PROBABILISTIC SPILT POINT DECISION TREES (PSPDT)

A Thesis

Presented to

the Faculty of the College of Graduate Studies

Tennessee Technological University

by

Sharanya D Aavunoori

In Partial Fulfillment

of the Requirements of the Degree

MASTER OF SCIENCE

Computer Science

August 2023

AN ABSTRACT OF A THESIS

PROBABILISTIC SPILT POINT DECISION TREES (PSPDT)

Sharanya D Aavunoori

Master of Science in Computer Science

Transparency in a machine learning model is not, by itself, sufficient for trust, particularly in fields like medicine. One way to engender trust in a learning system would be to enable end-users to reconcile the results with their domain knowledge. In this research, we introduce a modified version of decision trees called Probabilistic Split Point Decision Trees (PSPDTs). Our approach improves the transparency and believability of decision trees by building a variant of Consolidated Decision Trees (CDTs) using a combination of the Consolidated Tree Construction algorithm (CTC) and a bootstrapping approach to rule induction. PSPDTs aim to be a learning model algorithm that (at least) maintains accuracy while increasing stability and transparency, better reflecting real-world factors, and being honest with the uncertainty involved. By doing so, PSPDTs help to increase trust in decision trees. This thesis examines the validity of the mathematical assumption underlying PSPDTs and compares the accuracy of an initial PSPDT implementation to more traditional decision tree induction algorithms.

CONTENTS

CERTIFICATE OF APPROVAL OF THESIS

PROBABILISTIC SPILT POINT DECISION TREES (PSPDT)

by

Sharanya D Aavunoori

Graduate Advisory Committee:

Douglas Talbert, Chair Date

William Eberle Date

Mike Rogers Date

Approved for the Faculty:

Mark Stephens, Dean Date
College of Graduate Studies

ACKNOWLEDGMENTS

I want to thank my advisor, Dr. Doug Talbert, for his support and guidance throughout my graduate studies. I am grateful to him for allowing me to implement his groundbreaking idea as the basis for my thesis. I would also like to extend my heartfelt thanks to my advisory committee members, Dr. Bill Eberle and Dr. Mike Rogers. Furthermore, I would like to acknowledge the contribution of Ethan Owens and Matthew Beech. Their feedback after reviewing the code has been immensely helpful.

LIST OF LISTINGS

LIST OF LISTINGS

CHAPTER 1

INTRODUCTION

1.1 Overview

One of the most popular classification models in medical decision-making is the decision tree. Decision trees are effective, reliable, and easy to interpret, and their entire structure can be visualized in a simple flow chart, making them "white box" models. Decision trees have been extensively used in healthcare, where they have proven useful in diagnosing diseases such as breast cancer[25]. Despite their inherent transparency, there are aspects of the structure of decision trees that do not reflect the mental model of clinicians and could be a barrier to trust[45].

One of the challenges in rule learning, in general, is that many rule learning systems impose hard decision boundaries for continuous [53]. This is true for the most common variants of decision tree induction as well [2]. These hard boundaries and points can change with small perturbations to the training data due to algorithm instability, and rule induction typically produces a large number of rules that must be processed. To address these challenges, bootstrapping has been used to combine the results of multiple rule induction processes to provide a measure of variance for continuous attribute decision boundaries and accuracy point estimates [45].

To improve the trustworthiness of learning systems, they should be accurate, transparent, stable, and honest with respect to uncertainty, representing real-world variability and uncertainties. We introduce the Probability Split Point Decision Tree (PSPDT) algorithm to increase trust and transparency in classification decisions by incorporating stability, probabilistic reasoning, and real-world uncertainties with external variables. The PSPDT algorithm inherits the basic characteristics of decision trees, such as transparency and induces Gaussian distributions over split points to

model estimate probabilistic decision boundaries that are more reflective of the real world and, ideally, more consistent with mental models of domain experts.

In this work, we use a prototype implementation of PSPDT to assess the fundamental assumptions behind the algorithm. This work combines existing techniques that have been used to increase trust in data-driven decision models to generate a novel decision tree induction algorithm that we call Probabilistic-Split Point Decision Trees (PSPDTs) and seeks to test the following hypothesis: Modeling continuous decision boundaries as Gaussian distributions learned by a consolidated decision tree approach creates a *valid* and *useful* classification model.

To assess this hypothesis, we examine the following research questions:

- Do the split points follow Gaussian distribution?
- Do PSPDTs have an average accuracy that is not statistically significantly different (at the $p = 0.05$ level) than traditional decision trees built using CART or CTC with similar hyperparameters?

1.2 Motivation

In traditional decision-tree classification, a feature (an attribute) of a tuple is either categorical or numerical. When a numeric feature is identified as a split attribute, a precise and definite point value is usually assumed. Decision trees, however, are known to be highly sensitive to their training set, and, thus, this value would likely be different if the training set were slightly different[13, 46]. Therefore, we argue that numeric decision boundaries are best captured not by a single point value but by a range of values giving rise to a probability distribution. This research mainly focuses on the probabilistic approach of a decision tree.

Our research is inspired by R. Waitman's bootstrapping by rule induction model[45] and the Consolidated Tree Construction (CTC) algorithm by Jesús M. Pérez[20, 33]. CTC works by re-sampling the training sample and building a tree using votes from

each sub-sample, resulting in a final tree[20, 33]. CTC aims to provide a more stable explanation of the classification [33]. CTC induces a single tree, ensuring the base classifier's comprehensibility [20] and retaining the transparency of the classification model.

The bootstrapping rule induction method introduced by R. Waitman and others aims to reduce the number of rules presented to the analyst and measure and increase the stability of the rule induction process. In this work, we use the bootstrapping rule induction method over multiple samples constructed using the CTC algorithm to achieve improved transparency and rule stability.

The PSPDT algorithm adapts CTC to construct a tree using the training dataset and then uses a probabilistic approach to classify the test dataset. PSPDT uses a Consolidated Tree Construction (CTC) algorithm to combine multiple sub-samples using a voting mechanism to form a single consolidated tree. We then use the probabilistic distribution of the split points to inform that prediction process.

This work describes the PSPDT algorithm, which has the ultimate goal of improving explainability in classification models and trusts in the machine learning model. PSPDT incorporates a probabilistic classification model. We evaluate the algorithm with databases from the UCI repository benchmark [44]. We describe the PSPDT fit and prediction algorithms in section 3. We provide characteristics of the datasets used in the experiments along with details about the experimental methodology in Section 3. In Section 4, we analyze the behavior of the PSPDT algorithm and compare it to traditional decision trees. Finally, Section 5 presents the conclusions and future work.

1.3 Contributions

Our contributions from this research are as follows:

- Introduction of a novel PSPDT algorithm,
- Validation that split points follow a Gaussian distribution, and

- An initial assessment of PSPDT accuracy through a comparison to other selected decision tree induction algorithms.

CHAPTER 2

BACKGROUND

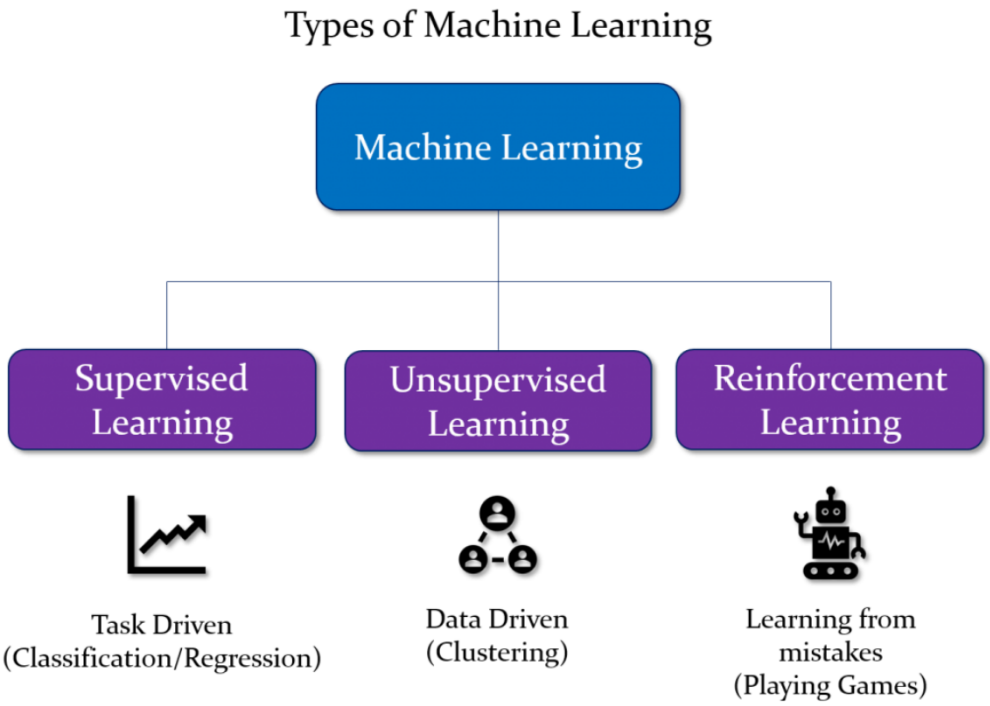
This chapter discusses various machine learning techniques, developments in decision trees, as well as rule induction methods and techniques. Here, we also discuss the CTC algorithm and bootstrapping rule induction techniques that have led to the PSPDT algorithm and experiments presented in this paper.

2.1 Machine learning

There are several machine learning algorithms, and they are typically classified into the following categories based on their modeling approach as shown in Figure 1:

Figure 1

Types of Machine Learning



This data can be found in [37]

2.1.1 Supervised learning

Supervised machine learning uses labeled data during the training phase of the model. In this approach, the training data is assigned labels during the preparation phase, and it is then used to train and test the model. The model aims to learn the relationship between the input features and output labels to make predictions about new, unseen data."Supervised" refers to the human oversight and labeling necessary for this approach[17]. Most available data are unlabeled, so human interaction is required to accurately label the data for supervised learning. This process can be resource-intensive since accurately labeled training data is needed.

Supervised machine learning is commonly used to classify unseen data into predefined categories. It can also be applied to tasks such as forecasting changes in house prices or predicting customer purchase trends. Some specific supervised machine learning applications include classifying attribute vectors, images, and text documents and forecasting future trends and outcomes by learning patterns from the training data. Supervised learning provides a robust framework for training models to recognize objects, classify data, and make predictions based on learned patterns and relationships between input data and the provided labels[29].

Supervised learning is classified into two categories, classification, and regression. For example, determining if an email is spam or not is a classification problem. K-nearest neighbor, Naive Bayes, Decision trees, etc., are commonly used for classification[29]. In regression, the labels are continuous values, such as the age or weight of a person as these variables have multiple values. Common regression algorithms include linear regression, support vector regression, and decision tree regression. In the latter part of this section, we will focus more on the decision tree classification algorithm[42].

2.1.2 Unsupervised Learning

Unsupervised machine learning involves training models on unlabeled data to identify patterns, trends, and groupings within the datasets. It can serve as a valuable approach during the initial exploratory phase to enhance comprehension of the data[42]. In contrast to supervised learning, unsupervised learning is more hands-off, as the model may require less human supervision, although a human may specify model hyperparameters like the number of clusters. Unsupervised learning is well-suited for uncovering hidden relationships and trends within the data itself[29].

Unsupervised learning harnesses techniques such as grouping data based on similar features or identifying underlying patterns to gain valuable insights from unlabeled data. By leveraging unsupervised learning, analysts can effectively extract knowledge

and make data-driven decisions without relying on pre-existing labels or extensive human guidance[42]

Unsupervised learning is used for the following three main tasks: clustering, dimensionality reduction, and association rule mining. In clustering, the data is grouped based on similarities or differences. The goal of clustering is to maximize the similarity between the samples in the same cluster while maximizing the distance between the samples in different clusters. Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic. K-means and Gaussian mixture models are examples of clustering algorithms[42].

Association rule mining is a rule-based method that finds relationships among variables in a given dataset. This method aims to discover meaningful relationships among samples. By determining the minimum support threshold, the algorithm identifies all frequent item sets present in the dataset. Subsequently, it generates association rules based on these frequent item sets based on the user-specified confidence threshold. Association rule-based mining is frequently used in market-based analysis, which allows companies to understand the purchasing patterns among various products and enables them to better understand consumer behavior to develop effective recommendation engines and cross-selling strategies. Apriori, Eclat, and FP-Growth are a few of the association-rule-based algorithms[29].

2.1.3 Reinforcement learning

Reinforcement learning is a branch of machine learning that focuses on making appropriate decisions that maximize rewards in specific situations. It is widely used to determine appropriate behaviors (i.e, a policy) to follow in a given context. Unlike supervised learning, where training data contains the correct answers and the model learns from those, reinforcement learning does not rely on a predefined answer key. Instead, a reinforcement agent relies on learning from experience with a user-specified reward function as its feedback mechanism. Reinforcement learning, often referred

to as RL, revolves around the science of decision-making. It involves learning the optimal behavior within an environment to achieve the highest possible reward. Unlike supervised or unsupervised machine learning, RL accumulates data through trial-and-error methods rather than relying on explicit input data[17].

RL employs algorithms that learn from the outcomes of actions and decide on subsequent steps to take. Feedback is received after each action, enabling the algorithm to evaluate its choices. This technique is particularly useful for automated systems that need to make numerous small decisions without human guidance. Reinforcement learning is characterized by its autonomous and self-teaching nature. It learns through a process of trial and error, where actions are performed with the objective of maximizing rewards. In essence, it learns by doing, aiming to achieve the best possible outcomes[17].

2.2 Decision tree and rule induction techniques

In AI and ML, sometimes learning why a classifier makes a decision is as important as the accuracy of that algorithm. This plays a crucial role especially when the classifier is used in decision support systems like medical diagnostics, fraud detection, counter-terrorism, etc[27]. Some classification algorithms can easily be interpreted by human beings. Decision trees and rule sets are such classifiers.

2.2.1 Decision tree induction

Decision trees are a modeling technique used to represent decisions and their outcomes in a branching structure. They have been employed in various fields such as business, economics, and operations management to analyze organizational decision-making processes. Decision trees can also be used in predictive modeling to assess the potential success of different decision sequences aimed at achieving a specific goal. In a decision tree, different nodes represent different stages of decision-making. The root node typically represents the entire dataset in machine learning. Leaf nodes, on

the other hand, mark the endpoint of a branch or the final output resulting from a series of decisions. Once a leaf node is reached, the decision tree does not branch further. In machine learning, the internal nodes of a decision tree correspond to the features of the data, while the leaf nodes represent the outcomes. Decision trees are a commonly used supervised learning approach, a technique that relies on labeled data to train models. Decision trees can also be applied to both classification and regression problems.

One of the key advantages of decision trees in machine learning is their simplicity, both in terms of model structure and interpretability. The tree-like structure makes it easy to understand the decision-making process of the model. This is important for explainability, as it allows humans to comprehend the reasoning behind a model's output. Decision trees offer transparency by providing clearly observable decision branches.

Example decision tree algorithms include ID3 [35], C4.5[36], and CART [7].

- **ID3 (Iterative Dichotomiser 3)**: ID3 is one of the earliest decision tree algorithms developed in the late 1970s by Quilan. It uses the information gain metric to determine the best feature to split on at each tree node. ID3 is a greedy algorithm, meaning it selects the best split at each node without considering the long-term effects on the tree[35].

- **C4.5**: C4.5 is an extension of ID3 and was developed in the early 1990s by Quilan. It uses the gain ratio metric and adds the ability to handle missing values and continuous features. C4.5 also adds pruning, a technique for reducing the tree size and avoiding overfitting[39, 52]

- **CART (Classification and Regression Trees)**: CART was developed in the late 1980s by Breiman. Unlike ID3 and C4.5, which only support categorical target variables, CART can be used for categorical and continuous target

variables. CART uses the Gini impurity metric to determine the best feature to split on at each tree node[8].

These algorithms can differ from each other in several ways. Here are some of the key distinctions:

- **Type of Attributes**: Decision tree algorithms can handle different types of attributes, including categorical, ordinal, and continuous. Some algorithms may be specifically designed to handle certain types of attributes more effectively.

- **Split Criteria**: When making decisions at each node of the tree, decision tree algorithms employ various split criteria. Common split criteria include information gain, gain ratio, Gini index, and variance reduction. These criteria measure the impurity or homogeneity of the data at each split and guide the algorithm's decision-making process[10, 43]

- **Type of Split**: Decision tree algorithms can perform different types of splits based on the attribute's nature and value. For categorical attributes, the splits can be based on exact matches or subset relationships. For continuous attributes, the splits can be based on threshold values or ranges[43].

- **Pruning Mechanism**: Pruning is a technique used to prevent overfitting in decision trees. It involves reducing the complexity of the tree by removing unnecessary branches. Some decision tree algorithms incorporate pruning mechanisms to optimize the tree's structure and improve generalization on unseen data. Pruning can be done using methods like reduced error pruning, cost-complexity pruning, or minimum description length[43].

It's important to note that different decision tree algorithms may have variations and specific implementations of these characteristics. The choice of a particular algorithm depends on the specific requirements of the problem, the nature of the

data, and the desired trade-offs between accuracy, interpretability, and computational complexity. Here are some of the key characteristics of decision trees:

- **Clear rule representation**: Decision trees are constructed using a series of if-else rules that partition the data based on feature values. These rules are straightforward and can be presented in a human-readable form, making it easy to interpret how the model arrives at its decisions[28].

- **Feature importance**: Decision trees provide insights into the importance of different features in the decision-making process. By examining the splits and branching points, it is possible to identify which features have the most significant influence on the predictions. This information can help understand the underlying patterns and relationships in the data[28].

- **Visualization**: Decision trees can be visualized graphically, which aids in comprehending the model's structure and logic. Visual representations, such as tree diagrams, show the flow of decisions, feature values, and prediction outcomes, facilitating interpretation and communication of the decision-making process[25, 28].

- **Explanation of predictions**: Since decision trees are based on explicit and transparent rules, individual predictions can be easily explained. By tracing the path from the root node to the leaf node corresponding to a particular prediction, one can understand the specific conditions and criteria that led to that prediction. This interpretability is valuable in domains where the reasoning behind decisions is critical, such as healthcare or finance[3, 28].

- **Error analysis**: Transparency in decision trees allows for analysis of prediction errors. By examining the misclassified instances and their corresponding rules, it is possible to gain insights into the limitations or weaknesses of the model. This analysis can guide improvements in the data or the decision tree itself[28].

- **Model validation and evaluation**: Transparency enables researchers, practitioners, and stakeholders to evaluate the decision tree model thoroughly. The interpretability of the rules allows for manual verification and validation of the model's correctness, making it easier to detect errors, biases, or ethical concerns[28].

- **Regulatory and legal compliance**: In domains where transparency, trust, and accountability are critical, such as finance or healthcare, decision trees' transparency can help meet regulatory and legal requirements. The ability to explain decisions and provide transparent reasoning can ensure compliance and facilitate audits or regulatory assessments[28].

2.2.2 Rule induction

Rule induction algorithms are designed to generate rules that describe a specific concept of interest. These rules consist of conditions based on the attributes that describe the examples. There are different approaches and methods for rule induction, and here are a few examples:

- **C4.5-Rules[35] and PART[15]**: These rule induction methods rely on decision trees.They analyze the decision tree structure and extract rules that represent the patterns and decisions made by the tree

- **Sequential Rule Induction**: Some rule induction algorithms adopt a separate-and-conquer strategy to build rule sets. The process starts with an empty rule set, and rules are sequentially constructed by selecting attributes and conditions that maximize their coverage of the training examples. After each rule is built, the examples covered by that rule are removed from the training set, and the process continues to build the next rule[18]. This iterative approach allows the rule set to separate examples belonging to the class of interest from the rest.

- **IREP and Ripper**: IREP[19] is an example of a rule induction algorithm that builds rules from scratch. It follows the separate-and-conquer strategy mentioned earlier. Cohen[11] proposed Ripper as an improvement over IREP, aiming to enhance the discriminating capacity of the generated rules while maintaining computational efficiency.

These rule induction algorithms can extract interpretable rules from data, enabling insights and understanding of complex patterns and decision-making processes. By formulating rules based on attributes, these algorithms provide a structured representation of knowledge that can be utilized for classification, prediction, and decision support.

2.2.3 Ensembles of trees

Decision trees are very sensitive to their training sample. A small change in the training sample can lead to the construction of a completely different tree using the same algorithm. This property is called instability[9]. To enhance the performance and stability of decision trees, a common approach is to construct ensembles of classifiers. Ensembles are composed of multiple individual classifiers whose decisions are combined to produce a final prediction. The use of ensemble classifiers often yields improved results compared to using a single classifier[5]. However, it is important that the individual classifiers within the ensemble disagree with each other [40], as this diversity contributes to the ensemble's effectiveness.

One challenge with ensemble classifiers is their lack of explainability. They are considered black box models because the process of combining the decisions made by individual classifiers obscures the underlying factors that influence the final prediction. Each individual classifier within the ensemble may have a distinct internal structure and different decision-making mechanisms. As a result, it becomes difficult to interpret and understand the specific reasons behind the ensemble's collective decision-making process.

Two well-known ensemble methods for decision trees are Bagging and Boosting. Bagging [8], short for bootstrap aggregating, involves creating multiple subsets of the original training data through sampling with replacement. Each subset is then used to train a separate decision tree classifier. The final prediction is obtained by aggregating the predictions of all the individual trees in the ensemble. Boosting [40], on the other hand, is an iterative approach that assigns weights to training examples and focuses on learning from the mistakes made by previous classifiers. Each subsequent classifier in the boosting process is designed to correct the errors of the previous ones, leading to an ensemble with improved overall performance.

Ensemble classifiers have proven to be effective in improving the accuracy and robustness of decision tree models. However, it is important to note that the trade-off for their improved performance is the loss of explainability. Despite this limitation, ensembles remain widely used and valued in various machine-learning applications. In order to address the lack of explaining capacity in ensemble classifiers, researchers have developed approaches to incorporate explanations into the ensemble framework. One such example is the Combined Multiple Models (CMM). approach[12]. CMM starts by creating a Bagging ensemble, where multiple decision tree classifiers are trained on different subsets of the original training data. Then, the ensemble is used to classify both the original training examples and additional artificially created examples. The labeled examples, consisting of the original examples and their corresponding labels assigned by the ensemble, are used to train a separate classifier, such as a C4.5-Rules rule set. This additional classifier aims to provide explanations for the ensemble's decisions.

Another approach that combines explaining capacity with ensemble methodology is the consolidation of decision tree algorithms [33]. This approach improves the discriminating capacity and stability of individual decision trees while preserving their explaining capacity. The ensemble methodology is applied during the tree-building process, resulting in what is referred to as "Inner Ensembles" [1]. By

leveraging the ensemble framework, the consolidation of decision tree algorithms enhances the performance of simple decision trees while retaining their ability to provide explanations.

2.3 Background for Consolidated Tree Algorithm:

The Consolidated Tree Construction (CTC) algorithm[32] was developed to address the specific problem of car insurance fraud detection, which involved a high degree of class imbalance and required explanations for the decisions made by the model. Traditional ensemble algorithms, such as Bagging and Boosting, have shown improved performance compared to individual classifiers. However, they lack explainability, as the decision-making process becomes obscured due to the combination of multiple classifiers with different internal structures.

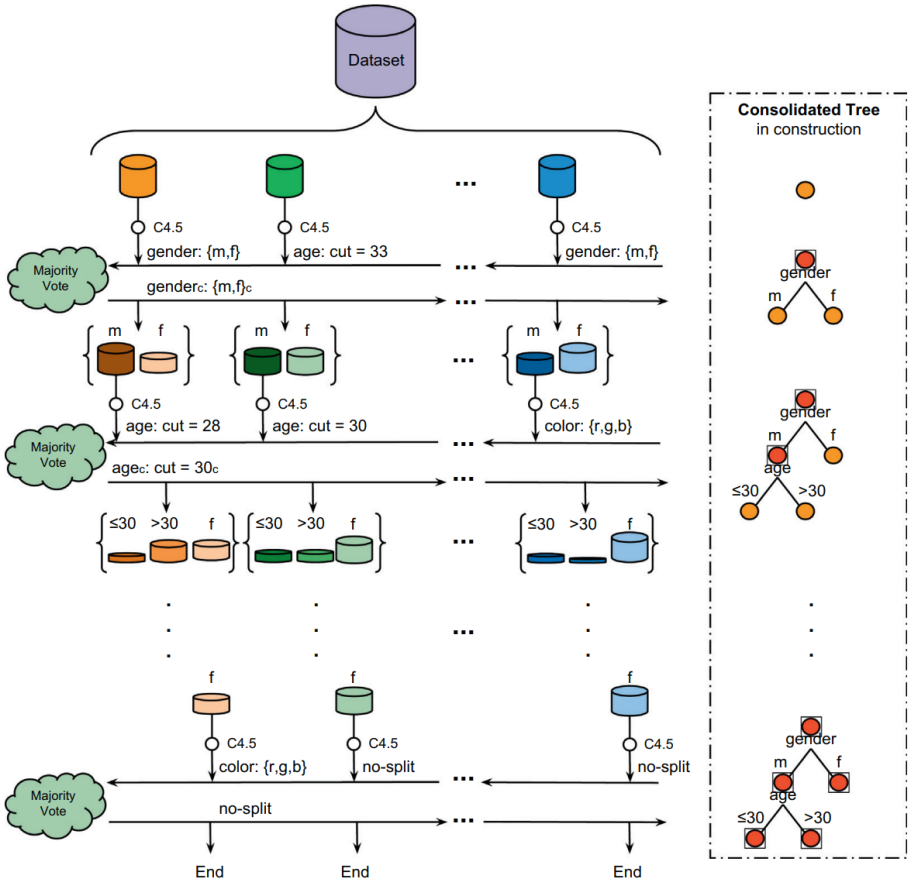
CTC tackles this challenge by constructing a set of subsamples from the training data and building a decision tree using votes from each subsample. Unlike independent tree construction in traditional ensembles, CTC involves a voting process in which the decision on each split is determined by the majority vote of all subsamples as shown in Figure 2. This ensures that all subsamples make the same split regardless of their individual preferences. The process is repeated iteratively until the stop condition is met. The resulting tree, while built using multiple subsamples, is a simple decision tree, which can be easily understood by humans. This distinguishes CTC from traditional ensemble algorithms, which typically produce complex models.

To delve into the details of the CTC algorithm, readers can refer to the comprehensive explanation provided in[32]. The CTC algorithm is available as an official package in WEKA[48], a popular machine-learning software suite, under the name J48Consolidated[22].

When using consolidated trees, the first re-sampling strategies involved building stratified subsamples that were proportional in size to the original training sample (typically set at 75 %)and also utilizing bootstrap samples. While CTC performed

Figure 2

Representation of CTC algorithm



This data can be found in [20]

better with bootstrap samples in some databases, stratified subsamples generally yielded superior results[30]. Comparisons were made between CTC, C4.5, Bagging, and Combined Multiple Models (CMM). The results indicated that CTC performed better than C4.5 but fell between Bagging and CMM[31].

Weiss and Provost[47] proposed that the class distribution present in the training sample might not be optimal for the problem at hand. They suggested an optimal class distribution for several datasets when using the C4.5 algorithm. Subsequently, Perez et al. [33] tested these datasets using the CTC algorithm with both the original class

distribution and the proposed optimal distribution. Their findings revealed that the optimal class distribution for C4.5 and CTC were not the same. In a later extension of their work[31], the researchers aimed to determine the optimal class distribution for CTC across 30 datasets. The experiment involved combining class distributions ranging from 2% to 98% and exploring multiple values for the number of samples (N_S) parameter between 5 and 50. The results indicated that the best performance was achieved when using balanced or nearly balanced class distributions[31].

CTC balances the performance of ensemble algorithms like Bagging and the explainability of individual decision trees. Researchers have explored different re-sampling strategies and investigated the optimal class distribution for CTC, highlighting the importance of balanced class distributions for achieving optimal results.

2.4 Background work on bootstrapping rule induction:

The objective of rule induction algorithms is to extract rules from datasets, revealing patterns and relationships within the data. However, these rule sets often become intricate and sensitive to data variations, resulting in reduced stability and interpretability. To tackle these challenges, [45] proposes a new method called bootstrapping rule induction which utilizes bootstrapping, a re-sampling technique, in rule induction.

To motivate boostrapping applied to rule induction, consider the domain of perioperative medicine, in which a total of 34,926 adverse events, referred to as "outcomes," have been identified for evaluating the post-surgical status of patients undergoing general anesthesia[14]. While serious injuries are infrequent, minor incidents can lead to additional costs, delays, patient discomfort, or prolonged recovery. Thus, perioperative knowledge discovery in databases (KDD) aims to identify at-risk sub-populations for adverse outcomes, enabling the implementation of preoperative safeguards. Rule induction aligns with this objective, focusing on positive examples representing surprising occurrences or anomalies to be monitored. This differs from classifiers, such as

decision trees, which categorize all data into mutually exclusive outcomes. Classifiers may prioritize overall classification accuracy at the expense of isolating abnormalities if most examples are negative.

However, existing rule induction systems in perioperative data analysis and similar domains suffer from certain drawbacks. These include the absence of measures for variance in rule accuracy and continuous attribute decision boundaries, which diminishes domain experts' trust in the discovered knowledge. It is valuable to know the standard deviation around a rule involving weight, such as a boundary of 67 kilograms, being 3 kilograms rather than 10[45]. Additionally, rule instability manifests in different rules being learned with small changes to the training data, including alterations to attribute decision boundaries, further eroding trust. The discovery of numerous rules, some of which are similar, poses challenges for domain experts in filtering and interpretation.

These limitations can be handled by introducing variance measures, mitigating rule instability, and facilitating rule filtering and interpretation. Bootstrapping rule induction aims to enhance the trustworthiness and usefulness of rule induction in perioperative data analysis and related domains[45].

Bootstrapping involves resampling the original dataset to create multiple bootstrap samples. These samples are then used to induce rules, and the results are combined to create a stable and reduced rule set. The aim is to improve the stability of the regulations by reducing their sensitivity to variations in the data. Experiments were conducted to evaluate the effectiveness of bootstrapping on different datasets and compare it with other rule induction methods. They analyze various measures, including rule stability, rule coverage, rule complexity, and rule accuracy, to assess the performance of bootstrapping. The results demonstrate that bootstrapping can significantly improve rule stability and reduce the number of rules while maintaining or enhancing rule accuracy. The technique is particularly effective in dealing with datasets that contain noise or have a high level of variability. The findings suggest

bootstrapping can help extract reliable and concise rules from complex datasets, making them more interpretable and robust.

2.5 Probabilistic decision tree

As mentioned in the earlier chapters, in certain domains like medicine where humans are licensed and not a machine, there is a need to increase trust in the predictive models as clinicians would like to verify results with their domain knowledge. In such scenarios, there is a need for a model to make sense in a way that the end user can believe and verify the results. This can be achieved only when the model is transparent, explainable, and believable. Probability split point decision trees address these significant concerns, as mentioned in this section.

Transparency in decision trees refers to the ability to interpret and understand the model's decision-making process. Decision trees are known for their inherent transparency, which means that the rules and criteria used to make predictions can be easily understood and explained. Transparency in decision trees empowers users to understand, validate, and trust the model's predictions. It promotes accountability, facilitates decision-making, and encourages adoption in domains where interpretability and explanation are vital. However, decision trees have the following drawbacks:

- **Lack of probabilistic output:** Decision trees produce deterministic predictions by assigning a single class label to each leaf node. They do not inherently provide probabilities or confidence levels associated with each prediction. This lack of probabilistic output can limit the transparency of the model, especially when assessing prediction uncertainties[21].

- **Complex decision rules**: While decision trees offer rule-based explanations, large and deep trees can generate complex decision rules with numerous conditions. As decision trees grow, the interpretability of the model diminishes[21],

making it more challenging to understand and communicate the decision-making process to non-technical stakeholders.

- **Sensitivity to data variations**: Decision trees are sensitive to changes in the training data. Minor modifications or additions to the dataset can result in different tree structures and decision rules, impacting the interpretability and stability of the model. Suppose the variability of the data being classified significantly deviates from the patterns observed in the training data[31]. In that case, the decision tree may need help to accurately classify or predict the outcomes, as it may have yet to learn the new patterns or variations. This sensitivity reduces transparency and may require additional effort to validate and explain the model's predictions[33].

- **Lack of global optimization**: Decision trees are constructed using a greedy, recursive algorithm that optimizes local decisions at each node. However, this local optimization may not produce the globally optimal tree structure. As a result, decision trees may not consistently provide the most accurate or interpretable models for complex datasets[26].

- **Model accuracy and interpretability trade-off**: Achieving high accuracy in decision trees may require sacrificing interpretability. Introducing additional complexity or employing advanced techniques like ensemble methods can enhance accuracy but may compromise the transparency of the model[38].

Consolidated tree construction uses a re-sampling technique to build a tree. This technique is different from bagging, boosting, etc., as consensus is achieved at each step of the tree-building process to build a structurally unified tree, giving stability to the explanation and with a low error rate [31].

Predicting uncertainties in a classification model is essential because it provides valuable information about the reliability and confidence of the model's predictions. Uncertainty estimation helps decision-makers understand individual predictions' potential errors or ambiguity. This information is crucial in situations where the consequences of misclassification can be significant, such as in healthcare or finance. By quantifying uncertainties, users can make more informed decisions, allocate resources effectively, and take appropriate actions to mitigate risks. Additionally, uncertainty estimation fosters trust and transparency in the model, allowing users to gauge the reliability of its outputs and make better-informed decisions based on the level of uncertainty involved.

Bootstrapping sampling techniques can generate multiple decision trees by resampling the training data or perturbing the feature space. By aggregating the predictions of these sampled decision trees, obtaining a distribution of predictions is possible, which can provide insights into uncertainties. Waitman's bootstrapping using rule-based induction uses this probabilistic idea to improve explainability, transparency, and probabilistic reasoning. Waitman and et al. experiment with different bootstrapping approaches and evaluate their effectiveness using real-world datasets. They compare the consolidated rule sets obtained through bootstrapping with those generated without bootstrapping. The results demonstrate that bootstrapping improves rule stability, reduces the complexity of rule sets, and enhances interpretability.

CHAPTER 3

METHODOLOGY AND DESIGN

In this chapter, we discuss the datasets we used to evaluate our algorithm. We also describe the PSPDT algorithm and the various hyper-parameters considered in this initial implementation of PSPDT.

3.1 Data

We applied the PSPDT algorithm to five different datasets. Most of our datasets were obtained from Kaggle[24], a well-known platform in the scientific community that hosts various datasets, including those from the widely used UCI repository[24]. One of the primary datasets used in our research is the famous Iris flower dataset. It contains 150 samples, with 50 samples each from three different species of Iris: Iris setosa, Iris virginica, and Iris versicolor. Each entry has measurements for four features, namely the length and width of the sepals and petals, measured in centimeters[49].

Additionally, we utilized a well-known diabetes dataset to classify and predict whether a patient is diabetic. The dataset contains various diagnostic measurements, including the number of pregnancies, BMI, insulin level, and age. It focuses on diagnosing diabetes in females of at least 21 years old from the Pima Indian heritage. We selected 2,000 samples from the original dataset to reduce computational costs[34].

The wine dataset, with around 200 samples, was also employed to predict the type of wine. This dataset contains information from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It includes 13 different constituents measured for each type of wine. Furthermore, we employed the Wisconsin breast cancer diagnostic dataset to predict whether a tumor is benign or malignant. This dataset consists of approximately 570 data samples, with 30 features such as radius, texture, area, smoothness, and compactness[6, 51].

3.2 PSPDT Algorithm

The aim of our research is to create a basic open-source PSPDT algorithm. For that purpose, we have developed an initial PSPDT implementation in Python. For the experiments we have used only numerical datasets since the probabilistic aspects of algorithm only apply to numeric features.

The PSPDT algorithm uses a consolidated tree construction algorithm [45] along with a bootstrapping rule-induction process to develop a probabilistic decision tree model. Several training sub-samples are extracted from training data, and a single tree is built based on consensus among voting of the subsamples. The decision regarding which variable will be used to make a split at a node in the decision tree is determined by considering multiple proposals originating from the subsamples (evaluating features using CART's Gini impurity measure). This selection process is conducted through voting, where each subsample suggests a splitting variable. By aggregating these votes, a final decision is made on the variable for the split at that specific node. Once the decision on the split variable is made, each subsample is forced to split using that variable. The collection of split points across the subsamples is used to estimate a Gaussian distribution of split points by computing a mean and standard deviation. The process is repeated recursively until no more decisions are possible or minimum depth or min leaves. The schema of PSPDT is in Figure 3, and it proceeds in the following way:

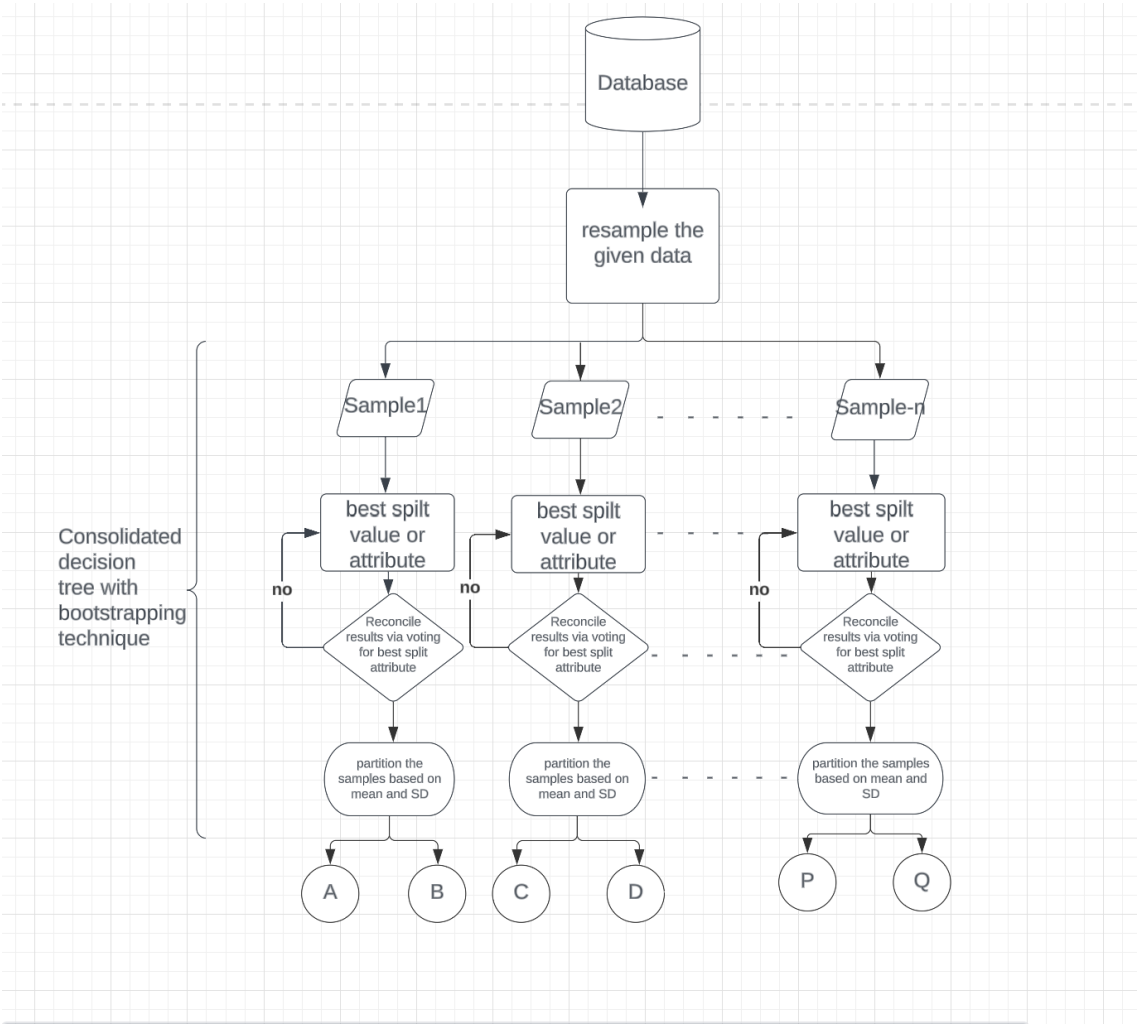
1. Partition the data into train and test datasets.

2. Use bootstrapping to create multiple samples. Extract a set of subsamples (Number_samples) from the original training dataset using the desired resampling technique, size of the subsamples, etc. with replacement or without replacement or stratified, etc.

3. At consolidated node is built as below:

Figure 3

Representation of PSPDT algorithm



a) At each node, make all the subsamples vote for the best attribute.

b) Attribute with the majority of the votes is used to split across all the samples, as shown in Figure 3.1

c) If appropriate, make it a leaf node. Otherwise, split each subsample using the winning attribute (with majority votes)

d) We get all the values for that attribute from each subsample, and that defines our distribution of split points for that node.

4. Store the median, mean, and standard deviation at each consolidated node

5. This process is repeated until it can no longer split (based on hyperparameters such as maximum depth of the tree, minimum samples, etc.), making it a leaf node.

6. For classification, we use the likelihood function based on induced Gaussian distributions to compute the probability that the test data point is classified down each path in the tree to compute the probability that the data point should be classified to each leaf in the tree. The probability at a leaf is the product of probabilities along each of the paths from the root node to the corresponding leaf node thus, presenting us with a probabilistic tree. The final prediction is selected by computing the most likely class based on the selected leaves. Leaf selection options are described below.

3.2.1 Gaussian distribution

A probability distribution is a statistical concept that defines the likelihood or probability of observing different values that a random variable can take. In the context of probability theory and statistics, a probability distribution provides information about the possible outcomes and their associated probabilities. It describes the relative likelihood of each value or outcome occurring, allowing us to understand and quantify

uncertainty in various phenomena. For example, if we have 20-30 patients with various insulin levels and if we were asked to pick a patient randomly and asked what their insulin level was, it would be impossible to guess. If, however, we were presented with the distributions of the insulin levels of all the patients, we can compute an expected value.

A normal (or Gaussian) distribution is one such probability distribution. Other functions can be used to estimate the distribution of the data, but the Gaussian distribution is the easiest to work with because we only need to estimate the mean and the standard deviation from the training data. The mean is the average, and the standard deviation is a measure of how far the data is spread out. The probability density function for normal distribution is

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where

$$\mu = mean$$

$$\sigma = standard\ deviation$$

$$x = input\ value$$

3.2.2 Probabilistic classification

In our project, you have primarily utilized two techniques for classification. The first technique is called Most Probable Leaves (PSPDT-MP), where the classification of a data sample is determined based on the leaf with the highest probability among all the leaves. The second technique is called ϵ based classification (PSPDT-ϵ). In this approach, a hyperparameter ϵ is used. All the leaves with probabilities greater

than ϵ are considered, and a weighted average is calculated for each class present. Subsequently, the class with the highest value is selected as the prediction for that data sample.

At each node, the split points for the attribute follow the normal distribution. The data sample is classified based on where it falls in the normal distribution curve. When working with the normal distribution in scipy, the scipy.stats.norm[41] class provides various methods for calculating probabilities and working with the normal distribution. One of the methods is norm.cdf(), which allows you to calculate the cumulative probability density value for a given data point. The cumulative probability value from -∞ to ∞ will always be equal to 1. To use norm.cdf(), we need to provide the data point, mean (loc), and standard deviation (scale) as input arguments. The method will then return the probability density value for that data point according to the normal distribution. The loc parameter represents the mean of the distribution, and the scale parameter represents the standard deviation. By specifying the mean and standard deviation, we determine the position and shape of the normal distribution. The probability for that data instance less than or greater than the mean split value of that attribute is calculated using the probabilities from the cumulative density function obtained. This process is repeated iteratively until no further divisions are possible(based on hyper-parameters) for the sample making it a leaf node. For each leaf, the cumulative probability at the leaf is calculated by multiplying all the probabilities along the path from the root node to the leaf. Thus presenting a probabilistic tree for each of the data samples. The final tree consists of the leaf nodes with cumulative probability and the corresponding predicted class. The leaf with the highest probability is considered as the most probable leaf along with the predicted class for that data sample.

3.2.3 Hyperparameters

Hyperparameters are essential for controlling the behavior of the algorithm and, thus, the structure of decision trees. When working with decision trees, adjusting the hyperparameters becomes necessary to optimize their performance for a specific task and dataset. Tuning hyperparameters is a vital process in finding the right configuration for decision trees, aiming to achieve the best possible performance and generalization of the model. In the context of PSPDTs, the following hyperparameters are used:

- Maximum Depth: This hyperparameter determines the maximum depth or levels of the decision tree. It controls the number of splits and branches the tree can have. Setting a higher maximum depth may result in over-fitting, while a lower value may lead to under-fitting.

- Minimum Sample Split: This hyperparameter specifies the minimum number of samples required to split an internal node. If the number of samples at a node falls below this threshold, further splitting is not performed. Adjusting this hyperparameter helps control the complexity of the tree and can prevent over-fitting.

- Minimum Sample Leaf: This hyperparameter sets the minimum number of samples required to be at a leaf node. If the number of samples at a leaf node is below this value, a split is not allowed. Similar to the minimum sample split, this hyperparameter helps control the complexity and generalization of the tree.

- Number of subsamples: This hyperparameter determines the number of subsamples that the training dataset will be split into.

By adjusting these hyperparameters, researchers and practitioners can find the right balance between model complexity and generalization, ultimately improving the performance of decision trees for a given task and dataset.

3.3 Experimental setup

Four databases have been in our experiments. The datasets used in the experiments were divided into test and train datasets using a stratified resampling method with a percentage split of 70% and 30%. This approach ensures that the distribution of classes in both the test and train datasets is balanced. Stratified resampling is a technique commonly used when dealing with imbalanced datasets, where the distribution of classes is uneven. Table 1 shows the various domains and their corresponding characteristics for each dataset, including the number of samples (N. of Samples), the number of features (N. of features), and the number of classes of the dependent variable (N. of classes).

Table 1

Description of Experimental Datasets

Description of experimental domains			
Domain	N. of Samples	N. of Features	N.of classes
IRIS	150	5	3
Breast-W	570	32	2
Diabetes	2000	9	2
Wine	170	14	3

The PSPDT algorithm is then compared with CTC-based decision trees and also with traditional decision trees (CART) using the same hyperparameter settings. All three trees are constrained using the same max-depth hyperparameter setting. Each experiment was run 10 times with reported results being the average across the 10 runs.

A novel PSPDT algorithm from scratch and conducting PSPDT learning on each of the four datasets. We divided the data into train and test sets (70/30) to conduct classification experiments. We experimented with 2 different scenarios after testing the dataset. the first scenario is where we have all the probable leaves for each data sample and the most probable leaf is considered for predicting the class and the for the

second scenario we have calculated the predicted class by using the weighted average of all the leaves greater than a certain value, ϵ where $0 \leq \epsilon \leq 1$.

We then calculated the accuracy for each of the datasets. We also calculate the minimum, maximum, mean, and standard deviation of the number of leaves used in the classification for the datasets where the ϵ is greater than or equal to 0.01, 0.05, 0.1, and 0.25 to understand the impact of probabilistic classification on performance across various ϵ values.

CHAPTER 4
EXPERIMENTAL RESULTS AND ANALYSIS

In this chapter, we present and analyze the results obtained from the experiments described in Chapter 4. First, we present the performance metrics used to evaluate our algorithm. Then we discuss the performance of the classifier. Finally, we describe how the introduction of the probabilistic approach has influenced the performance of decision trees.

4.1 Measuring the accuracy of PSPDTs

Classification accuracy is a commonly used performance metric for evaluating classification models in machine learning. It provides a measure of how well the model predicts the correct class labels for a given set of samples. Accuracy is the fraction or percentage of correct predictions made by the model. In its formal definition, accuracy is the ratio of the number of correct predictions to the total number of predictions made by the model [10]. It is a straightforward metric that indicates the overall correctness of the model's predictions. One popular approach is the use of a confusion matrix, which provides a detailed breakdown of the predictions made by the model, including true positives, true negatives, false positives, and false negatives.

$$\textbf{Accuracy} = \frac{\textbf{Number of correct predictions}}{\textbf{Total number of predictions}}$$

For binary classification, the accuracy can also be calculated in terms of the prediction value (positive and negative) of the classifier.

$$\textbf{Accuracy} = \frac{\textbf{TP} + \textbf{TN}}{\textbf{TP} + \textbf{TN} + \textbf{FP} + \textbf{FN}}$$

Here, TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative.

In our experiments, we used accuracy as the primary metric to evaluate the performance of the PSPDT algorithm. Our main focus was to compare the performance of PSPDTs with that of CTC and CART to see if the algorithms' approach to estimating rule probability had any statistically significant impact on accuracy.

In order to assess if PSPDTs have an average accuracy that is not statistically significantly different (at the $p = 0.05$ level) than traditional decision trees built using CART or CTC with similar hyperparameters, we use the t-test. A t-test is a statistical test used to compare the means of two groups and determine whether there is a significant difference between them. It is commonly used in hypothesis testing to evaluate whether a process or treatment has an effect on a population or to compare two groups for differences[4]. In our work, we calculate the accuracies for 10 runs for PSPDTs, CTC, and CART for the 4 datasets. We then evaluate the t-test measure for PSPDT vs. CTC and PSPDT vs. CART algorithms. If the p-value is less than 0.05 we reject the null hypothesis which is that PSPDTs have an average accuracy that is not statistically significantly different (at the $p = 0.05$ level) than traditional decision trees built using CART or CTC with similar hyperparameters.

4.2 Testing the Gaussian distribution assumption

In order to address see if the split points are following a Gaussian distribution, we use Gaussian hypothesis testing. Gaussian distribution hypothesis testing is a statistical method used to assess whether a given data set follows a Gaussian (normal) distribution or not. The hypothesis testing process involves formulating null and alternative hypotheses and using statistical tests to evaluate the evidence against the null hypothesis[16]. The goal is to make inferences about the population parameters based on the available sample data. Normality tests are employed to evaluate how likely it is for a random variable underlying the data to be normally distributed.

The Shapiro-Wilk test is a statistical test used to determine if a sample comes from a normally distributed population. It was developed by Samuel Sanford Shapiro

and Martin Wilk in 1965 [50]. The test calculates a test statistic based on the sample data, and the null hypothesis is that the sample comes from a normally distributed population. The test statistic for the Shapiro-Wilk test is calculated using coefficients and a covariance matrix. The cutoff values for the test statistic are determined through Monte Carlo simulations. The interpretation of the test relies on the p-value obtained from the test. If the p-value is less than the chosen alpha level (typically 0.05), the null hypothesis is rejected, suggesting that the data is not normally distributed. In this test at each node, we calculate the p-value and check if the value is greater than 0.05 for our null hypothesis to be accepted, suggesting that the data does follow a Gaussian (or normal) distribution[50].

4.3 Performance of PSPDT-MP Vs. CTC Vs CART

The results presented in Table 4.1 indicate that the accuracies achieved by PSPDTs for all four datasets are comparable to those of traditional decision tree algorithms, specifically CTC (Classification and Regression Trees) and CART (C4.5). To further evaluate the performance of PSPDTs, a t-test was conducted to determine if they are statistically significantly different than decision trees built using CART or CTC with similar hyperparameters. In our experiments, accuracies were generated for 10 runs of PSPDTs, CTC, and CART algorithms for each dataset. The hyperparameters used were a maximum depth of 10 and a minimum number of leaves set to 3. The t-test was performed to compare PSPDTs against CTC and CART algorithms.

The p-values obtained were never less than 0.05 when comparing PSPDTs to CTC. The lowest p-value of 0.42 was observed for the IRIS dataset, while the highest p-value of 0.92 was obtained for the breast-w dataset. On the other hand, when comparing PSPDTs to CART, the p-values were lower for CART than for CTC. Overall, the p-values were greater than 0.05, with the lowest value of 0.057 for the wine dataset and the highest value of 0.468 for the diabetes dataset.

Based on these findings, we can accept the null hypothesis that PSPDTs are not statistically significantly different than traditional decision trees built using CART or CTC with similar hyperparameters. In other words, there is no strong evidence to suggest that PSPDTs perform significantly different than CART or CTC when considering the accuracies and p-values obtained for all datasets.

Table 2 presents the distribution of the leaves for each of the ϵ values for all four datasets.

Table 2

Performance measure(accuracy and t-test summary) PSPDT Vs CTC Vs CART

Accuracy and t-test summary - PSPDT Vs CTC Vs CART			
	PSPDT	CTC(p-value)	CART(p-value)
IRIS	89.5%	84.97% (0.43)	91.12% (0.12)
Wine	87.56%	88.97% (0.92)	90.53% (0.06)
Diabetes	71.25%	73.86% (0.86)	80.64% (0.47)
Breast -W	89.75%	87.32% (0.93)	90.56% (0.11)

4.4 PSPDT-ϵ

The results of our experiments, as depicted in Tables tables 3 to 6, show the accuracies and distribution of leaves for various ϵ values across all four datasets. The tables include PSPDT-MP and CART results for easy comparison.

Accuracies across different ϵ values for each dataset have shown varying performance, with some ϵ values outperforming others, while overall impacting the performance of the data model. Consequently, it is necessary to tune the ϵ value similar to other hyperparameters. It is worth noting that, except for the diabetes dataset, some ϵ value has consistently outperformed the Classification and Regression Trees (CART) algorithm for every dataset.

For the IRIS dataset, the maximum number of leaves decreased from 4 to 2, and the average number of leaves decreased from 2.43 to 1.26 as the ϵ value increased

from 0.01 to 0.25. Similarly, for the Wine dataset, the maximum number of leaves decreased from 6 to 2, and the average number of leaves decreased from 2.16 to 1.22 with increasing ϵ value. In the case of the Breast-w dataset, the maximum number of leaves decreased from 5 to 2, and the average number of leaves decreased from 1.75 to 1.14. For the diabetes dataset, the maximum number of leaves reduced from 5 to 3, and the average number of leaves decreased from 1.87 to 1.12 as the ϵ value increased. Such a trend is consistent with what we expect. More leaves are likely to have probabilities of at least ϵ when its value is lower, and this is indeed reflected in the maximum and average number of leaves satisfying this condition. Likewise, as the range decreases, the standard deviation does as well. This too is expected.

It is worth noting that for all the datasets, there is at least one leaf for each ϵ value. In theory, however, a tree might not have any leaves with a probability above ϵ. This was not observed in our experiments. To account for this potential issue, however, we would need to extend our algorithm to, perhaps, use the most probable leaf, regardless of its probability, if no leaf meets the ϵ threshold.

Table 3

Distribution of leaves for various ϵ values for IRIS dataset

Leaf distribution and accuracy for ϵ values IRIS dataset					
	Mean	std dev	Min	Max	Accuracy
$\epsilon = 0.01$	2.43	0.96	1	4	93.25%
$\epsilon = 0.05$	1.93	0.67	1	3	90.85%
$\epsilon = 0.1$	1.76	0.55	1	3	86.57%
$\epsilon = 0.25$	1.26	0.44	1	2	89.56%
Acc(PSPDT-MP)	-	-	-	-	89.5%
Acc(CART)	-	-	-	-	91.12%

Table 4

Distribution of leaves for various ϵ values for Wine dataset

Leaf distribution and accuracy for ϵ values Wine dataset					
	Mean	std dev	Min	Max	Accuracy
$\epsilon = 0.01$	2.16	1,16	1	6	83.95%
$\epsilon = 0.05$	1.66	0.78	1	4	81.23%
$\epsilon = 0.1$	1.47	0.60	1	3	92.84%
$\epsilon = 0.25$	1.22	0.415	1	2	93.84%
Acc(PSPDT-MP)	-	-	-	-	87.56%
Acc(CART)	-	-	-	-	90.53%

Table 5

Distribution of leaves for various ϵ values for Diabetes dataset

Leaf distribution and accuracy for ϵ Diabetes dataset					
	Mean	std dev	Min	Max	Accuracy
$\epsilon = 0.01$	1.87	1.12	1	5	74.52%
$\epsilon = 0.05$	1.63	0.86	1	4	68.48%
$\epsilon = 0.1$	1.46	0.70	1	3	69.87%
$\epsilon = 0.25$	1.12	0.46	1	3	74.87%
Acc(PSPDT-MP)	-	-	-	-	71.25%
Acc(CART)	-	-	-	-	80.64%

Table 6

Distribution of leaves for various ϵ values for Breast-w dataset

Leaf distribution and accuracy for ϵ Breast-w dataset					
	Mean	std dev	Min	Max	Accuracy
$\epsilon = 0.01$	1.75	0.95	1	5	93.71%
$\epsilon = 0.05$	1.47	0.66	1	4	89.54%
$\epsilon = 0.1$	1.32	0.53	1	3	90.26%
$\epsilon = 0.25$	1.14	0.36	1	2	87.66%
Acc(PSPDT-MP)	-	-	-	-	88.75%
Acc(CART)	-	-	-	-	90.56%

4.5 Gaussian distribution test results

After evaluating the distribution of split points at each node using the Shapiro-Wilk test for normality, we obtained p-values that were never less than 0.05, with the lowest p-value recorded at 0.132. Consequently, we accepted the null hypothesis, which supports the assertion that the split points follow a Gaussian or normal distribution. Throughout our experiments, the Shapiro-Wilk test consistently demonstrated p-values greater than or equal to 0.05, indicating that the observed split points conform to a Gaussian or normal distribution.

Overall, the results suggest that PSPDTs perform comparably to traditional decision tree algorithms (CTC and CART) in terms of accuracy. Additionally, the distribution of split points follows a Gaussian or normal distribution, supporting a fundamental assumption of PSPDTs.

CHAPTER 5

SUMMARY AND FUTURE WORK

5.1 Conclusion

The Probability Split Point Decision Tree (PSPDT) algorithm was introduced as a means to enhance trust and transparency in classification decisions, particularly in the field of healthcare. In particular, it aims to bridge the gap between the structure of decision trees and the mental models of clinicians. This algorithm should enhance stability, probabilistic reasoning, and modeling of real-world uncertainties while retaining the transparency and interpretability of decision trees. It induces Gaussian distributions over split points to model probabilistic decision boundaries, aiming to better reflect the real-world variability and align more closely with the mental models of domain experts. The PSPDT algorithm provides a measure of variance for continuous attribute decision boundaries and offers probabilistic decision boundaries that are more consistent with real-world scenarios.

We introduced a novel supervised, classification machine learning algorithm called PSPDT. Our algorithm specifically focused on numerical datasets without any missing values. To assess the performance of the PSPDT algorithm, four different datasets were used and compared against decision trees built using the CART and CTC algorithms. The accuracies obtained were averaged across 10 runs for each dataset and all algorithms, ensuring a fair comparison with the same hyperparameters. We have observed that there was not a statistically significant difference in the accuracies for all three algorithms although accuracy using CART was highest for all the datasets when PSPDT only used a single leaf node. Additionally, Shapiro-Wilk hypothesis testing was conducted at each node for all the datasets, consistently resulting in a p-value greater than 0.05, indicating that the split points follow a normal distribution.

Furthermore, the impact of leaves on the overall performance of the PSPDTs was investigated by conducting experiments with four different values of ϵ-0.01, 0.05, 0.1 and 0.25 (a hyperparameter) to consider a more probabilistic classification approach. A weighted average was computed to predict the classification, considering a given ϵ value.

5.2 Future Work

Regression modeling is a valuable area for future work in extending this research. The current implementation of PSPDTs is considered naive and would benefit from further research to enhance its capabilities. One important aspect to consider in the extension of PSPDTs is the handling of missing values. Developing algorithms that can effectively handle missing values would improve the robustness and applicability of the model[23]. Additionally, research can be invested in making PSPDTs more cost-effective.

Accommodating various sampling methods as hyperparameters would be another worthwhile addition to the algorithm. By allowing flexibility in the sampling process, researchers and practitioners can adapt the algorithm to different scenarios and data characteristics. This flexibility can enhance the model's performance and applicability in diverse contexts.

Furthermore, it would be valuable to assess the impact of PSPDTs on tree stability and probability estimates. Understanding how PSPDTs influence the stability of decision trees would provide insights into the reliability and consistency of the model's predictions. Additionally, evaluating the probability estimates generated by PSPDTs would contribute to assessing the model's calibration and its ability to quantify uncertainty accurately. Expanding the capabilities of the algorithm to handle mixed-type data would also be beneficial.

REFERENCES

[1] Abbasian, H., Drummond, C., Japkowicz, N., & Matwin, S. (2013, January 1). *Inner ensembles: Using ensemble methods inside the learning algorithm*. Springer Science+Business Media. https://doi.org/10.1007/978-3-642-40994-3_3

[2] Albisua, I., Arbelaitz, O., Gurrutxaga, I., Martín, J., & Muguerza, J. (2009, November 9). *Obtaining optimal class distribution for decision trees: Comparative analysis of ctc and c4.5*. Springer Science+Business Media. https://doi.org/10.1007/978-3-642-14264-2_11

[3] Argerich, M. F. (2022). How to explain decision trees' predictions - towards data science. https://towardsdatascience.com/how-to-explain-decision-trees-predictions-7a10834fe54d

[4] Bevans, R. (2022). An introduction to t tests | definitions, formula and examples. *Scribbr*. https://www.scribbr.com/statistics/t-test/

[5] Braga, P. L., Oliveira, A. L. I., Ribeiro, G., & De Lemos Meira, S. R. (2007). Bagging predictors for estimation of software project effort. https://doi.org/10.1109/ijcnn.2007.4371196

[6] *Breast cancer wisconsin (diagnostic) data set*. (2016, September 25). https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

[7] Breiman, L. (1984). Classification and regression trees. *Biometrics*, *40*(3), 874. https://doi.org/10.2307/2530946

[8] Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/bf00058655

[9] Chawla, N. V., Hall, L. J., Bowyer, K. W., Moore, T. A., & Kegelmeyer, W. P. (2002, January 1). *Distributed pasting of small votes*. Springer Science+Business Media. https://doi.org/10.1007/3-540-45428-4_5

[10] Classification: Accuracy. (n.d.). *Google for Developers*. https://developers.google.com/machine-learning/crash-course/classification/accuracy

[11] Cohen, W. W. (1995, January 1). *Fast effective rule induction*. https://doi.org/10.1016/b978-1-55860-377-6.50023-2

[12] Domingos, P. (1999). Metacost. https://doi.org/10.1145/312129.312220

[13] Dwyer, K., & Holte, R. C. (2007, September 7). *Decision tree instability and active learning*. Springer Science+Business Media. https://doi.org/10.1007/978-3-540-74958-5_15

[14] Forrest, J., Rehder, K., Goldsmith, C. H., Cahalan, M. D., Levy, W. J., Strunin, L., Bota, W., Boucek, C. D., Cucchiara, R. F., Dhamee, S., Domino, K. B., Dudman, A. J., Hamilton, W., Kampine, J. P., Kotrly, K. J., Maltby, J. R., Mazloomdoost, M., MacKenzie, R. A., Melnick, B. M., ... Munshi, C. A. (1990). Multicenter study of general anesthesia. i. design and patient demography. *Anesthesiology*, *72*(2), 252–261. https://doi.org/10.1097/00000542-199002000-00008

[15] Frank, E., & Witten, I. (1998). Generating accurate rule sets without global optimization. *ResearchGate*. https://www.researchgate.net/publication/2779742_Generating_Accurate_Rule_Sets_Without_Global_Optimization

[16] Frost, J. (2022). Statistical hypothesis testing overview. *Statistics By Jim*. https://statisticsbyjim.com/hypothesis-testing/statistical-hypothesis-testing-overview/

[17] Fumo, J. (2018, June 21). *Types of machine learning algorithms you should know*. Retrieved June 21, 2023, from https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[18] Fürnkranz, J. (1999). Separate-and-conquer learning. *Artificial Intelligence Review*, *13*(1), 3–54. https://doi.org/10.1023/a:1006524209794

[19] Fürnkranz, J., & Widmer, G. (1994, January 1). *Incremental reduced error pruning*. https://doi.org/10.1016/b978-1-55860-335-6.50017-9

[20] Ibarguren, I., Pérez, J. M., Muguerza, J., Gurrutxaga, I., & Arbelaitz, O. (2015). Coverage-based resampling: Building robust consolidated decision trees. *Knowledge Based Systems*. https://doi.org/10.1016/j.knosys.2014.12.023

[21] insidelearningmachines. (2023, May 9). *8 key advantages and disadvantages of decision trees - inside learning machines*. https://insidelearningmachines.com/advantages_and_disadvantages_of_decision_trees

[22] *J48consolidated: An implementation of ctc for weka*. (n.d.). http://www.aldapa.eus/res/weka-ctc/galdetegia_jaso.php

[23] Jiang, W., Josse, J., & Lavielle, M. (2020). Logistic regression with missing covariates—parameter estimation, model selection and prediction within

a joint-modeling framework. *Computational Statistics Data Analysis*, *145*, 106907. https://doi.org/10.1016/j.csda.2019.106907

[24] *Kaggle: Your machine learning and data science community.* (n.d.).

[25] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, *13*, 8–17. https://doi.org/10.1016/j.csbj.2014.11.005

[26] M, V. D. L. J. G. (2023, May 31). *Optimal decision trees for separable objectives: Pushing the limits of dynamic programming.* https://arxiv.org/abs/2305.19706

[27] *Machine learning classifiers - the algorithms how they work.* (2020, December 14). https://monkeylearn.com/blog/what-is-a-classifier/

[28] Molnar, C. (2023, March 2). *5.4 decision tree | interpretable machine learning.* https://christophm.github.io/interpretable-ml-book/tree.html

[29] Patni, S. (2019). Know about the types of machine learning. part 2. - shubham patni - medium. https://medium.com/@shubhapatnim86/know-about-types-of-machine-language-part-2-19ca69071ed

[30] Pérez, J. M. (2004). *A new algorithm to build consolidated trees: Study of the error rate and steadiness.* https://www.semanticscholar.org/paper/A-new-algorithm-to-build-consolidated-trees%3A-study-P%C3%A9rez-Muguerza/6da70e966518cd452420aff84c55e217480de868

[31] Pérez, J. M., Albisua, I., Arbelaitz, O., Gurrutxaga, I., Martín, J., Muguerza, J., & Perona, I. (2010). Consolidated trees versus bagging when explanation is required. *Computing*, *89*(3-4), 113–145. https://doi.org/10.1007/s00607-010-0094-z

[32] Pérez, J. M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., & Martín, J. (2006, January 1). *Consolidated trees: An analysis of structural convergence.* Springer Science+Business Media. https://doi.org/10.1007/11677437_4

[33] Pérez, J. M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I., & Martín, J. (2007). Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, *28*(4), 414–422. https://doi.org/10.1016/j.patrec.2006.08.013

[34] *Pima indians diabetes database.* (2016, October 6). https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

[35] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/bf00116251

[36] Quinlan, J. R. (1992, October 15). *C4.5: Programs for machine learning*. https://cds.cern.ch/record/2031749

[37] Rishabh, P. (2020). 3 types of machine learning. *New Tech Dojo*. https://www.newtechdojo.com/3-types-of-machine-learning

[38] Sajee, A. (2021). Model complexity, accuracy and interpretability - towards data science. https://towardsdatascience.com/model-complexity-accuracy-and-interpretability-59888e69ab3d

[39] Salzberg, S. L. (1994). C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, *16*(3), 235–240. https://doi.org/10.1007/bf00993309

[40] Schapire, R. E. (1989). The strength of weak learnability. https://doi.org/10.1109/sfcs.1989.63451

[41] *Scipy.stats.norm — scipy v1.10.1 manual*. (n.d.). https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html

[42] , S. (2023, June 16). *Supervised vs unsupervised learning explained*. https://www.seldon.io/supervised-vs-unsupervised-learning-explained

[43] Sharma, A. (2023). 4 simple ways to split a decision tree in machine learning (updated 2023). *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2020/06/4-ways-split-decision-tree/

[44] *Uci machine learning repository*. (n.d.). https://archive.ics.uci.edu/

[45] Waitman, L. R., Fisher, D., & King, P. I. (2006). Bootstrapping rule induction to achieve rule stability and reduction. *Journal of Intelligent Information Systems*, *27*(1), 49–77. https://doi.org/10.1007/s10844-006-1626-z

[46] Wang, S., & Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. https://doi.org/10.1109/cidm.2009.4938667

[47] Weiss, G. M., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, *19*, 315–354. https://doi.org/10.1613/jair.1199

[48] *Weka 3 - data mining with open source machine learning software in java*. (n.d.). https://www.cs.waikato.ac.nz/ml/weka/

[49] Wikipedia contributors. (2023a). Iris flower data set. *Wikipedia*. https://en.wikipedia.org/wiki/Iris_flower_data_set

[50] Wikipedia contributors. (2023b). Shapiro–wilk test. *Wikipedia*. https://https://en.wikipedia.org/wiki/Shapiro-Wilk_test

[51] *Wine quality prediction | kaggle*. (n.d.). https://www.kaggle.com/competitions/wine-quality-pred

[52] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A. S. K., Liu, B., Yu, P. S., Zhou, Z., Steinbach, M., Hand, D. J., & Steinberg, D. (2007). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37. https://doi.org/10.1007/s10115-007-0114-2

[53] Zafar, M. B. (2015, July 19). *Fairness constraints: Mechanisms for fair classification*. https://arxiv.org/abs/1507.05259

VITA

Sharanya D Aavunoori was born in Telangana, India. She pursued a Bachelor's degree in Electronics and Communication Engineering at Methodist College, affiliated with Osmania University, and graduated in May 2012. Following her undergraduate studies, she worked at IBM India as an application developer from 2014 to 2020. In June 2021, she obtained a post-graduate Diploma in Data Science with a specialization in Natural Language Processing (NLP) from the International Institute of Information Technology-Bangalore (IIIT-B). Later in August 2021, she joined the Master's program at Tennessee Technological University (TTU) and pursued a Master of Science degree in Computer Science. She successfully completed her master's degree in Computer Science in August 2023