, as evidenced by the performance of LRCT compared to other decision tree models, it can be concluded that feature engineering and selection using linear regression is a powerful technique which can be applied to decision tree learning to create multivariate decision trees without harming explainability. Secondly, applying these methods for feature engineering and selection do typically result in improved predictive performance compared to traditional decision trees and other leading multivariate decision tree algorithms. Lastly, these techniques also rival the performance of many other interpretable model types; depending on the circumstances surrounding the problem at hand, applying these techniques also provide important benefits in terms of added explanatory value.

Based on the findings of the literature review, this work makes several original contributions to the total body of knowledge in decision trees and XAI. Firstly, as noted previously, very little work has been done to truly improve the way decision trees make their splits. Instead, most recent work has been focused on creating ensembles of decision trees to improve overall performance. While there has been some work to create multivariate decision trees in particular, these models are note widespread in practice. In contrast to this paradigm, this work successfully creates a multivariate classification tree algorithm which improves performance compared to traditional classification trees and other multivariate trees, marking an improvement over the state of the art and a contribution of original knowledge.

A second contribution this work provides is in XAI. In contrast to much of the literature surrounding XAI, which focuses on bringing higher levels of interpretability and explainability to opaque models, this work focuses on developing an inherently interpretable model with greater predictive power than existing interpretable models. By taking this opposite approach, this work provides original contribution to the field by showing that interpretable models can be developed which provide more accurate and powerful predictions instead of relying on building explainability into powerful black box models such as neural networks.

Recommendations for Future Research

While much was discovered and analyzed in this study, there are gaps in knowledge and additional research which should be conducted to address these gaps. Generally, these gaps can be categorized as relating to the implementation of the LRCT algorithm or to the performance of the LRCT algorithm compared to other model types.

With respect to the LRCT algorithm's implementation, there are several questions and recommendations for future research which can be made. Firstly, as noted previously in this work, the implementation used for the LRCT algorithm is much slower than other models. This leads to the possible research question: can the LRCT algorithm be optimized for speed without hindering performance? Currently, the algorithm uses a brute force technique to test all combinations of variables under the hyperparameters considered. This hyperparameter search is computationally costly, and it would be highly beneficial if a heuristic search could be applied to this method. It is currently unknown whether such a search could take place without significantly hindering the resulting model's performance, so this problem is left as a major area of future research.

Another possible method for addressing the high computational cost associated with the LRCT algorithm would be to investigate whether quantum computing could be utilized to speed up the training phase. As quantum computing continues to mature, its applicability in training machine learning models continues to become more and more apparent (Biamonte, et al., 2017). Furthermore, it has been shown that linear regression models, in particular, can be trained using a quantum algorithm, and that training using this algorithm does result in faster performance than when using a classical computer (Schuld, Sinayskiy, & Petruccione, 2016). It would stand to reason, then, that the LRCT algorithm could be adapted to run on a quantum computer as well.

A second area of future research and improvement in the implementation of the LRCT algorithm is whether the algorithm can be modified and applied to regression problems as well. The current implementation of the algorithm only supports classification tasks, as it relies on there being distinct groups or classes in the target variable. Distinct clusters such as this do not exist in a regression problem, but it is potentially possible to apply forms of clustering or other aggregation technique to artificially impose such groups or classes on the otherwise continuous target variable. Such research is beyond the scope of this work, but it would be a logical following stage for the research.

Regarding LRCT's performance against other model types, one of the primary areas of future research would be to test the algorithm's performance against some of the ensemble methods discussed previously, such as the random forest (Breiman, Random Forests, 2001). While many of these methods are not inherently as explainable or interpretable as CART or LRCT, they do consistently outperform other model types and often offer interpretability or explainability interfaces. Future research should therefore be conducted to test LRCT's performance against such model types to help understand whether LRCT can provide the same predictive power of ensembles within a single, easy to understand model. The final question which follows would be whether ensembles of LRCT models, either created following the methods for random forests or for other types of models, would provide additional performance improvements compared to other ensemble methods.

Chapter Summary

In this chapter, the final of this work, the results of the experiments presented in Chapter 4 are analyzed in greater detail, the answers to the RQs posed at the beginning of this work are explored, general conclusions are derived, and recommendations for future research are made.

In summary, the results of the experiments conducted in this research positively answer the RQs posed: applying linear regression as an intermediate feature engineering step during splitting in a classification tree does improve the tree's performance compared to traditional classification tree algorithms. Furthermore, the LRCT algorithm proposed here also sees increased performance compared to other multivariate tree methods, and it also performs competitively with all other interpretable methods tested. These results, combined with the improved interpretability of the LRCT algorithm compared even to some other interpretable methods, leads to the conclusion that inherently interpretable models can be created with improved predictive performance compared to traditional models. With further research into improving the training speed and types of questions which can be trained using the methods proposed here, as well as additional research in comparing LRCT's performance with ensemble methods, it is possible that inherently interpretable methods can help bring ML and AI to many industries which require better performance and more explainability than the current state of the art.

REFERENCES

Allen-Zhu, Z., Li, Y., & Liang, Y. (2020). Learning and Generalization in Overparametrized Neural Networks, Going Beyond Two Layers. arXiv Preprint.

Anderson, E. (1935). The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, pp. 2-5.

Awaysheh, A., Wilcke, J., Elvinger, F., Rees, L., Fan, W., & Zimmerman, K. (2019). Review of Medical Decision Support and Machine-Learning Methods. *Veterinary Pathology*, 512-525.

Barredo Arrieta, A., Diaz-Rodriguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., . . . Herrera, F. (2019). Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 195-202.

Bianchini, S., Müller, M., & Pelletier, P. (2020). Deep Learning in Science.

Breiman, L. (2001). Random Forests. *Machine Learning*, 5-32.

Breiman, L. (2001). Statistical Modeling: The Two Cultures. *Statistical Science*, 199-231.

Breiman, L. (2017). *Classification and regression trees.* Routledge.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees.* Wadsworth International Group.

Brodley, C., Utgoff, P., & Kibler, D. (1995). Multivariate Decision Trees. *Machine Learning*.

Cao, H., Si, C., Sun, Q., Liu, Y., Li, S., & Gope, P. (2022). ABCAttack: A Gradient-Free Optimization Black-Box Attack for Fooling Deep Image Classifiers. *Entropy*, 412.

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature*, 20-23.

Chander, A., Srinivasan, R., Chelian, S., Wang, J., & Uchino, K. (2018). Working with beliefs: AI transparency in the enterprise. *Workshops of the ACM Conference on Intelligent User Interfaces.*

Che, Z., Purushotham, S., Khemani, R., & Liu, Y. (2016). Interpretable deep models for ICU outcome prediction. *AMIA Annual Symposium Proceedings* (p. 371). American Medical Informatics Association.

Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*.

Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data 5*, 153-163.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 273-297.

Cuartas, M., Ruiz, E., Ferreño, D., Setién, J., Arroyo, V., & Gutiérrez-Solana, F. (2021). Machine learning algorithms for the prediction of non-metallic inclusions in steel wires for tire refinement. *Journal of Intelligent Manufacturing*, 1739-1751.

Dev, S., & Phillips, J. (2019). Attenuating Bias in Word Vectors.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.

Du, S. S., Zhai, X., Poczos, B., & Singh, A. (n.d.). Gradient Descent Provably Optimizes Over-parametrized Neural Networks. *International Conference on Learning Representations.* 2019.

Dua, D., & Graff, C. (2017). UCI Machine Learning Repository.

European Commission. (2018, 05 27). *2018 reform of EU data protection rules.* Retrieved 05 27, 2020

Ferndandez, A., Herrera, F., Cordon, O., Jose del Jesus, M., & Marcelloni, F. (2019). Evolutionary Fuzzy Systems for Explainable Artificial intelligence: Why, When, What for, and Where to? *IEEE Computational Intelligence Magazine*, 69-81.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 179-188.

Frankle, J., & Carbin, M. (2018). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computational Systems Science*, 119-139.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 367-378.

Fu, C., Guo, C.-Y., Lin, X.-R., & Lu, C.-J. (2010). Tree Decomposition for Large-Scale SVM Problems. *The Journal of Machine Learning Research*.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining Explanations: An Overview of Interpretability of Machine Learning.

Gleicher, M. (2016). A Framework for Considering Comprehensibility in Modeling. *Big Data*.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). *A Survey of Methods for Explaining Black Box Models.* New York, NY, USA: Association for Computing Machinery.

Gunning, D., & Aha, D. (2019). DARPA's Explainable Artificial Intelligence (XAI) Program. *AI Magazine*, 44-58.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning : with Applications in R.* Springer.

Jia, R., & Liang, P. (2017). Adversarial Examples for Evaluating Reading Comprehension Systems. arXiv Preprint.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 119-127.

Kim, H., & Loh, W.-Y. (2001). Classification Trees with Unbiased Multiway Splits. *Journal of the American Statistical Association*, 589-604.

Kindermans, P.-J., Schutt, K., Alber, M., Muller, K.-R., Erhan, D., Kim, B., & Dahne, S. (2017). Learning how to explain neural networks: Patternnet and patternattribution.

Kodiyan, A. A. (2019). An overview of ethical issues in using AI systems in hiring with a case study of Amazon's AI based hiring tool.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.

Langley, P., Meadows, B., Sridharan, M., & Choi, D. (2017). Explainable agency for intelligent autonomous systems. *AAAI Conference on Artificial Intelligence*, (pp. 4762-4763).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition.

Lee, T.-H., Ullah, A., & Wang, R. (2020). Bootstrap Aggregating and Random Forest. *Macroeconomic Forecasting in the Era of Big Data: Theory and Practice*. Springer International Publishing.

Liang, X. (2016). Exploring the Big Data Using a Rigorous and Quantitative Causality Analysis. *Journal of Computer and Communications*, 54-60.

Lipton, Z. C. (2017). The Mythos of Model Interpretability. arXiv Preprint.

Liu, F. T., Ting, K. M., & Zhou, Z. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, (pp. 413-412).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Lou, Y., Caruana, R., Hooker, G., & Gehrke, J. (2013). Accurate Intelligible Models with Pairwise Interactions. *KDD'13, August 11-14, 2013, Chicago, Illinois, USA*.

Luna, J. M., Gennatas, E. D., Ungar, L. H., Eaton, E., Diffenderfer, E. S., Jensen, S. T., . . . Valdes, G. (2019). Building more accurate decision trees with the additive tree. *Proceedings of the National Academy of Sciences*, 19887-19893.

Masegosa, A. (2020). Learning under Model Misspecification: Applications to Variational and Ensemble methods. *NeurIPS*, 5479-5491.

Master, N., Zhou, Z., Miller, D., Scheinker, D., Bambos, N., & Glynn, P. (2017). Improving predictions of pediatric surgical durations with supervised learning. *International Journal of Data Science and Analytics*.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.

Messenger, R., & Mandell, L. (1972). A Modal Search Technique for Predictive nominal Scale Multivariate Analysis. *Journal of the American Statistical Association*, 768-722.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Muller, K.-R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 211-212.

Montavon, G., Samek, W., & Muller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 1-15.

Mood, C. (2010). Logistic regression: Why we cannot do what we think we can do, and what we can do about it. *European sociological review*, 67-82.

Moosavi-Dezfooli, S.-M., Fawsi, O., & Frossard, P. (2017). Universal adversarial perturbations. arXiv Preprint.

Morgan, J. N., & Sonquist, J. A. (1963). Problems in the Analysis of Survey Data, and a Proposal. *Journal of the American Statistical Association*, 415-434.

Murthy, S., Kasif, S., & Salzberg, S. (1994). A System for Induction of Oblique Decision Trees. arXiv Preprint.

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). InterpretML: A Unified Framework for Machine Learning. arXiv Preprint.

Odense, S., & Garcez, A. S. (2020). Layerwise Knowledge Extraction from Deep Convolutional Networks. ArXiv.

Pearl, J. (2022). *Direct and Indirect Effects.* New York: Association for Computing Machinery.

Pedregosa, F., Varoquaux, G., Gramfort, A., & Michel, V. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 81-106.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier.

Roscher, R., Bohn, B., Duarte, M. F., & Garcke, J. (2020). Explainable Machine Learning for Scientific Insights and Discoveries. *IEEE Access*, 42200-42216.

Rosenfeld, A., & Richardson, A. (2019). Explainability in human-agent systems. *Autonomous Agents and Multi-Agent Systems*.

Ross, Q. J. (1993). *C4.5: Programs for Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Sahand, H., Kind, M. C., & Brunner, R. J. (2019). Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 1-1.

Schuld, M., Sinayskiy, I., & Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*.

Searle, J. R. (1980). Minds, Brains, and Programs. *Behavioral and Brain Sciences*.

Selvaraju, R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE International Conference on Computer Vision*, (pp. 618-626).

Shrikumar, A., Greenside, P., Shcherbina, A., & Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

Stanton, J. M. (2001). Galton, Pearson, and the Peas: A Brief History of Linear Regression for Statistics Instructors. *Journal of Statistics Education*.

Sun, L., Versteeg, S., Boztas, S., & Rao, A. (2016). Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study.

Sun, L., Zou, B., Fu, S., Chen, J., & Wang, F. (2019). Speech emotion recognition based on DNN-decision tree SVM model. *Speech Communication*, 29-37.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomaticattribution for deep networks. *International Conference on Machine Learning*, (pp. 3319-3328).

Tarsala, P., & Kasprzyk, R. (2021, 05). Deep Learning Algorithms in Computer Vision.

The Age of AI (2019). [Motion Picture].

Therneau, T., & Atkinson, B. (2019). rpart: Recursive Partitioning and Regression Trees.

Thudumu, S., Branch, P., Jin, J., & Singh, J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*.

Tolomei, G., Silvestri, F., Haines, A., & Lalmas, M. (2017). Interpretble predictions of tree-based ensembles via actionable feature tweaking. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 465-474).

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, 433-460.

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 2579-2605.

Vapnik, V. (1999). *The Nature of Statistical Learning Theory.* Springer New York.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need.

Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to Use t-SNE Effectively. *Distill.*

Wenzel, F., Galy-Fajou, T., Deutsch, M., & Kloft, M. (2017). Bayesian Nonlinear Support Vector Machines for Big Data. arXiv.

Wilkinson, L. (1992). Tree Structured Data Analysis: AID, CHAID and CART. *Sawtooth/SYSTAT Joint Software Conference*.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems*, 37-52.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 2921-2929).

Zhu, J., Liapis, A., Risi, S., Bidarra, R., & Youngblood, G. (2018). Explainable AI for Designers: A Human-Centered Perspective on Mixed-Initiative Co-Creation. *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, (pp. 1-8).

Zilke, J., Mencia, E., & Janssen, F. (2016). Deepred-rule extraction from deep neural networks. *International Conference on Discovery Science* (pp. 457-473). Springer.