# BUILDING A RULE-BASED EXPERT SYSTEM TO ENHANCE THE HARD DISK DRIVE MANUFACTURING PROCESSES

SUPPAKRIT KIRDPONPATTARA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
DOCTOR OF ENGINEERING
IN ROBOTICS AND COMPUTATIONAL INTELLIGENCE SYSTEM
SCHOOL OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2024

KMITL-2024-EN-D-408-205

| | |
|---|---|
| **หัวข้อวิทยานิพนธ์** | การสร้างระบบผู้เชี่ยวชาญที่ใช้กฎเกณฑ์เพื่อเพิ่มประสิทธิภาพในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ |
| **นักศึกษา** | นายศุภกฤต เกิดผลภัทระ |
| **รหัสประจำตัว** | 63601230 |
| **ปริญญา** | วิศวกรรมศาสตรดุษฎีบัณฑิต |
| **สาขาวิชา** | วิศวกรรมหุ่นยนต์และระบบอัจฉริยะเชิงคำนวณ (หลักสูตรสหวิทยาการ) |
| **พ.ศ.** | 2567 |
| **อาจารย์ที่ปรึกษาวิทยานิพนธ์** | ศ.ดร.ปิติเขต สู้รักษา |
| **อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม** | รศ.ดร.วีระ บุญจริง |

## บทคัดย่อ

การผลิตฮาร์ดดิสก์ไดรฟ์เป็นกระบวนการที่ซับซ้อนซึ่งต้องการการประกอบชิ้นส่วนต่าง ๆ เข้าด้วยกัน และการทดสอบที่ครอบคลุมหลังจากการผลิต เพื่อให้แน่ใจว่าทุกชิ้นส่วนสามารถทำงานร่วมกันได้อย่างถูกต้องและมีประสิทธิภาพ การแยกไดรฟ์ที่คาดว่าจะล้มเหลวออกจากกระบวนการทดสอบทั้งหมดสามารถช่วยเพิ่มประสิทธิภาพในการผลิตและการทดสอบ ดังนั้นการศึกษานี้นำเสนอระบบผู้เชี่ยวชาญที่ใช้กฎเกณฑ์โดยใช้ข้อมูลการผลิตเพื่อแยกแยะไดรฟ์ที่คาดว่าจะล้มเหลว วิธีการนี้สามารถลดเวลาในการทดสอบและเพิ่มประสิทธิภาพให้กับเครื่องที่ใช้ทดสอบ ตัวชี้วัดความสำเร็จของวิธีการที่เสนอนี้คือความถูกต้องของโมเดลการทำนาย การศึกษานี้ใช้อัลกอริทึมต้นไม้การตัดสินใจสามแบบ - ID3, C4.5, และ CART - เพื่อหาตัวจำแนกที่มีประสิทธิภาพที่สุด และใช้เทคนิคการเลือกคุณลักษณะสี่แบบ - Information Gain, Gain Ratio, Chi-Square, และ Symmetrical Uncertainty – เพื่อระบุคุณลักษณะที่มีผลกระทบสูง ผลการทดลองพบว่าการรวม Information Gain กับอัลกอริทึม C4.5 ให้ผลลัพธ์ที่ดีที่สุดในเรื่องความถูกต้องของการทำนาย ระยะเวลาในการสร้างโมเดล และจำนวนกฎ นอกจากนี้การศึกษานี้ยังระบุโมเดลที่เหมาะสมที่สุดที่มีความน่าจะเป็นในการล้มเหลวอยู่ในช่วง 0.15 ถึง 0.70 ซึ่งจะลดเวลาทดสอบทั้งหมดสำหรับกระบวนการที่เสนอ ด้วยความมั่นใจ 95% เวลาทดสอบที่ลดลงนี้แสดงถึงการเพิ่มประสิทธิภาพอย่างมีนัยสำคัญเมื่อเทียบกับกระบวนการเดิม

| | |
|---|---|
| **Thesis** | Building a Rule-Based Expert System to Enhance the Hard Disk Drive Manufacturing Processes |
| **Student** | Mr. Suppakrit Kirdponpattara |
| **Student ID.** | 63601230 |
| **Degree** | Doctor of Engineering |
| **Program** | Robotics and Computational Intelligence Systems |
| **Year** | 2024 |
| **Thesis Advisor** | Prof. Dr. Pitikhate Sooraksa |
| **Co-Thesis Advisor** | Assoc. Prof. Dr. Veera Boonjing |


## ABSTRACT

In hard disk drive production, each unit undergoes a comprehensive testing process due to the multitude of components involved. Each component must function correctly and in harmony with others. Removing known defective drives from the full testing process can enhance the manufacturing process by freeing up test capacity. The study suggests a rule-based expert system that uses assembly process data to predict and identify faulty drives before they undergo expensive and time-consuming tests. This approach could drastically cut down testing time and boost tester capacity. The success of this proposed method hinges on the prediction model's accuracy. Given that the assembly data are categorical and imbalanced, the Decision Tree is selected as the prediction model. The study scrutinizes three Decision Tree algorithms - ID3, C4.5, and CART - to find the most effective defect classifier. Four feature selection techniques - Information Gain, Gain Ratio, Chi-Square, and Symmetrical Uncertainty - are employed to pinpoint high-impact features. The experimental findings reveal that the combination of Information Gain with the C4.5 algorithm offers the best results in terms of prediction accuracy, modeling time, and rule count. Furthermore, the study identifies the optimal model with a failure probability threshold ranging from 0.15 to 0.70, which minimizes the total test time for the proposed process. With 95% confidence, this reduced test time represents a statistically significant enhancement over the existing process.

# ACKNOWLEDGEMENT

After earning my master's degree, I began my career in Korat. The idea of pursuing a PhD was always present, but the timing was unclear. However, after nearly a decade in the workforce and with the rise of AI/ML applications in the industry, it seemed like the perfect time to start my doctoral journey. This decision was also influenced by the life plans that my wife, Nalinart Kirdponpattara, and I had, especially considering the potential busyness of starting a family. Dr. Veera Boonjing, my master's degree advisor, was the first person I consulted for advice. My wife and I decided to apply for the Robotic and AI major based on his guidance. Later, I found my second advisor, Dr. Pitikhate Sooraksa. The Covid-19 pandemic shifted the educational landscape to online learning, a significant change from my previous academic experiences. However, this change was beneficial as it eliminated the need for commuting from Korat to Bangkok. The busiest phase of my life began in 2021 with the birth of my adorable child, Karnthapaphar Kirdponpattara. Now, at the end of this journey, we have navigated through with love and support from everyone involved. However, the journey of learning never ends.

I would like to express my deepest gratitude to my advisor, Dr. Pitikhate Sooraksa and my co-advisor, Dr. Veera Boonjing, for their invaluable guidance, support, and encouragement throughout this research. Their expertise and insights have been instrumental in the development and completion of this work.

I extend my appreciation to King Mongkut's Institute of Technology Ladkrabang for providing the opportunity and facilities to conduct this research.

I also acknowledge the hard disk drive manufacturer that supplied the data for this research. The data was crucial for the analysis and evaluation of the proposed approach.

I would like to express my thanks to my father, mother, and mother-in-law for their support and encouragement.

I am grateful to my senior VP, senior director, and senior manager for their support.

Lastly, I extend my heartfelt thanks to my family and friends for their unwavering love, support, and encouragement throughout my academic journey. They have been my source of inspiration and motivation.

I dedicate all the good or benefits from this thesis to my advisors, father, mother, mother-in-law, wife, child, family, all respected professors, friends, and everyone involved.

Suppakrit Kirdponpattara

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| $\chi^2$ | Chi-Square |
| ANN | Artificial Neural Network |
| BPNN | Back Propagation Neural Network |
| C4.5 | Improved version of ID3 |
| CART | Classification and Regression Tree |
| CT | Classification Tree |
| DT | Decision Tree |
| FN | False Negative |
| FP | False Positive |
| FS | Feature Selection |
| GA | Genetic Algorithm |
| GBT | Gradient Boosting Tree |
| GR | Gain Ratio |
| HDD | Hard Disk Drive |
| HGA | Head Gimbal Assembly |
| HSA | Head Stack Assembly |
| ID3 | Iterative Dichotomiser 3 |
| IG | Information Gain |
| MBA | Motor Base Assembly |
| MLR | Multiple Linear Regression |
| NB | Naive Bayes |
| NN | Neural Network |
| ORF | Online Random Forest |
| PCBA | Printed Circuit Board Assembly |
| RAT | Rank-sum test Attribute |
| RF | Random Forest |
| RNN | Recurrent Neural Network |
| SMART | Self-Monitoring, Analysis, and Reporting Technology |
| SU | Symmetrical Uncertainty |
| SVM | Support Vector Machine |
| TN | True Negative |
| TP | True Positive |
| VCM | Voice Coil Motor |

# CHAPTER 1
# INTRODUCTION

## 1.1 BACKGROUND AND MOTIVATION

The Hard Disk Drive (HDD) industry is currently witnessing a steady growth, driven by several factors such as the increasing demand for storage space, technological advancements, the rise in cloud usage, growth in electronic product manufacturing, and increased usage in consumer electronics. The manufacturing of HDDs entails the assembly of various components using a range of machines. After assembly, each HDD is subjected to a crucial testing phase to detect any defects, ensuring that only fully operational HDDs are delivered to customers. This process is depicted in Figure 1.1. The testing of each HDD unit is an expensive and lengthy procedure due to the multitude of components that make up each unit. Each component must be verified to function correctly and in harmony with the others [1]-[4]. As a result, testing a high-capacity product, such as a 20 TB HDD, takes at least a month, regardless of whether it is defective. However, a known defective HDD can be excluded from the full testing process, thereby increasing the testing capacity and enhancing the manufacturing process. This study proposes a rule-based expert system to identify potentially defective HDDs, as illustrated in Figure 1.2. A predicted defect undergoes a shorter testing process, where the operations are chosen from a comprehensive list of all test operations for quick defect identification. This method not only expedites defect detection but also reduces the use of testing capacity, ultimately enhancing the manufacturing process. The effectiveness of this approach hinges on the accuracy of the prediction model, which is solely based on information about the components and assembly machines used, potentially supplied by various vendors.

This study faces two main challenges in building a prediction model: (1) all input features to the model are categorical, and (2) there is a large number of input features. A Decision Tree (DT) is a comprehensible binary classifier and is commonly used for categorical data [5]. It often performs well on imbalanced data due to its hierarchical structure, which allows it to learn signals from both classes. Therefore, this study selects the DT as the prediction model. This study assesses the ID3, C4.5, and CART algorithms [6] to pinpoint the most precise one. The profusion of input features

could lead to DTs that learn from all input features to exhibit low predictive accuracy due to input features unrelated to the target class. To address this challenge, feature selection (FS) [7] is introduced by examining all combinations of four FS techniques (Information Gain, Gain Ratio, Chi-Square, and Symmetrical Uncertainty) with the three DTs mentioned earlier. The best combination is identified based on prediction accuracy, modeling time, top N features, and the number of rules. The study further delves into this optimal combination to discover its ideal parameters that reduce the total test time of the proposed process.

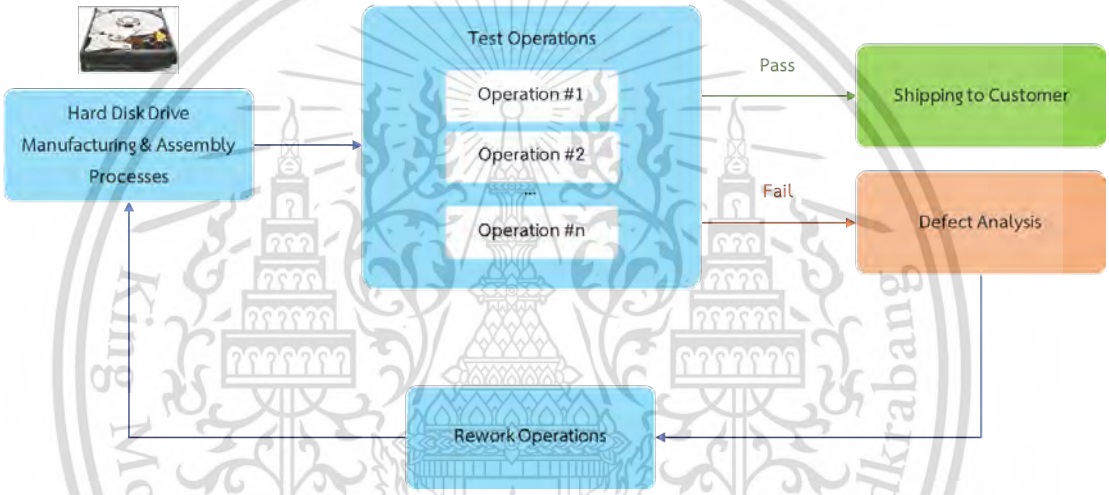


**Figure 1.1** Current production process

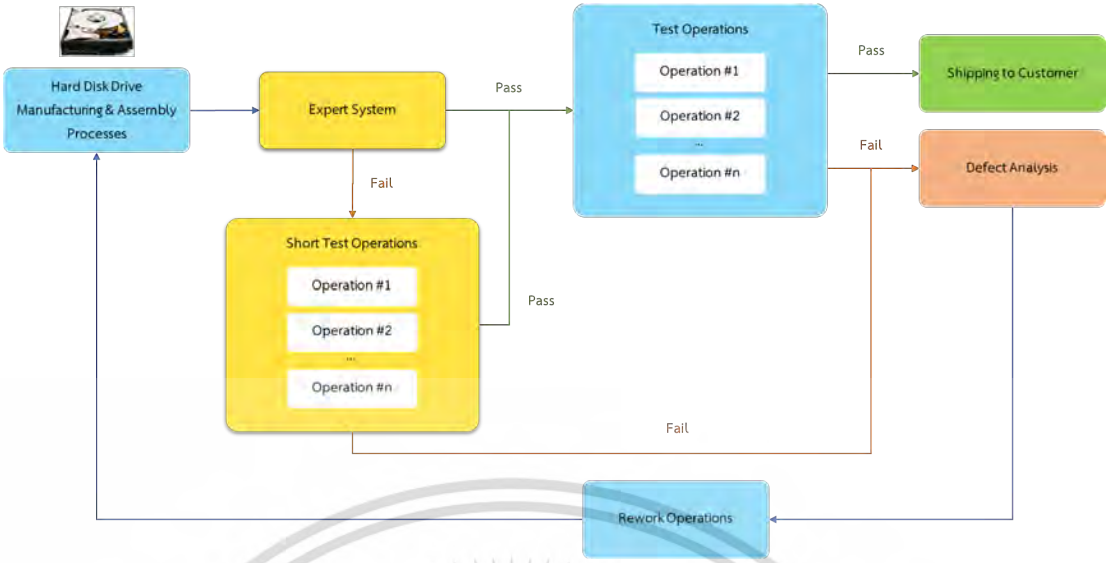


**Figure 1.2** Proposed production process

## 1.2 PROBLEM STATEMENT AND OBJECTIVES

The main problem addressed in this research is how to reduce the testing time and resource consumption of defective HDDs in the manufacturing process. The objective is to develop a rule-based expert system that can predict the defect probability of each HDD based on the assembly data, and assign it to a tailored testing process that can quickly detect the defect. The expert system uses FS techniques and DT algorithms to generate minimal and interpretable classification rules from the assembly data. The performance of the expert system is evaluated in terms of accuracy, modeling time, rule generation, and test time reduction.

## 1.3 SCOPE AND LIMITATIONS

The scope of this research is limited to a single timeframe of production and a single HDD product. The data used for the analysis is obtained from a real-world HDD manufacturer, and it contains 43 categorical features related to the components and machines used in the assembly process, and two classes indicating whether the HDD is defective or not. The data is complete and does not contain any missing values. The FS techniques used in this research are Information Gain (IG), Gain Ratio (GR), Chi-Square ($\chi^2$), and Symmetrical

Uncertainty (SU). The DT algorithms used are ID3, C4.5, and CART. The expert system is implemented in R software and uses five-fold cross-validation for model evaluation.

The limitations of this research are as follows:

1) The expert system may not generalize well to other timeframes or HDD products, as the data and the rules may change over time or across products.
2) The expert system relies on the quality and availability of the assembly data, which may vary depending on the vendors and machines involved in the production process.
3) The expert system may not account for all factors influencing the defect probability, such as environmental conditions, human errors, or hardware failures, which are not captured by the assembly data.
4) The expert system may have ethical implications, such as data privacy and security, and impact on workers, which need to be considered and addressed.

## 1.4 CONTRIBUTIONS AND SIGNIFICANCE

The main contributions and significance of this research are as follows:

1) It proposes a novel approach to enhance the manufacturing process of HDDs by using a rule-based expert system to identify defective HDDs before extensive testing.
2) It applies FS techniques and DT algorithms to generate accurate, efficient, and interpretable rules for defect prediction from assembly data.
3) It optimizes the testing process by assigning HDDs to a tailored testing process based on their defect probability, resulting in test time and resource reduction.
4) It provides a practical and robust solution for HDD manufacturers, as well as a methodological framework for other similar domains.

## 1.5 THESIS ORGANIZATION

The rest of the thesis is organized as follows:

1) Chapter 2 provides a review of the relevant literature, including HDD production, FS techniques, DT algorithms, and related works.

2) Chapter 3 describes the materials and methods used in this research, including the data, experiment preparation, the FS techniques, the DT algorithms, the expert system, and the evaluation metrics.
3) Chapter 4 presents the results and discussion of the experiments, including the accuracy, modeling time, top N features, rule generation, and test time reduction of the expert system.
4) Chapter 5 concludes the thesis and suggests some directions for future work.

# CHAPTER 2

# LITERATURE REVIEW

This chapter provides the background and related works pertaining to the present study. The initial part outlines the HDD product and its example dataset. The following parts present illustrative on FS techniques and DT algorithms employed in this study. Finally, the last part of this section provides an overview of the relevant works in this area.

## 2.1 HARD DISK DRIVE

A Hard Disk Drive (HDD) is a type of storage device that retains digital data even when powered off. It uses magnetic storage technology to store and retrieve data on robust material disks [8]. The manufacturing process of an HDD involves the assembly of various hardware components, each sourced from different vendors and assembled by specific production machines.

As depicted in Figure 2.1, an HDD consists of six primary components [9]:

1) Base Deck/Motor Base Assembly (MBA): This is the aluminum casing that protects all the internal hardware components of the HDD from external elements [10].
2) Spindle Motor: This component manages and controls the rotational speed of the disks [11].
3) Disk Platters: These are the components where data is stored. They consist of platinum group metals deposited on a substrate [12].
4) Head Stack Assembly (HSA): Composed of the Actuator Arm and Head Gimbal Assembly (HGA), which includes the assembly of the slider and the suspension. The HSA synchronizes the movement of the heads to the correct reading/writing position, in line with the motor's rotational speed [13].
5) Voice Coil Motor (VCM): This is an electromechanical linear motor that positions the read/write heads by moving the HSA [14].
6) Printed Circuit Board Assembly (PCBA): This is a controller board that manages the HDD and serves as an interface between the HDD and the computer [15].

The assembly of all these components follows a specific process flow for each production machine. Therefore, the manufacturing of each HDD component, sourced from various vendors and produced by different machines, will determine whether the HDD is classified as defective or passing.


**Figure 2.1** Hard disk drive components

The construction of HDDs involves the combination of various components using a range of machinery. After the assembly process, each HDD undergoes a comprehensive inspection to confirm its flawless operation before it is dispatched. This inspection phase is particularly extensive and resource-intensive, especially for high-capacity HDDs like 20 TB models, which may necessitate up to a month for testing, regardless of whether they are defective.

The current production process for HDDs is represented in Figure 2.2. It outlines the sequence of operations required to assemble an HDD, with a special emphasis on the crucial testing phase that verifies the proper functioning of each unit and its freedom from defects. This flow chart is a fundamental component of the study, as it demonstrates the intricacy of the HDD manufacturing process and underscores the significance of efficient testing for quality control.

To enhance this procedure, the document presents a rule-based expert system devised to identify potential defects early in the assembly data. This system utilizes predictive models that facilitate a more efficient and expedited testing of HDDs identified as potentially defective, thereby saving time and resources, as shown in Figure 2.3. The system's efficiency is dependent on the precision of its prediction model,

which leverages FS techniques and DT algorithms to generate clear and interpretable classification rules. These rules consider the various sources of the components and machines involved in the HDD assembly process.

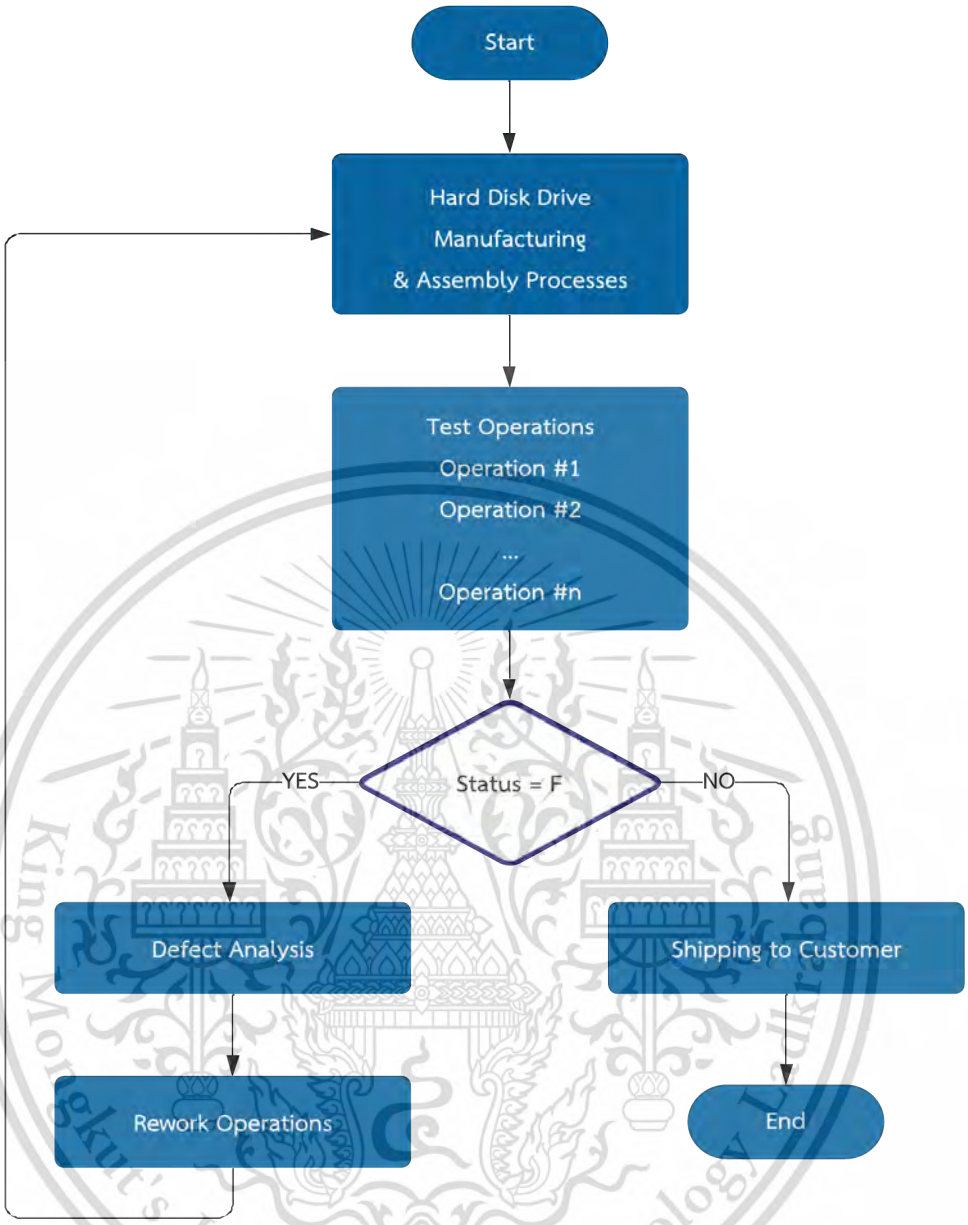


**Figure 2.2** Current production process flow chart

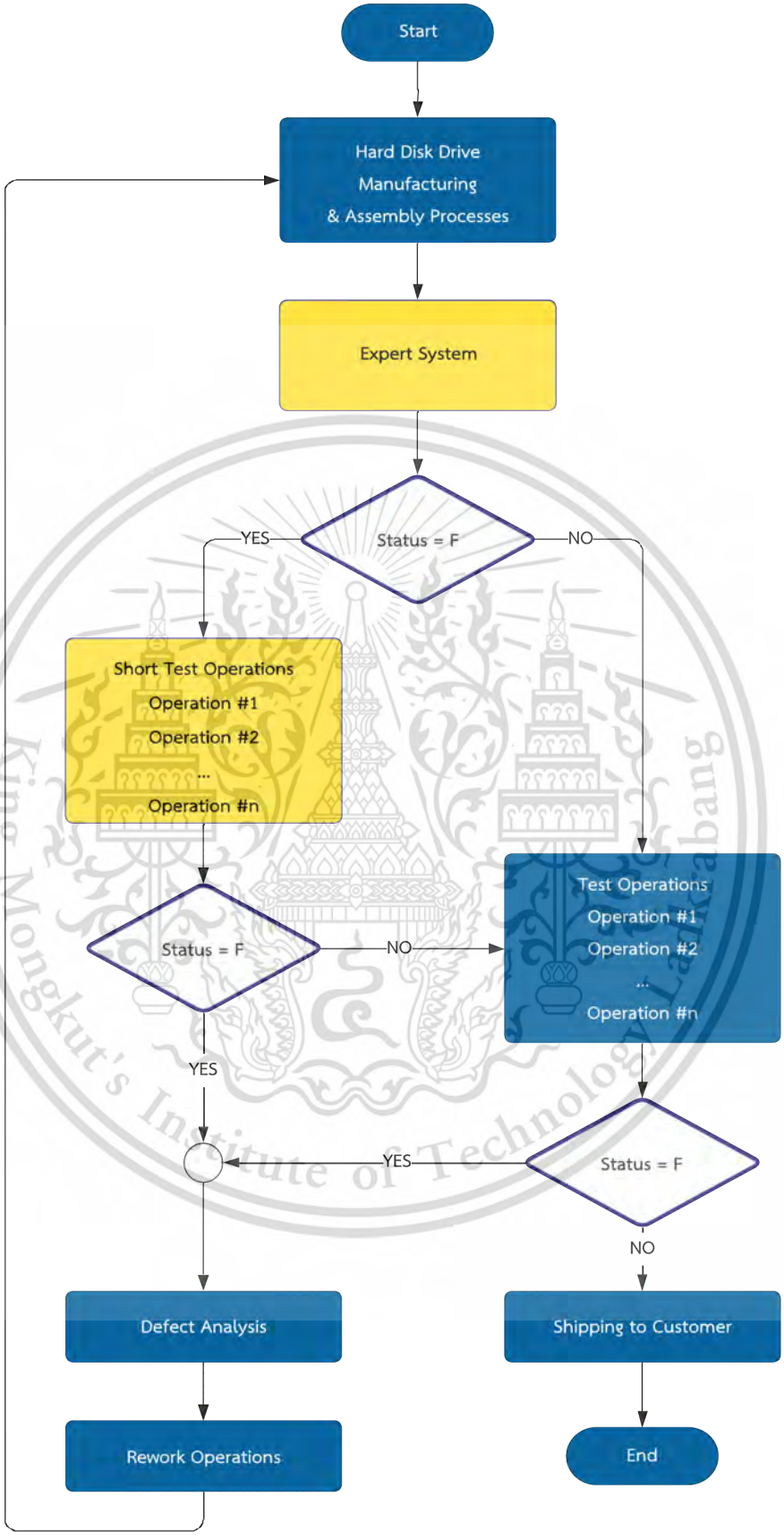


**Figure 2.3** Proposed production process flow chart

Table 2.1 represents an example dataset utilized in this study for the purpose of demonstrating the application of FS techniques and DT algorithms used in this study. The dataset includes three feature types, namely, components vendors, production machines, and the target class.

**Table 2.1 Example HDD Manufacturing Dataset**

| Component_#1 | Component_#2 | Machine_#1 | Machine_#2 | Class |
|---|---|---|---|---|
| C1_A | C2_A | M1_A | M2_A | F |
| C1_A | C2_A | M1_A | M2_B | F |
| C1_B | C2_A | M1_A | M2_A | P |
| C1_C | C2_B | M1_A | M2_A | P |
| C1_C | C2_C | M1_B | M2_A | P |
| C1_C | C2_C | M1_B | M2_B | F |
| C1_B | C2_C | M1_B | M2_B | P |
| C1_A | C2_B | M1_A | M2_A | F |
| C1_A | C2_C | M1_B | M2_A | P |
| C1_C | C2_B | M1_B | M2_A | P |
| C1_A | C2_B | M1_B | M2_B | P |
| C1_B | C2_B | M1_A | M2_B | P |
| C1_B | C2_A | M1_B | M2_A | P |
| C1_C | C2_B | M1_A | M2_B | F |
| C1_A | C2_A | M1_A | M2_A | F |

## 2.2 FEATURE SELECTION TECHNIQUES

Feature selection (FS) is a process of selecting a subset of relevant features from the original feature set to improve the performance of the prediction model. FS can reduce the dimensionality of the data, enhance the accuracy and interpretability of the model, and decrease the computation time and complexity. FS techniques can be categorized into three types: filter, wrapper, and embedded methods. Filter methods rank the features based on some statistical measures and select the top-ranked features. Wrapper methods use a search strategy to find the optimal subset of features that maximizes the performance of a specific prediction model. Embedded methods

integrate the FS process within the model construction, such as DTs. In this study, four FS techniques, namely IG, GR, $\chi^2$, and SU, are utilized to rank features based on their importance. These techniques leverage various criteria, including entropy, split information, impurity, and correlation, to determine the association degree between the features and the class.

### 2.2.1 Information Gain (IG)

This technique calculates the difference in Entropy between initial data and the data after separation into classes as:

$$IG(T) = Entropy(T) - [P(T|f = v_1) \times Entropy(T|f = v_1) + P(T|f = v_2) \times Entropy(T|f = v_2) + \dots] \quad (2.1)$$

$$Entropy(T) = -[P(c_1)log_2P(c_1) + P(c_2)log_2P(c_2) + \dots] \quad (2.2)$$

when $T$ is the dataset, $f$ is the feature, $P(T|f = v)$ is the probability of each value in feature $f$, and $P(c)$ is the probability of each class of dataset $T$.

The example dataset consists of 14 instances with 2 classes (9 instances of P and 5 instances of F). First, we calculate $Entropy(T)$ by

$$Entropy(T) = -P(Y)log_2P(P) - P(N)log_2P(F)$$
$$= -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14}$$
$$= 0.940$$

, then calculate IG for each column by starting at Machine_#2 feature.

$$IG(T, Machine\_\#2) = Entropy(T) - [P(T|Machine\_\#2 = M2\_A) \times Entropy(T|Machine\_\#2 = M2\_A) + P(T|Machine\_\#2 = M2\_B) \times Entropy(T|Machine\_\#2 = M2\_B)]$$
$$= 0.940 - \left[\frac{8}{14}\left(-\frac{6}{8}log_2\frac{6}{8} - \frac{2}{8}log_2\frac{2}{8}\right) + \frac{6}{14}\left(-\frac{3}{6}log_2\frac{3}{6} - \frac{3}{6}log_2\frac{3}{6}\right)\right]$$
$$= 0.048$$

$IG(T, Component\_\#1) = 0.246$
$IG(T, Component\_\#2) = 0.029$
$IG(T, Machine\_\#1) = 0.151$

The feature with the highest IG value is considered the best feature for separating the classes. Based on this criterion, the best feature is Component_#1.

### 2.2.2 Gain Ratio (GR)

Gain Ratio (GR) is defined as the ratio of IG and Split Information and can be represented as Eq. (2.3).

$$GR(T,f) = \frac{IG(T,f)}{SplitInfo(T,f)} \quad \textbf{(2.3)}$$

$$SplitInfo(T,f) = -[P(T|f=v_1)log_2P(T|f=v_1) + P(T|f=v_2)log_2P(T|f=v_2) + \ ...] \quad \textbf{(2.4)}$$

When utilizing the example dataset, the calculation process commences by computing the SplitInfo for each feature, which is achieved through the following steps.

$$SplitInfo(T, Machine\_\#2) = -[P(T|Machine\_\#2 = M2\_A)log_2P(T|Machine\_\#2 = M2\_A) + P(T|Machine\_\#2 = M2\_B)log_2P(T|Machine\_\#2 = M2\_B)]$$

$$= -\left[\frac{8}{14}log_2\frac{8}{14} + \frac{6}{14}log_2\frac{6}{14}\right]$$

$$= 0.985$$

so

$$GR(T, Machine\_\#2) = \frac{IG(T, Machine\_\#2)}{SplitInfo(T, Machine\_\#2)}$$

$$= \frac{0.048}{0.985}$$

$$= 0.049$$

$$GR(T, Component\_\#1) = 0.15,$$
$$GR(T, Component\_\#2) = 0.019$$
$$GR(T, Machine\_\#1) = 0.151$$

The feature with the highest GR value is deemed the best feature for segregating the classes. Based on this evaluation, Component_#1 is identified as the best feature.

### 2.2.3 Chi-Square ($\chi^2$)

The Chi-Square statistical method is utilized to measure dependence between each feature and the classes. If a feature exhibits a high $\chi^2$ value, it signifies a high level of dependence on the output classes. The calculation formula is presented in Eq. (2.5) and (2.6).

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \quad \textbf{(2.5)}$$

$$E_i = \frac{R_T \times C_T}{N} \quad \textbf{(2.6)}$$

when $O_i$ is an observed value, $E_i$ is an expected value, $R_T$ is total row instances, $C_T$ is total column instances, and $N$ is total instances.

In the example dataset, the calculation of $\chi^2$ for Machine_#2 and Class begins with constructing a contingency table, as illustrated in Table 2.2.

**Table 2.2 Contingency table for Machine_#2 and Class**

| Machine_#2 | Class | | Total |
|---|---|---|---|
| | F | P | |
| M2_A | 2 | 6 | 8 |
| M2_B | 3 | 3 | 6 |
| Total | 5 | 9 | 14 |

Then calculate expected values by Eq. (2.6).

$E_{M2\_A,F} = \frac{5 \times 8}{14} = 2.857$

$E_{M2\_A,P} = 5.143$, $E_{M2\_B,F} = 2.143$, and $E_{M2\_B,P} = 3.857$

so

$$\chi^2(Machine\_\#2) = \frac{(2-2.857)^2}{2.857} + \frac{(6-5.143)^2}{5.143} + \frac{(3-2.143)^2}{2.143} + \frac{(3-3.857)^2}{3.857} = 0.933$$

$\chi^2(Component\_\#1) = 3.547$
$\chi^2(Component\_\#2) = 0.570$
$\chi^2(Machine\_\#1) = 2.8$

The feature demonstrating the highest $\chi^2$ value is considered the most dependent on the class. Based on this assessment, Component_#1 is the best feature.

### 2.2.4 Symmetrical Uncertainty (SU)

This refers to the correlation measurement between the features and the classes. The formula is defined as Eq. (2.7).

$$SU(T,f) = \frac{2 \times IG(T|f)}{Entropy(T) + Entropy(f)} \quad \textbf{(2.7)}$$

when $T$ is the dataset class and $f$ is the feature.

From the example dataset, SU of Machine_#2 and Class starts with calculating Entropy of each feature and class by

$$\begin{aligned} Entropy(Machine\_\#2) &= -P(M2\_A)log_2P(M2\_A) - P(M2\_B)log_2P(M2\_B) \\ &= -\frac{8}{14}log_2\frac{8}{14} - \frac{6}{14}log_2\frac{6}{14} \\ &= 0.985 \end{aligned}$$

$$Entropy(Component\_\#1) = 1.577$$
$$Entropy(Component\_\#2) = 1.557$$
$$Entropy(Machine\_\#1) = 1$$

From Eq. (2.1) and Eq. (2.2), so

$$\begin{aligned} SU(T, Machine\_\#2) &= \frac{2 \times IG(T|Machine\_\#2)}{Entropy(T) + Entropy(Machine\_\#2)} \\ &= \frac{2 \times 0.048}{0.940 + 0.985} \\ &= 0.050 \end{aligned}$$

$$SU(T, Component\_\#1) = 0.195$$
$$SU(T, Component\_\#2) = 0.310$$
$$SU(T, Machine\_\#1) = 0.293$$

The feature demonstrating the highest SU value is considered the most correlated to the classes. Based on this evaluation, Component_#2 is deemed the best feature.

Table 2.3 shows rank of feature importance pertains to the process of assessing the most significant features within a given dataset based on predetermined evaluation metrics. Component_#1 is the most important for IG, GR, and $\chi^2$, but Compo-nent_#2 is the most important for SU.

**Table 2.3 Rank of feature importance**

| Feature Selection Technique | Rank of Feature | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| IG | Component_#1 | Machine_#1 | Machine_#2 | Component_#2 |
| GR | Component_#1 | Machine_#1 | Machine_#2 | Component_#2 |
| $\chi^2$ | Component_#1 | Machine_#1 | Machine_#2 | Component_#2 |
| SU | Component_#2 | Machine_#1 | Component_#1 | Machine_#2 |

## 2.3 DECISION TREE ALGORITHMS

A decision tree (DT) is a graphical representation of either a classification or regression problem. Each node in this tree corresponds to a feature, each branch represents a decision rule, and each leaf node represents an outcome. The DT has the capability to recursively partition data into smaller subsets based on feature values until a stopping criterion is met. One of the advantages of using a DT is its ease of interpretation and explanation. It also has the ability to handle both categorical and numerical data. However, DTs can be prone to overfitting, sensitive to noise and outliers, and unstable to small changes in the data. This section provides a description of three popular DT algorithms, namely ID3, C4.5, and CART.

### 2.3.1 Iterative Dichotomiser 3 (ID3)

ID3 is considered one of the most well-known DT algorithms and was first introduced in 1986 [16]. It utilizes IG to select features for modeling and is suitable for analyzing categorical data. The following steps are taken to calculate the ID3 for the example dataset.

Step 1: Begin by selecting the root node, which involves considering the feature with the highest IG. As per section 2.3.1, Component_#1 is identified as the root node.

Step 2: Component_#1 can assume three values, namely C1_C, C1_B, and C1_A. Calculate IG for each feature for all values of Component_#1 to determine the second level of the DT. Begin with Component_#1 = C1_C, as presented in Table 2.4.

**Table 2.4 Simple HDD Manufacturing dataset where Component_#1 = C1_C**

| Component_#1 | Component_#2 | Machine_#1 | Machine_#2 | Class |
|---|---|---|---|---|
| C1_C | C2_B | M1_A | M2_A | P |
| C1_C | C2_C | M1_B | M2_A | P |
| C1_C | C2_C | M1_B | M2_B | F |
| C1_C | C2_B | M1_B | M2_A | P |
| C1_C | C2_B | M1_A | M2_B | F |

$$
\begin{aligned}
&IG(Component\_\#1 = C1\_C, Machine\_\#2) \\
&\quad = Entropy(Component\_\#1 = C1\_C) \\
&\quad - [P(Component\_\#1 = C1\_C | Machine\_\#2 = M2\_A) \\
&\quad \times Entropy(Component\_\#1 = C1\_C | Machine\_\#2 = M2\_A) \\
&\quad + P(Component\_\#1 = C1\_C | Machine\_\#2 = M2\_B) \\
&\quad \times Entropy(Component\_\#1 = C1\_C | Machine\_\#2 = M2\_B)] \\
&\quad = 0.971 - \left[\frac{3}{5}\left(-\frac{3}{3}log_2\frac{3}{3}\right) + \frac{2}{5}\left(-\frac{2}{2}log_2\frac{2}{2}\right)\right] \\
&= 0.971
\end{aligned}
$$

$$IG(Component\_\#1 = C1\_C, Component\_\#2) = 0.020$$
$$IG(Component\_\#1 = C1\_C, Machine\_\#1) = 0.020$$

Machine_#2 produces the highest IG, so Machine_#2 is the 2nd level when Component_#1 = C1_C. Then continue to Component_#1 = C1_A.

$$IG(Component\_\#1 = C1\_A, Machine\_\#2) = 0.019$$
$$IG(Component\_\#1 = C1\_A, Component\_\#2) = 0.570$$
$$IG(Component\_\#1 = C1\_A, Machine\_\#1) = 0.970$$

Machine_#1 produces the highest of IG, so Machine_#1 is the 2nd level when

Component_#1 = C1_A. Then continue to Component_#1 = C1_B.

$$IG(Component\_\#1 = C1\_B, Machine\_\#2) = 0$$
$$IG(Component\_\#1 = C1\_B, Component\_\#2) = 0$$
$$IG(Component\_\#1 = C1\_B, Machine\_\#1) = 0$$

All features produce IG = 0, so this branch is a leaf node.

Step 3: Repeat Step 2 for the next levels until all features produce IG = 0. Based on the calculation steps outlined earlier, the DT resulting from ID3 can be visualized in Figure 2.4.

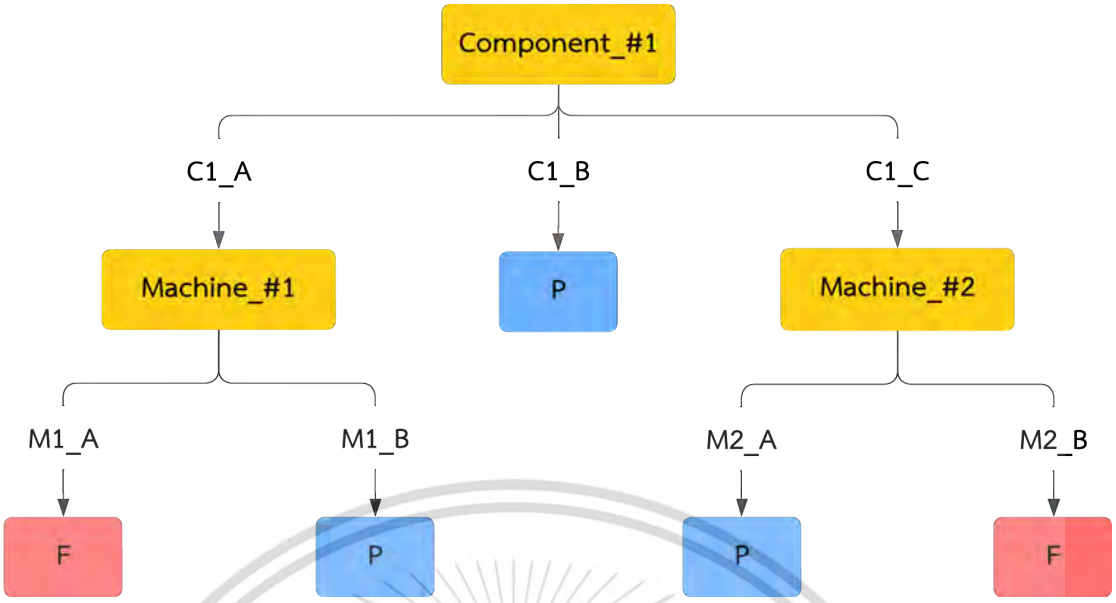


**Figure 2.4** Decision Tree from ID3, C4.5, and CART algorithms

### 2.3.2 C4.5

The C4.5 algorithm, published in 1993 [17], was an extension of ID3 that could handle both categorical and numerical data. C4.5 uses GR to select features for modeling. The following steps are taken to calculate the C4.5 for the example dataset:

Step 1: Begin by selecting the root node, which involves considering the feature with the highest GR value. As per section 2.3.2, Component_#1 is identified as the root node.

Step 2: Component_#1 can assume three values, namely C1_C, C1_B, and C1_A. Calculate the GR for each feature for all values of Component_#1 to determine the second level of DT. Begin with Component_#1 = C1_A, as presented in Table 2.5.

**Table 2.5 Simple HDD Manufacturing dataset where Component_#1 = C1_A**

| Component_#1 | Component_#2 | Machine_#1 | Machine_#2 | Class |
|---|---|---|---|---|
| C1_A | C2_A | M1_A | M2_A | F |
| C1_A | C2_A | M1_A | M2_B | F |
| C1_A | C2_B | M1_A | M2_A | F |
| C1_A | C2_C | M1_B | M2_A | P |
| C1_A | C2_B | M1_B | M2_B | P |

$$
\begin{aligned}
&IG(Component\_\#1 = C1\_A, Machine\_\#1)\\
&\quad = Entropy(Component\_\#1 = C1\_A)\\
&\quad - [P(Component\_\#1 = C1\_A | Machine\_\#1 = M1\_A)\\
&\quad \times Entropy(Component\_\#1 = C1\_A | Machine\_\#1 = M1\_A)\\
&\quad + P(Component\_\#1 = C1\_A | Machine\_\#1 = M1\_B)\\
&\quad \times Entropy(Component\_\#1 = C1\_A | Machine\_\#1 = M1\_B)]\\
&\quad = 0.971 - \left[\frac{3}{5}\left(-\frac{3}{3}log_2\frac{3}{3}\right) + \frac{2}{5}\left(-\frac{2}{2}log_2\frac{2}{2}\right)\right]\\
&= 0.971
\end{aligned}
$$

$$
\begin{aligned}
&SplitInfo(Component\_\#1 = C1\_A, Machine\_\#1)\\
&\quad = -[P(Component\_\#1 = C1\_A | Machine\_\#1\\
&\quad = M1\_A) log_2 P(Component\_\#1 = C1\_A | Machine\_\#1 = M1\_A)\\
&\quad + P(Component\_\#1 = C1\_A | Machine\_\#1\\
&\quad = M1\_B) log_2 P(Component\_\#1 = C1\_A | Machine\_\#1 = M1\_B)]\\
&\quad = -\left[\frac{3}{5}log_2\frac{3}{5} + \frac{2}{5}log_2\frac{2}{5}\right]\\
&\quad = 0.971
\end{aligned}
$$

so

$$
\begin{aligned}
&GR(Component\_\#1 = C1\_A, Machine\_\#1)\\
&\quad = \frac{IG(Component\_\#1 = C1\_A, Machine\_\#1)}{SplitInfo(Component\_\#1 = C1\_A, Machine\_\#1)}\\
&\quad = 1
\end{aligned}
$$

$$GR(Component\_\#1 = C1\_A, Component\_\#2) = 0.507$$
$$GR(Component\_\#1 = C1\_A, Machine\_\#2) = 0.037$$

Machine_#1 produces the highest of GR, so Machine_#1 is the 2nd level when Component_#1 = C1_A. Then continue to Component_#1 = C1_C.

$$GR(Component\_\#1 = C1\_C, Machine\_\#1) = -0.112$$
$$GR(Component\_\#1 = C1\_C, Component\_\#2) = -0.112$$
$$GR(Component\_\#1 = C1\_C, Machine\_\#2) = 1$$

Machine_#2 produces the highest of GR, so Machine_#2 is the 2nd level when Component_#1 = C1_C. Then continue to Component_#1 = C1_B.

$$GR(Component\_\#1 = C1\_B, Machine\_\#2) = 0$$
$$GR(Component\_\#1 = C1\_B, Component\_\#2) = 0$$
$$GR(Component\_\#1 = C1\_B, Machine\_\#1) = 0$$

All features produce GR = 0, so this branch is a leaf node.

Step 3: Repeat Step 2 for subsequent levels until all features produce a GR = 0. The resulting DT generated from C4.5 algorithm is shown in Figure 2.4.

### 2.3.3 Classification and Regression Tree (CART)

CART, which was published in 1984 [18], uses Gini Index to measure the importance of the features for modeling. The formula is represented by Eq. (2.8), and CART can support both categorical and numerical data.

$$Gini = 1 - \sum_{i} P(c_i)^2 \tag{2.8}$$

when $P(c_i)$ is the probability of each class of the example dataset.

From the example dataset, the steps for calculation are shown below.

Step 1: Begin with selecting the root node by considering the lowest Gini feature. From the below calculations, the root node is Component_#1.

$$Gini(Component\_\#1 = C1\_B) = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2 = 0$$
$$Gini(Component\_\#1 = C1\_C) = 0.48$$
$$Gini(Component\_\#1 = C1\_A) = 0.48$$
$$Gini(Component\_\#1) = \frac{4}{14} \times 0 + \frac{5}{14} \times 0.48 + \frac{5}{14} \times 0.48 = 0.342$$
$$Gini(Component\_\#2) = 0.439$$
$$Gini(Machine\_\#1) = 0.367$$
$$Gini(Machine\_\#2) = 0.428$$

Step 2: There are three values for Component_#1, i.e., C1_C, C1_B, and C1_A. Calculate Gini for each feature for each value of Component_#1 to be the 2nd level of DT. Start with Component_#1 = C1_B, which is shown in Table 2.6.

**Table 2.6 Simple HDD Manufacturing dataset where Component_#1 = C1_B**

| Component_#1 | Component_#2 | Machine_#1 | Machine_#2 | Class |
|---|---|---|---|---|
| C1_B | C2_A | M1_A | M2_A | P |
| C1_B | C2_C | M1_B | M2_B | P |
| C1_B | C2_B | M1_A | M2_B | P |
| C1_B | C2_A | M1_B | M2_A | P |

$$Gini(Component\_\#1 = C1\_B, Machine\_\#2 = M2\_A) = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$
$$Gini(Component\_\#1 = C1\_B, Machine\_\#2 = M2\_B) = 0$$
$$Gini(Component\_\#1 = C1\_B, Machine\_\#2) = 0$$
$$Gini(Component\_\#1 = C1\_B, Component\_\#2) = 0$$
$$Gini(Component\_\#1 = C1\_B, Machine\_\#1) = 0$$

All features produce Gini = 0, so this branch is a leaf node. Then continue to Component_#1 = C1_A.

$$Gini(Component\_\#1 = C1\_A, Machine\_\#2) = 0.466$$
$$Gini(Component\_\#1 = C1\_A, Component\_\#2) = 0.2$$
$$Gini(Component\_\#1 = C1\_A, Machine\_\#1) = 0$$

Machine_#1 produces the lowest of Gini, so Machine_#1 is the 2nd level when Component_#1 = C1_A. Then continue to Component_#1 = C1_C

.

$$Gini(Component\_\#1 = C1\_C, Machine\_\#2) = 0$$
$$Gini(Component\_\#1 = C1\_C, Component\_\#2) = 0.466$$
$$Gini(Component\_\#1 = C1\_C, Machine\_\#1) = 0.466$$

Machine_#2 produces the lowest of Gini, so Machine_#2 is the 2nd level when Component_#1 = C1_C.

Step 3: Repeat Step 2 for the next levels until all features produce Gini = 0. So, DT from CART can be built as shown in Figure 2.4.

For real-world datasets, the models generated by each algorithm may differ, however, for the given example dataset, all three DT algorithms yield the same final model. The summary of these algorithms is presented in Table 2.7.

**Table 2.7 Summary of 3 decision tree algorithms**

| Algorithm | Supported Feature | Formula | Missing Values |
|---|---|---|---|
| ID3 | Categorical | Information Gain | Not support |
| C4.5 | Categorical & Continuous | Gain Ratio | Support |
| CART | Categorical & Continuous | Gini Index | Support |

## 2.4 RELATED WORKS

Fault prediction in the HDD industry is bifurcated into two main categories: hard drive failure and manufacturing defects. The former leverages operating data from hard drives, specifically SMART (Self-Monitoring, Analysis, and Reporting Technology), to predict HDD status. Prominent data sources for this research encompass Backblaze and Baidu, as encapsulated in Table 2.8 [19]-[23].

The latter approach harnesses manufacturing data to prognosticate defects. Over the past decade, a handful of machine learning algorithms have been implemented to augment HDD manufacturing. These methodologies aim at (1) augmenting the yield of the hard drive manufacturing process [14] and (2) refining the accuracy of HDD yield forecasts [25]. A specific study endeavored to enhance manufacturing yields by employing a DT method to identify parameters that could be manipulated to minimize HDD defects. These parameters were categorized into three groups: uncontrollable, controllable, and dependent. The C4.5 algorithm was deployed to construct the most accurate model. The rules derived from this model were subsequently applied to adjust the controllable parameters, leading to an increase in pass rates. However, since these tests were solely conducted in a controlled environment, their performance in actual production settings remains uncertain. Another research initiative aimed at refining HDD yield predictions employed various machine learning and FS methods. Seven FS techniques were utilized, including DTs (C5 and CART), Support Vector Machine (SVM), Stepwise Regression, Genetic Algorithm (GA), $\chi^2$, and IG, to identify the top ten features for model creation. The modeling was executed using Multiple Linear Regression (MLR) and Artificial Neural Networks (ANN). The findings indicated that the GA and MLR combination yielded the most optimal prediction results, although GA necessitated the most computational time.

**Table 2.8 Studies on hard drive failure**

| Previous Related Research | Data Source | Machine Learning Algorithm | Feature Selection Technique | Number of SMART Attributes | Failure Detection Rate |
|---|---|---|---|---|---|
| Li (2014) [19] | Baidu W and Q | DT (CART) and BPNN | RAT, rank-sum test, and z-scores | 12-19 features | 95% |
| Xu (2016) [20] | Baidu W, S, and M | RNN, CT, and Binary NN | RAT, rank-sum test, and z-scores | 10 features | 97% |
| Aussel (2017) [21] | Backblaze (2014) | SVM, RF and GBT | RAT, rank-sum test, z-scores, and quantile function | 9 features | 95% |
| Xiao (2018) [22] | Backblaze (2013-2014) | ORF | Rank-sum test | 19 features | 98% (Dataset: STA) 85% (Dataset: STB) |
| Wasim (2020) [23] | Backblaze (Q1-Q3 2015) | NB and SVM | Genetic Algorithm | 42 features | 98.4% (ML: NB) 40% (ML: SVM) |

This thesis introduces an expert system that utilizes assembly data to pinpoint defective hard drives prior to extensive testing. The system's accuracy in predicting failures is pivotal.

The methodology presented surpasses previous research by potentially diminishing test time and resource consumption. It enhances HDD manufacturing by augmenting testing capabilities and refining the defect detection process. The application of FS techniques and DT algorithms yields minimal and interpretable classification rules.

Nonetheless, the approach has its constraints that warrant further investigation. The model hinges on data from a singular production timeframe, possibly not reflective of other periods or HDD products. Its efficacy is contingent on the assembly data's quality and uniformity, which may fluctuate across different vendors and machines. Moreover, the model might overlook variables influencing defect probability, such as environmental conditions, human errors, or hardware failures.

Contrasting with prior studies that concentrated solely on predicting HDD failures or advancing HDD yield predictions, this model is distinguished by its comprehensiveness and efficiency in addressing both facets of the manufacturing process. It eschews the complex or resource-intensive algorithms and FS methods of earlier research, opting for a more pragmatic and robust approach with straightforward techniques suitable for categorical and imbalanced data. This model also offers greater adaptability and scalability, enabling the fine-tuning of feature selection and thresholds to align with the data and intended results.

# CHAPTER 3

# MEDTHODOLOGY

This chapter describes the methodology used to develop a rule-based expert system for defect prediction and test time reduction in HDD manufacturing. The methodology consists of five main steps: data collection and preparation, FS and ranking, DT modeling and evaluation, rule generation and defect prediction, and test time reduction and optimization. Figure 3.1 shows the overview of the methodology.

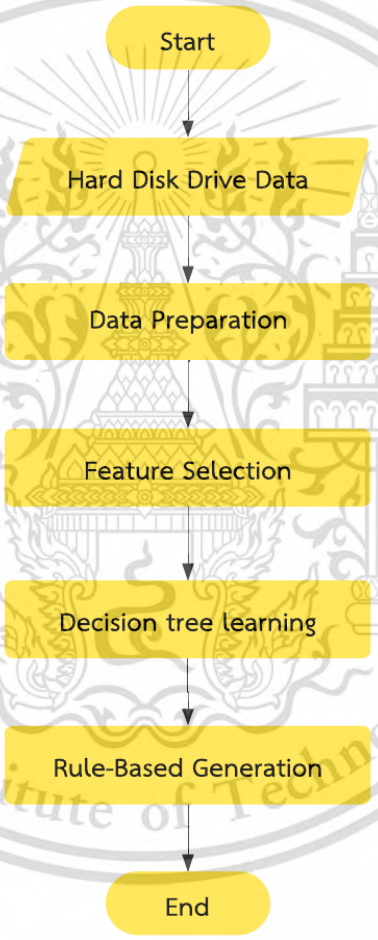


**Figure 3.1** Methodology flow chart

## 3.1 DATA COLLECTION AND PREPARATION

The data used in this study was obtained from a real-world HDD manufacturing process, consisting of a single timeframe of production. The data contains 53,451 instances and 43 features related to component vendors and assembly machines. All features are categorical data, and the data includes two classes: F (denoting defective instances) and P (denoting non-defective instances). The data was complete, without any missing values. The data was divided into five folds using stratified sampling, ensuring that each fold had the same proportion of classes as the original data. One fold was used as the test data, while the remaining four folds were used as the training data.

The experiments were performed using R code, which is shown in Appendix B. The dataset involved is proprietary and is not available for public access.

The experiments were carried out on a system running Windows 10, with R version 3.6.3. FS techniques such as IG, GR, $\chi^2$, and SU were implemented using the FSelector package [26]. DT algorithms like ID3 and CART were executed using the rpart package [27], and C4.5 was run through the RWeka package [28].

## 3.2 FEATURE SELECTION AND RANKING

Feature selection is a technique that aims to reduce the dimensionality of the data by selecting a subset of relevant features that can improve the performance of the prediction model. Feature selection can also enhance the interpretability and simplicity of the model, as well as reduce the computation time and complexity. In this study, four FS techniques were used to rank the features according to their importance for defect prediction. These techniques were IG, GR, $\chi^2$, and SU. Each technique computed a score for each feature based on different criteria, such as entropy, impurity, or correlation. The features were then sorted in descending order of their scores, indicating the most important features at the top.

## 3.3 DECISION TREE MODELING AND EVALUATION

Decision tree is a machine learning algorithm that can create a graphical representation of either a classification or regression problem. DT can handle both categorical and numerical data, and it is easy to interpret and explain. DT can also deal

with imbalanced data, which is the case in this study, as the number of defective instances is much lower than the number of non-defective instances. In this study, three DT algorithms were used to build prediction models for defect detection. These algorithms were ID3, C4.5, and CART. Each algorithm used a different criterion for feature selection and splitting, such as IG, GR, or Gini Index. Each algorithm was applied to the training data with all features as a baseline, and then with each FS technique ranking, starting with the top-ranked feature until achieving the highest accuracy. The accuracy of each model was measured by the confusion matrix, which shows the number of true positives ($TP$), true negatives ($TN$), false positives ($FP$), and false negatives ($FN$) for each class. The accuracy was calculated as the proportion of correctly classified instances over the total number of instances as shown in Eq. (3.1).

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} \quad (3.1)$$

## 3.4 RULE GENERATION AND DEFECT PREDICTION

Rule generation is a process that converts the DT model into a set of IF-THEN rules that can be used to classify new instances. Rule generation can simplify the DT and make it more understandable and applicable. In this study, the rules were generated from the best DT model, which was the one with the highest accuracy and the lowest number of features. The rules used categorical features to classify HDDs into passers or defectives. The rules were applied in a top-down manner, starting from the first rule and moving to the next rule if the condition was not satisfied. The class of the instance was determined by the first rule that matched the values of the features. The rules were then used to assign a failure probability to each hard drive in the test data, based on the proportion of defective instances in the leaf node of the DT. If the failure probability exceeded a predefined threshold, the hard drive was classified as defective; otherwise, it was classified as a passer.

## 3.5 TEST TIME REDUCTION AND OPTIMIZATION

Test time reduction is a goal that aims to optimize the manufacturing process by reducing the testing time and resource consumption of HDDs. Test time reduction

can be achieved by identifying defective drives before they undergo extensive testing, and subjecting them to a shorter testing process that only involves the most relevant test operations. This can also increase the test capacity and improve the quality of the product. In this study, the test time reduction was calculated by comparing the total test time of the current process and the proposed process. The current process involved testing all hard drives with the same average test time, regardless of their defect status as shown in Eq. (3.2).

$$TTT_{Current\ Process} = Q \times T \tag{3.2}$$

when $TTT$ is the total test time, $Q$ is the number of instances, $T$ is the average test time.

The proposed process involved testing the hard drives with different test times, depending on their predicted defect status as shown in Eq. (3.3). The test time of the proposed process was determined by the failure probability threshold, the short test divider, and the normal test time. The failure probability threshold was the cut-off value that separated the defective and non-defective drives. The short test divider was the factor that reduced the test time of the defective drives. The normal test time was the test time of the non-defective drives. The optimal values of the failure probability threshold and the short test divider were the ones that minimized the total test time of the proposed process.

$$TTT_{Proposed\ Process,\alpha} = \left(QTP_\alpha \times \frac{T}{r}\right) + \left(QFP_\alpha \times \frac{(1+r)\times T}{r}\right) + ((Q - QTP_\alpha - QFP_\alpha) \times T) \tag{3.3}$$

when $TTT$ is the total test time, $Q$ is the number of instances, $T$ is the average test time, $r$ is the short test divider, $QTP_\alpha$ is the number of true positives at threshold, and $QFP_\alpha$ is the number of false positives at threshold $\alpha$.

The total test time reduction was calculated as the difference between the total test time of the current process and the total test time of the proposed process at the optimal values. The total test time reduction was also evaluated by statistical tests, such as t-test and confidence interval, to determine its significance and reliability.

# CHAPTER 4

# RESULTS AND DISCUSSION

This chapter presents the results and discussion of the experiments conducted in this study. The experiments aimed to evaluate the performance of the rule-based expert system for defect prediction in HDD manufacturing, using FS techniques and DT algorithms. The performance metrics used were predictive accuracy, total model-building time, top N features, number of rules, and total test time. The results were compared with the baseline of using all features and the current production process.

## 4.1 FEATURE SELECTION AND DECISION TREE RESULTS AND COMPARISON

Table 4.1 provides a comparison of the accuracy of each DT algorithm when using all features and each FS technique. The table also includes the minimum and maximum values of the 95% confidence intervals for each accuracy value, representing the range of potential accuracy values if the experiment were to be conducted with different data samples.

From the data in Table 4.1, it is clear that the C4.5 algorithm, when using the IG technique, achieved the highest accuracy, with a mean value of 0.8817. This is closely followed by the ID3 algorithm also using the IG technique, with a mean value of 0.8813. The CART algorithm, when using all features, yielded the lowest accuracy, with a mean value of 0.8718. The non-overlapping confidence intervals for most cases indicate that these differences in accuracy are statistically significant.

These results suggest that FS techniques can enhance the accuracy of DT algorithms, as models using these techniques outperformed those using all features. Among the FS techniques, IG was the most effective, followed by $\chi^2$, SU, and GR. This indicates that these techniques can efficiently identify the most relevant and informative features for predicting HDD failure. In terms of DT algorithms, C4.5 and ID3 outperformed CART, suggesting that these algorithms can generate more accurate and robust models for the given dataset.

**Table 4.1 Comparison of accuracy at 95% confidence intervals**

| Decision Tree Algorithm | Feature Selection Technique | Accuracy | Minimum | Maximum |
|---|---|---|---|---|
| ID3 | All Features | 0.8759 | 0.8737 | 0.8782 |
| | IG | **0.8813** | 0.8793 | 0.8832 |
| | GR | 0.8808 | 0.8790 | 0.8826 |
| | $\chi^2$ | 0.8812 | 0.8792 | 0.8832 |
| | SU | 0.8810 | 0.8789 | 0.8832 |
| C4.5 | All Features | 0.8770 | 0.8755 | 0.8785 |
| | IG | **0.8817** | 0.8799 | 0.8836 |
| | GR | 0.8809 | 0.8791 | 0.8827 |
| | $\chi^2$ | 0.8816 | 0.8797 | 0.8835 |
| | SU | 0.8815 | 0.8797 | 0.8833 |
| CART | All Features | 0.8718 | 0.8695 | 0.8742 |
| | IG | **0.8813** | 0.8793 | 0.8832 |
| | GR | 0.8808 | 0.8790 | 0.8826 |
| | $\chi^2$ | 0.8811 | 0.8791 | 0.8832 |
| | SU | 0.8810 | 0.8788 | 0.8832 |

Table 4.2 displays the total times taken for building models using different DT algorithms and FS techniques. From the data in Table 4.2, it is evident that the C4.5 algorithm with the IG technique was the most efficient, requiring only 27.9 seconds. Conversely, the CART algorithm using the SU technique was the least efficient, taking the longest total time of 78.4 seconds. This suggests that the choice of both the DT algorithm and the FS technique can significantly impact the efficiency of model building.

**Table 4.2 Comparison of total times for building models (second)**

| Decision Tree Algorithm | Feature Selection Technique | Total times |
|---|---|---|
| ID3 | All Features | 99.0 |
| | IG | **28.6** |
| | GR | 62.8 |
| | $\chi^2$ | 41.6 |
| | SU | 78.0 |
| C4.5 | All Features | 146.5 |
| | IG | **27.9** |
| | GR | 57.4 |
| | $\chi^2$ | 29.7 |
| | SU | 49.4 |
| CART | All Features | 101.2 |
| | IG | **30.0** |
| | GR | 62.3 |
| | $\chi^2$ | 42.9 |
| | SU | 78.4 |

Table 4.3 illustrates the top N features used for building models with different DT algorithms and FS techniques. The table shows that FS techniques can significantly decrease the number of features needed for modeling, compared to using all features. This can enhance the efficiency and interpretability of the models, while also avoiding overfitting, noise, and redundancy.

Among the FS techniques, IG was the most effective, requiring the fewest features (6 for ID3 and CART, 7 for C4.5), followed by $\chi^2$ and GR. This suggests that these techniques can effectively identify the most relevant and informative features for predicting HDD failure.

In terms of DT algorithms, specifically, C4.5, CART, and ID3 required only 7, 6, and 6 features respectively when using the IG technique, compared to 43 features when using all features. This demonstrates the effectiveness of FS techniques in improving model performance.

**Table 4.3 Comparison of top N features for building models**

| Decision Tree Algorithm | Feature Selection Technique | Top N Features |
|---|---|---|
| ID3 | All Features | 43 |
| | IG | **6** |
| | GR | 9 |
| | $\chi^2$ | 7 |
| | SU | 10 |
| C4.5 | All Features | 43 |
| | IG | **7** |
| | GR | 10 |
| | $\chi^2$ | **7** |
| | SU | 10 |
| CART | All Features | 43 |
| | IG | **6** |
| | GR | 9 |
| | $\chi^2$ | 7 |
| | SU | 10 |

Table 4.4 presents the number of rules generated for building models with different DT algorithms and FS techniques. The table indicates that FS techniques can significantly reduce the number of rules needed for modeling, compared to using all features. This can enhance the efficiency and interpretability of the models, while also avoiding overfitting, noise, and redundancy.

Among the FS techniques, GR was the most effective, generating the fewest rules (29 for ID3, 7 for C4.5, and 28 for CART), followed by IG. This suggests that these techniques can effectively identify the most relevant and informative rules for predicting HDD failure.

In terms of DT algorithms, both C4.5 and ID3 generated fewer rules to achieve comparable or higher accuracy than CART, implying that these algorithms can generate more accurate and robust models for the given dataset. Specifically, C4.5 and ID3 generated only 7 and 50 rules respectively when using the IG technique, compared to 387 and 164 rules when using all features. This demonstrates the effectiveness of FS techniques in improving model performance.

**Table 4.4 Comparison of number of rules**

| Decision Tree Algorithm | Feature Selection Technique | N rules |
|---|---|---|
| ID3 | All Features | 164 |
| | IG | 50 |
| | GR | **29** |
| | $\chi^2$ | 73 |
| | SU | 71 |
| C4.5 | All Features | 387 |
| | IG | **7** |
| | GR | **7** |
| | $\chi^2$ | **7** |
| | SU | 8 |
| CART | All Features | 189 |
| | IG | 49 |
| | GR | **28** |
| | $\chi^2$ | 80 |
| | SU | 79 |

In summary, the results in Tables 4.1, 4.2, 4.3, and 4.4 indicate that FS techniques, especially IG, can significantly enhance the accuracy and efficiency of DT algorithms. IG, when incorporated with DT algorithms, proves to be an efficient and effective FS technique as it can pinpoint the most pertinent and informative features for predicting the class variable. Utilizing IG allows DT algorithms to decrease data dimensionality, prevent overfitting, noise, and redundancy, and boost the accuracy, efficiency, and interpretability of the models.

The effectiveness of integrating IG with three DT algorithms (ID3, C4.5, and CART) varies depending on the characteristics of the dataset and the application domain. For example, ID3 and C4.5 are more suitable for categorical features, while CART can manage both categorical and continuous features. ID3 and C4.5 employ entropy as the measure of impurity, while CART can use either entropy or Gini impurity. C4.5 uses normalized IG or GR as the splitting criterion, whereas ID3 uses the raw IG value. ID3 and C4.5 are more susceptible to overfitting, while CART employs pruning to prevent

overfitting. ID3 and C4.5 are more prevalent in machine learning and natural language processing, while CART is more adaptable and can be used for both classification and regression tasks.

However, the specific choices of the DT algorithm and FS technique may depend on the acceptable trade-off between accuracy and computational efficiency for a given application. The IG FS technique was found to be the most effective and efficient across all DT algorithms, influencing the total time required for building models, the number of features used, and the number of rules generated.

The results also display the statistical tests and confidence intervals for comparing the accuracy and total model-building time for each DT algorithm between using all features and each FS technique. The results confirm that using FS techniques resulted in statistically significant improvements over using all features for all DT algorithms in terms of accuracy and total model-building time. The results also reveal that C4.5 with IG achieved the best performance among all combinations, with an accuracy of 0.8817, a total model-building time of 27.9 seconds, and 7 rules.

## 4.2 RULE-BASED EXPERT SYSTEM RESULTS AND COMPARISON

The results from the rule-based expert system, which utilized the C4.5 algorithm with IG to generate rules, are presented. These rules were used to categorize HDDs as either passers (P) or defectives (F) based on assembly data. The rules are as follows:

1) If C29 is "V0", then the drive is defective (F).
2) If C29 is either "V1" or "V2" and C23 is "V0", then the drive is defective (F).
3) If C29 and C23 are either "V1" or "V2" and C29 is "V1", then the drive is a passer (P).
4) If C29 and C23 are either "V1" or "V2", C29 is "V2", and C43 is within ["V0", "V1", "V2", "V3", "V4", "V5", "V6"], then the drive is defective (F).
5) If C29 and C23 are either "V1" or "V2", C29 is "V2", C43 is within ["V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16", "V17", "V18", "V19", "V20", "V21", "V22", "V23", "V24", "V25", "V26", "V27", "V28", "V29"], and C26 is "V0", then the drive is defective (F).

6) If C29 and C23 are either "V1" or "V2", C29 is "V2", C43 is within ["V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16", "V17", "V18", "V19", "V20", "V21", "V22", "V23", "V24", "V25", "V26", "V27", "V28", "V29"], and C26 is "V1", then the drive is a passer (P).
7) If C29 and C23 are either "V1" or "V2", C29 is "V2", C43 is within ["V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16", "V17", "V18", "V19", "V20", "V21", "V22", "V23", "V24", "V25", "V26", "V27", "V28", "V29"], and C26 is either "V2" or "V3", then the drive is defective (F).

Table 4.5 displays the accuracy of each failure probability threshold. It provides the average accuracy and the accuracy for each of the five data subsets, known as folds. The table highlights that the optimal threshold range is between 0.15 and 0.70, where the accuracy peaks at 0.88172.

This rule-based system can be easily understood and implemented in HDD manufacturing processes. In addition, the wide range of failure probability thresholds can provide flexibility in application.

**Table 4.5 Comparison of accuracy for each failure probability cut-off (threshold)**

| Cut-off | Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | Average | Fold #1 | Fold #2 | Fold #3 | Fold #4 | Fold #5 |
| 0.00 | 0.13405 | 0.13535 | 0.13330 | 0.13611 | 0.13452 | 0.13096 |
| 0.05 | 0.13405 | 0.13535 | 0.13330 | 0.13611 | 0.13452 | 0.13096 |
| 0.10 | 0.13405 | 0.13535 | 0.13330 | 0.13611 | 0.13452 | 0.13096 |
| 0.15 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.20 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.25 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.30 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.35 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.40 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.45 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.50 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.55 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.60 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.65 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.70 | **0.88172** | **0.87934** | **0.88269** | **0.88064** | **0.88110** | **0.88485** |
| 0.75 | 0.88169 | **0.87934** | **0.88269** | **0.88064** | **0.88110** | 0.88466 |
| 0.80 | 0.88159 | 0.87924 | 0.88251 | 0.88045 | **0.88110** | 0.88466 |
| 0.85 | 0.87989 | 0.87092 | 0.88251 | 0.88045 | 0.88092 | 0.88466 |
| 0.90 | 0.87315 | 0.87092 | 0.87446 | 0.87100 | 0.87250 | 0.87689 |
| 0.95 | 0.87069 | 0.86877 | 0.87212 | 0.86848 | 0.87007 | 0.87399 |
| 1.00 | 0.86595 | 0.86465 | 0.86670 | 0.86389 | 0.86548 | 0.86904 |

## 4.3 TOTAL TEST TIME RESULTS AND COMPARISON

Table 4.6, derived from the optimal model, presents a comparison of the total test times (in millions of hours) for the current and proposed processes across various threshold values. The minimum values are highlighted in bold. The total test times for both processes are computed for $T$ = 720 hours and $r$ = 2. The findings reveal that the model with an optimal cut-off between 0.15 and 0.70 yields the least total test time.

The selection of the failure probability threshold range from 0.15 to 0.70 was influenced by accuracy-efficiency trade-offs, empirical evidence from our experiment, and its practical implications for HDD manufacturing. Lower thresholds enhance efficiency but increase false positives, while higher thresholds reduce false positives but extend testing time. Our experiment identified the optimal range to be between 0.15 and 0.70. This range ensures that hard drives with either low or high failure probabilities undergo standard testing, optimizing testing time and resource utilization while preserving quality.

Figure 4.1 depicts the optimal failure probability cut-off against the total test time for each fold and the average. It demonstrates that the optimal cut-off, which minimizes the total test time of the proposed process, lies between 0.15 and 0.70.

Figure 4.2 displays the boxplots of the total test time for the current and proposed processes at the best threshold for each fold and the average. The boxplots outline the minimum, maximum, median, and quartiles of the total test time for each process and each fold. The figure shows that the proposed process has a lower total test time than the current process for all folds and the average. It also indicates that the proposed process exhibits less variation in the total test time than the current process, signifying its superior consistency and stability. The figure supports the conclusion that the proposed process can significantly reduce the total test time compared to the current process.

**Table 4.6 Comparison of the total test time (in millions of hours) of the current process and the proposed process for each failure probability cut-off (threshold)**

| Cut-off | Current Process | Proposed Process | | | | | |
|---|---|---|---|---|---|---|---|
| | | Average | Fold #1 | Fold #2 | Fold #3 | Fold #4 | Fold #5 |
| 0.00 | 7.6968 | 10.5136 | 10.5044 | 10.5192 | 10.4976 | 10.5098 | 10.5372 |
| 0.05 | 7.6968 | 10.5136 | 10.5044 | 10.5192 | 10.4976 | 10.5098 | 10.5372 |
| 0.10 | 7.6968 | 10.5136 | 10.5044 | 10.5192 | 10.4976 | 10.5098 | 10.5372 |
| 0.15 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.20 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.25 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.30 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.35 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.40 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.45 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.50 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.55 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.60 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.65 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.70 | 7.6968 | **7.6363** | **7.6410** | **7.6352** | **7.6324** | **7.6367** | **7.6360** |
| 0.75 | 7.6968 | 7.6364 | **7.6410** | **7.6352** | **7.6324** | **7.6367** | 7.6367 |
| 0.80 | 7.6968 | 7.6368 | 7.6414 | 7.6360 | 7.6331 | **7.6367** | 7.6367 |
| 0.85 | 7.6968 | 7.6433 | 7.6734 | 7.6360 | 7.6331 | 7.6374 | 7.6367 |
| 0.90 | 7.6968 | 7.6692 | 7.6734 | 7.6669 | 7.6694 | 7.6698 | 7.6666 |
| 0.95 | 7.6968 | 7.6787 | 7.6817 | 7.6759 | 7.6792 | 7.6792 | 7.6777 |
| 1.00 | 7.6968 | 7.6969 | 7.6975 | 7.6968 | 7.6968 | 7.6968 | 7.6968 |

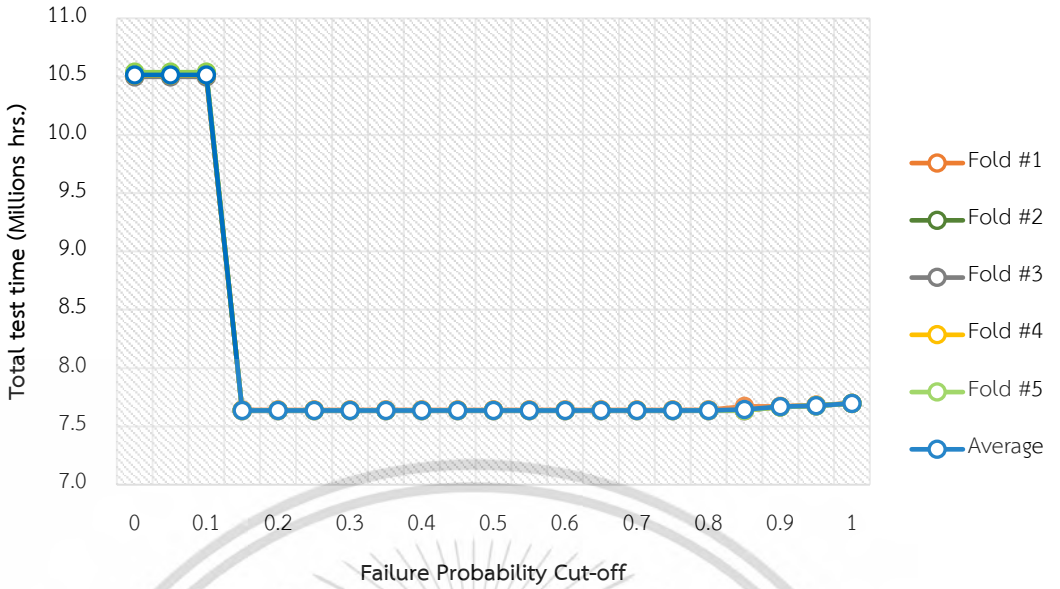


**Figure 4.1** Optimal failure probability cut-off vs total test time for each fold and the average

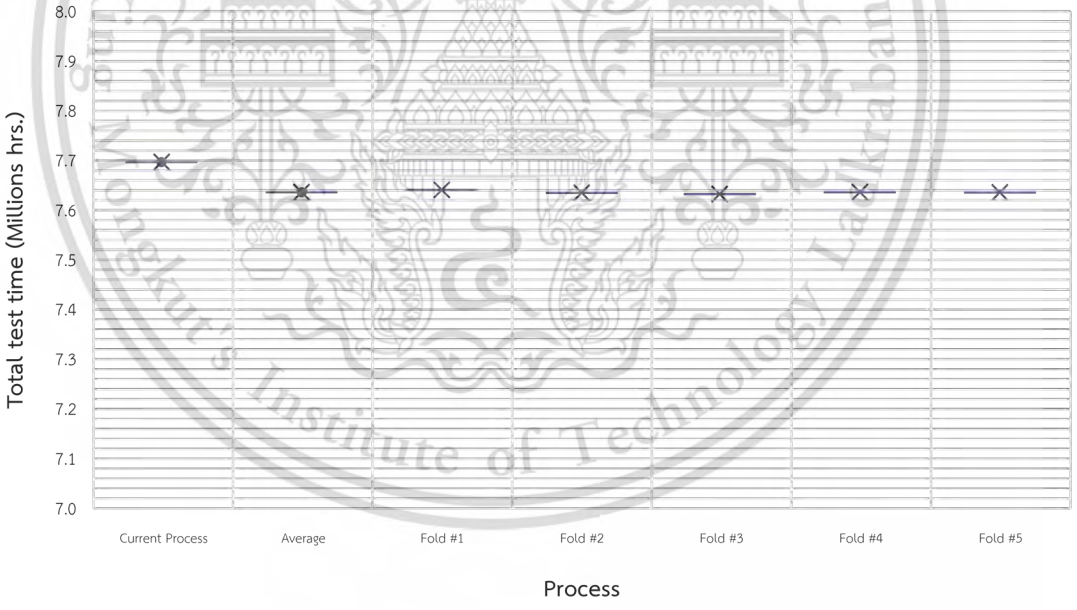


**Figure 4.2** Boxplots of the total test time of the current process and the proposed process at the best threshold for each fold and the average

Additionally, Table 4.7 provides the differences in total test times between the current process and the proposed process at the optimal threshold. Table 4.8 presents the average difference between the total test time and the 95% confidence interval, calculated from the data in Table 4.7. The results indicate that the total test time of the proposed process is significantly lower than that of the current process.

**Table 4.7 Differences in total test times between the current process and the proposed process at the best threshold**

| Fold # | (1) Current Process | (2) Proposed Process | (1) - (2) |
|---|---|---|---|
| 1 | 7.6968 | 7.6410 | 0.0558 |
| 2 | 7.6968 | 7.6352 | 0.0616 |
| 3 | 7.6968 | 7.6324 | 0.0644 |
| 4 | 7.6968 | 7.6367 | 0.0601 |
| 5 | 7.6968 | 7.6360 | 0.0608 |
| | | Mean | 0.0605 |
| | | Standard deviation | 0.0031 |

**Table 4.8 Mean total reduction test time at 95% confidence intervals between normal process and proposed process**

| Total reduction test time (Million Hrs.) | | |
|---|---|---|
| Mean | Min | Max |
| 0.0605 | 0.0578 | 0.0633 |

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

This chapter concludes the thesis by summarizing the main findings and contributions of the research and discussing the practical of the proposed rule-based expert system for HDD manufacturing, identifying the limitations and challenges encountered during the research process, and providing recommendations and future directions for further research in this area.

## 5.1 SUMMARY OF FINDINGS AND CONTRIBUTIONS

The main objective of this research was to develop a rule-based expert system that leverages predictive models constructed from assembly process data to identify potentially defective hard drives before undergoing extensive testing. By preemptively identifying defects, this approach aimed to substantially reduce testing time and enhance tester capacity, thereby optimizing the manufacturing process of HDDs.

To achieve this objective, the research employed a real-world dataset obtained from HDD manufacturing, consisting of a single timeframe of production. The dataset contained 53,451 instances and included 26 features related to components vendors and 17 features related to production machines. All 43 features were categorical data, and the dataset included two classes: F (denoting defective instances, which number 7,165) and P (denoting non-defective instances, which number 46,286).

The research explored the combination of four FS techniques (IG, GR, $\chi^2$, and SU) and three DT algorithms (ID3, C4.5, and CART) to construct predictive models that classify hard drives into passers or defectives based on assembly data. The performance of the models was evaluated in terms of prediction accuracy, modeling time, top N features, rule generation, and test time reduction. In the comparison of decision tree algorithms, it was found that C4.5 surpassed both ID3 and CART. This suggests that these algorithms have the ability to create models that are not just more precise, but also more resilient when applied to the given dataset. This observation highlights the capability of C4.5 in managing intricate datasets and producing dependable predictions. Furthermore, the application of FS and DT algorithms resulted in comparable accuracy levels, attributable to several key factors. First, the inherent robustness of DT algorithms enables them to effectively manage noise and unrelated

data without a significant decrease in accuracy. Second, the efficiency of FS techniques facilitates the identification of the most influential features, thereby preserving the model's accuracy even after feature reduction. Lastly, DTs are particularly adept at handling imbalanced datasets, as they can learn effectively from both classes, ensuring consistent accuracy throughout.

The experimental results revealed that IG coupled with the C4.5 algorithm yielded the most favorable results in terms of prediction accuracy, modeling efficiency, and rule generation. Moreover, the research established that setting the failure probability threshold between 0.15 and 0.70 provided the shortest total test time for the proposed process, as supported by a 95% confidence level. This achievement represented a statistically significant enhancement compared to the existing manufacturing process.

The main contributions of this research are as follows:

1) It proposed a novel rule-based expert system that utilizes data from the assembly process to pinpoint defects in HDDs, with the caveat that the accuracy of these predictions is a determining factor.
2) It demonstrated the effectiveness of using FS techniques and DT algorithms to formulate minimal and interpretable classification rules that can be applied to hard drive defect prediction.
3) It showed the potential of using predictive modeling, tailored testing, and feature selection to streamline defect detection, reduce resource consumption, and ultimately optimize HDD production.

## 5.2 LIMITATIONS AND CHALLENGES

The research also encountered some limitations and challenges during the research process. Some of these limitations and challenges are:

The research was based on a single production timeframe and a single HDD product, which may limit the generalizability and applicability of the results to other timeframes or products. The expert system may need to be updated or retrained to accommodate changes in the production process, such as new components, machines, or vendors.

The research relied on the quality and availability of the assembly data, which may vary depending on the sources and methods of data collection, processing, and storage. The assembly data may also contain errors, noise, or outliers that could affect the accuracy and reliability of the predictive models.

The research did not account for all factors influencing the defect probability, such as environmental conditions, human errors, or hardware failures, which may not be captured by the assembly data. The expert system may not be able to detect or handle these factors, which could lead to false positives or false negatives in defect prediction.

The research used a fixed number of features and a predefined threshold for defect prediction, which may not be optimal or adaptable to different scenarios or objectives. The expert system may need to incorporate feature selection and threshold optimization techniques that can dynamically adjust to the data and performance criteria.

## 5.3 RECOMMENDATIONS AND FUTURE DIRECTIONS

Based on the findings, contributions, implications, limitations, and challenges of the research, some recommendations and future directions for further research in this area are:

To extend the scope and validity of the research by using more production timeframes and HDD products, and comparing the results with different datasets and domains.

To improve the quality and availability of the assembly data by implementing data quality assurance and management techniques, such as data cleaning, validation, integration, and security.

To incorporate more factors influencing the defect probability by collecting and analyzing additional data sources, such as environmental sensors, human feedback, or hardware diagnostics.

To enhance the flexibility and scalability of the expert system by applying adaptive or hybrid feature selection and threshold optimization techniques that can learn from the data and performance feedback.

# REFFERENCES

[1] Samattapapong N. and Afzulpurkar N. “A production throughput forecasting system in an automated hard disk drive test operation using GRNN” **Journal of Industrial Engineering and Management**, vol. 9, no. 2, 2016. pp. 330-358.

[2] Sankar S., Shaw M., Vaid K., and Gurumurthi S. “Datacenter scale evaluation of the impact of temperature on hard disk drive failures” **ACM Transactions on Storage (TOS)**, vol. 9, no. 2, 2013. pp. 1-24.

[3] Song W., Ovcharenko A., Knigge B., Yang M., and Talke F.E. “Effect of contact conditions during thermo-mechanical contact between a thermal flying height control slider and a disk asperity” **Tribology International**, vol. 55, 2012. pp. 100-107.

[4] Ye Z.S., Xie M., and Tang L.C. “Reliability evaluation of hard disk drive failures based on counting processes” Reliability Engineering & System Safety, vol. 109, 2013. pp. 110-118.

[5] Sharma H. and Kumar S. “A survey on decision tree algorithms of classification in data mining” **International Journal of Science and Research (IJSR)**, vol. 5(4), 2016. pp. 2094-2097.

[6] Lavanya D. and Rani D. “Evaluation of Decision Tree Classifiers on Tumor Datasets” **International Journal of Emerging Trends Technology in Computer Science (IJETTCS)**, vol. 2, 2013. pp. 418-423.

[7] Jovic A., Brkic K., and Bogunovic N. “A review of feature selection methods with applications” **Information and Communication Technology Electronics and Microelectronics (MIPRO)**, vol. 38, 2015. pp. 1200-1205.

[8] Hirunyawanakul A., Kaoungku N., and Kerdprasop K. “Efficient Machine Learning Methods for Hard Disk Drive Yield Prediction Improvement” **International Journal of Machine Learning and Computing**, vol. 10, Feb. 2020. pp. 240–246.

[9] Spekking R. “Seagate Barracuda Green ST2000DL003 - platter and head” **Wikimedia Commons**, CC BY-SA 4.0, 2023.

[10] Al Mamun A., Guo G., and Bi C. “Hard Disk Drive: Mechatronics and Control” **CRC press**, 2017.

[11] Taktak-Meziou M., Chemori A., Ghommam J., and Derbel N. “Mechatronics of hard disk drives: Rise feedback track following control of a R/W head” **Mechatronics: Principles, Technologies and Applications**, 2015.

[12] Kondo M., Lim S., Koita T., Namihira T., and Tokoro C. “Application of electrical pulsed discharge to metal layer exfoliation from glass substrate of hard-disk platter” **Results in Engineering**, vol. 12, Dec. 2021.

[13] Pimpanont P. and Chutima P. “Applications of value engineering in head stack assembly process: A case study” **International Journal of Materials, Mechanics and Manufacturing**, vol. 4, no. 1, 2016. pp. 46-51.

[14] Oboe R., Marcassa F., Capretta P., and Chrappan Soldavini F. “Realization of a hard disk drive head servo-positioning system with a voltage-driven voicecoil motor” **Microsystem Technologies**, vol. 9, 2003. pp. 271-281.

[15] Shamkhalichenar H., Bueche C. J., and Choi J.-W. “Printed Circuit Board (PCB) Technology for Electrochemical Sensors and Sensing Platforms” **Biosensors**, vol. 10, no. 11, 2020.

[16] Quinlan J.R. “Induction of decision trees” **Machine learning 1**, vol. 1, 1986. pp.81-106.

[17] Quinlan J.R. “C4.5: programs for machine learning” **Morgan Kaufmann Publishers Inc.**, San Francisco, CA, USA, 1993.

[18] Breiman L., Friedman J.H., Olshen R.A., and Stone C.J. “Classification and regression trees” **Monterey**, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[19] Li J., Ji X., Jia Y., and Zhu B. “Hard drive failure prediction using classification and regression trees” **Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.**, 2014. pp. 383-394

[20] Xu C., Wang G., Liu X., Guo D., and Liu T.-Y. “Health status assessment and failure prediction for hard drives with recurrent neural networks” **IEEE Transactions on Computers**, vol. 65.11, 2016. pp. 3502-3508.

[21] Aussel N., Jaulin S., Gandon G., Petetin Y., Fazli E., and Chabridon S. “Predictive models of hard drive failures based on operational data” **Machine Learning and Applications (ICMLA) 16th IEEE International Conference**, 2017. pp. 619-625.

[22] Xiao J. “Disk failure prediction in data centers via online learning” **Proceedings of the 47th International Conference on Parallel Processing**, 2018. pp. 1-10.

[23] Ahmad W., Khan S.A., Kim C.H., and Kim J.-M. “Feature Selection for Improving Failure Detection in Hard Disk Drives Using a Genetic Algorithm and Significance Scores” **Applied Sciences**, vol. 10, no. 9, 3200, 2020.

[24] Siltepravet A., Sinthupinyo S., and Chongstitvatana P. “Improving Quality of Products in Hard Drive Manufacturing by Decision Tree Technique” **Int. J. Comput. Sci. Issues**, vol. 9, no. 3, 2012. pp. 1-5.

[25] Hirunyawanakul A., Kaoungku N., Kerdprasop N., and Kerdprasop K. “Feature Selection to Improve Performance of Yield Prediction in Hard Disk Drive Manufacturing” **International Journal of Electrical and Electronic Engineering & Telecommunications**, vol. 9, no. 6, 2020.

[26] Romanski P., Kotthoff L., Schratz P. **FSelector: Selecting Attributes** [Online]. Available: https://cran.r-project.org/web/packages/FSelector/index.html.

[27] Therneau T., Atkinson B., Ripley B. **Rpart: Recursive Partitioning and Regression Trees** [Online]. Available: https://cran.r-project.org/web/packages/rpart/index.html.

[28] Hornik K., Buchta C., Hothorn T., Karatzoglou A., Meyer D., Zeileis A. **RWeka: R/Weka Interface** [Online]. Available: https://cran.r-project.org/web/packages/RWeka/index.html.

# Appendix A

# PUBLICATION LIST

# Publication List

## JOURNAL

1. **S. Kirdponpattara**, P. Sooraksa and V. Boonjing, "Building a Rule-Based Expert System to Enhance the Hard Disk Drive Manufacturing Processes," in IEEE Access, vol. 12, pp. 29558-29570, 2024, doi: 10.1109/ACCESS.2024.3369443. (Q1, Scopus and Web of Science)
2. **S. Kirdponpattara**, P. Sooraksa and V. Boonjing, "An Alternative Method for Upgrading the Conventional Decision Tree Algorithm," Sensors and Materials, vol. 36, no. 4, pp. 1461-1471, 2024, doi: 10.18494/SAM5029. (Q3, Scopus and Web of Science)

# Appendix B

# R CODE

# R Code

SOURCE CODE

```r
#----- Library -----#
list_packages <- c(
  "dplyr",
  "tidyr",
  "data.table",
  "rpart",
  "rpart.plot",
  "FSelector",
  "stringr",
  "caret",
  "e1071",
  "magrittr",
  "lattice",
  "ggplot2",
  "tibble",
  "plyr",
  "Matrix",
  "data.table",
  "knitr",
  "RWeka",
  "data.tree",
  "stats",
  "RWekajars",
  "reservr"
)
#Install packages
new.packages <-
  list_packages[!(list_packages %in% installed.packages()[, "Package"])]
if (length(new.packages))
  install.packages(new.packages, repos = 'https://cloud.r-project.org/')
#Load packages
lib <- lapply(list_packages, library, character.only = TRUE)
#----- Library -----#


#----- Function -----#
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

#----- Function -----#


start_time <- Sys.time()
```

```
45
46 path = 'C:/'
47
48 # Import to charecter
49 dt_raw <-
50  read.csv(paste(path, 'DATA.csv', sep = ""),
51           colClasses = c('character'))
52
53 # Section: Data prepocessing --------------------------------
54 # 1. Remove singleton attribute ---------------------------------
55
56 col_str = ""
57 col_toomany = ""
58 col_unwanted = ""
59 col_singleton = ""
60 for (i in 1:(length(dt_raw))) {
61  # Check unique value
62  if (length(unique(dt_raw[[i]])) == 1) {
63   col_singleton = paste(col_singleton, colnames(dt_raw[i]), sep = ", ")
64  }
65  else if (length(unique(dt_raw[[i]])) > 100) {
66   col_toomany = paste(col_toomany, colnames(dt_raw[i]), sep = ", ")
67  }
68  else {
69   if (grepl(tolower(colnames(dt_raw[i])), "date", fixed = TRUE) ||
70     grepl(tolower(colnames(dt_raw[i])), "yield", fixed = TRUE)
71     ||
72     grepl(tolower(colnames(dt_raw[i])), "run_type", fixed = TRUE)) {
73    col_unwanted = paste(col_unwanted, colnames(dt_raw[i]), sep = ", ")
74   }
75   else {
76    col_str = paste(col_str, colnames(dt_raw[i]), sep = ",")
77   }
78  }
79 }
80 col_str = substr(col_str, 2, nchar(col_str))
81 col_singleton = substr(col_singleton, 3, nchar(col_singleton))
82 col_toomany = substr(col_toomany, 3, nchar(col_toomany))
83 col_unwanted = substr(col_unwanted, 3, nchar(col_unwanted))
84
85 # Write log singleton & too many
86 write.table(
87  paste("Singleton Attribute:", col_singleton),
88  'log.csv',
89  row.names = FALSE,
90  col.names = FALSE
91 )
92 write.table(
93  paste("Too Many Values Attribute(>100):", col_toomany),
```

```
  'log.csv',
  row.names = FALSE,
  col.names = FALSE,
  append = TRUE
 )
 write.table(
     paste("Unwanted Attribute:", col_unwanted),
     'log.csv',
     row.names = FALSE,
     col.names = FALSE,
     append = TRUE
   )

   # Select non singleton column
   dat <- strsplit(col_str, split = ",")
   dt_select <- dt_raw[unlist(dat)]
   rm(dt_raw)

   # Find unique value for each column
   dt_uni_col <- lengths(lapply(dt_select, unique))

   # 2. Re-code each column to numeric and One-hot ----------------------------
   ---

   # Move EVENT_STATUS to First
   dt_full_recode <-
    dt_select[, c(which(colnames(dt_select) == "EVENT_STATUS"),
          which(colnames(dt_select) != "EVENT_STATUS"))]
   dt_full_recode_int <-
    dt_select[, c(which(colnames(dt_select) == "EVENT_STATUS"),
          which(colnames(dt_select) != "EVENT_STATUS"))]

   for (i in 2:(length(dt_select))) {
    # Check unique value
    cname <- paste("C", toString(i - 1), sep = "")
    dt_full_recode[, cname] <-
     as.character(as.numeric(factor(dt_select[[i]])) - 1)
    dt_full_recode_int[, cname] <-
     as.numeric(factor(dt_select[[i]])) - 1
   }

   # Output for FC
   dt_sel_recode                =                cbind(dt_full_recode[c(1)],
dt_full_recode[c((length(dt_select) + 1):length(dt_full_recode))])
   dt_sel_recode_int            =            cbind(dt_full_recode_int[c(1)],
dt_full_recode_int[c((length(dt_select) + 1):length(dt_full_recode))])

   # Section: Feature Important ---------------------------------
```

```
140   st_time <- Sys.time()
141   # 1. IG ---------------------------------
142   dr1_ig <- information.gain(EVENT_STATUS ~ ., dt_sel_recode)
143   dr1_ig <-
144    cbind(col_names = rownames(dr1_ig), data.frame(dr1_ig, row.names =
    NULL))
145   dr1_ig <- dr1_ig[order(-dr1_ig$attr_importance), ]
146   dr1_ig$algo = "IG"
147   dr1_ig$attr_importance_norm <- normalize(dr1_ig$attr_importance)
148   dr1_ig$time <- Sys.time() - st_time
149
150   st_time <- Sys.time()
151   # 2. GR ---------------------------------
152   dr2_gr <- gain.ratio(EVENT_STATUS ~ ., dt_sel_recode)
153   dr2_gr <-
154    cbind(col_names = rownames(dr2_gr), data.frame(dr2_gr, row.names =
    NULL))
155   dr2_gr <- dr2_gr[order(-dr2_gr$attr_importance), ]
156   dr2_gr$algo = "GR"
157   dr2_gr$attr_importance_norm <- normalize(dr2_gr$attr_importance)
158   dr2_gr$time <- Sys.time() - st_time
159
160   st_time <- Sys.time()
161   # 3. Chi-squared ---------------------------------
162   dr3_chi <- chi.squared(EVENT_STATUS ~ ., dt_sel_recode)
163   dr3_chi <-
164    cbind(col_names = rownames(dr3_chi), data.frame(dr3_chi, row.names =
    NULL))
165   dr3_chi <- dr3_chi[order(-dr3_chi$attr_importance), ]
166   dr3_chi$algo = "Chi"
167   dr3_chi$attr_importance_norm <- normalize(dr3_chi$attr_importance)
168   dr3_chi$time <- Sys.time() - st_time
169
170   st_time <- Sys.time()
171   # 4. SU ---------------------------------
172   dr4_su <- symmetrical.uncertainty(EVENT_STATUS ~ ., dt_sel_recode)
173   dr4_su <-
174    cbind(col_names = rownames(dr4_su), data.frame(dr4_su, row.names =
    NULL))
175   dr4_su <- dr4_su[order(-dr4_su$attr_importance), ]
176   dr4_su$algo = "SU"
177   dr4_su$attr_importance_norm <- normalize(dr4_su$attr_importance)
178   dr4_su$time <- Sys.time() - st_time
179
180   write.table(
181    dr1_ig,
182    'al.csv',
183    row.names = FALSE,
184    append = FALSE,
```

```
    quote = FALSE,
    sep = ","
  )
  write.table(
    dr2_gr,
    'al.csv',
    row.names = FALSE,
    append = TRUE,
    quote = FALSE,
    sep = ",",
    col.names = FALSE
  )
  write.table(
    dr3_chi,
    'al.csv',
    row.names = FALSE,
    append = TRUE,
    quote = FALSE,
    sep = ",",
    col.names = FALSE
  )
  write.table(
    dr4_su,
    'al.csv',
    row.names = FALSE,
    append = TRUE,
    quote = FALSE,
    sep = ",",
    col.names = FALSE
  )

  dr_list <- list(dr1_ig, dr2_gr, dr3_chi, dr4_su)

  val_list <- c(1:10)

  # Section: Feature Selection ------------------------------

  # 1. Select N and run model -------------------------------
  top_n <- c(58)
  df_param <- tibble()
  df_pred <- tibble()

  top_n_previous <- 0

  # All columns
  dt_algo <- c(1, 7, 8)

  dt_sel_recode_tmp <- dt_sel_recode

```

```
for (a in 1:length(dt_algo)) {
 for (k in 1:length(dr_list)) {
  dt_a <- do.call(rbind.data.frame, dr_list[k])

  dt_sel_recode <- dt_sel_recode_tmp

  for (j in 1:length(val_list)) {
   st_time <- Sys.time()

   # All columns
   top_n <- val_list[j]

   if (top_n < 44) {
    # Select Top N columns
    lst_col <- paste(dt_a$col_names[1:top_n], sep = "")

    #Normal
    dt_tmp0 <-
     cbind(dt_sel_recode[c(1)], dt_sel_recode[lst_col])

    algo <- dt_a$algo[1]

    #Randomly shuffle the data
    set.seed(123)
    dt_tmp0 <- dt_tmp0[sample(nrow(dt_tmp0)), ]

    print(colnames(dt_tmp0))

    #-----------------Decision tree-----------------#

    #Create 5 equally size folds
    k_fold <- 5

    folds <-
     cut(seq(1, nrow(dt_tmp0)), breaks = k_fold, labels = FALSE)

    for (K in 1:k_fold) {
     #Segement your data by fold using the which() function
     dt_tmp <-
      transform(dt_tmp0, EVENT_STATUS = as.factor(EVENT_STATUS))

     testIndexes <-
      which(folds == K, arr.ind = FALSE)
     df_train <- dt_tmp[-testIndexes, ]

     df_test <- dt_tmp[testIndexes, ]

     #i <- 1
     for (i in 1:1) {
```

```
283           minsplit = i
284           minbucket = floor(minsplit / 3)
285           maxdepth = 11
286           cp = 0
287
288           if (dt_algo[a] == 1) {
289            # ID3
290            model <- "ID3"
291            control <-
292             rpart.control(
293              minsplit = minsplit,
294              minbucket = minbucket,
295              maxdepth = maxdepth,
296              cp = cp
297             )
298            d3d <-
299             rpart(
300              EVENT_STATUS ~ .,
301              data = df_train,
302              method = "class",
303              control = control,
304              parms = list(split = "information")
305             )
306            nrule <-
307             nrow(as.data.table(rpart.rules(
308              d3d, cover = T, nn = T
309             )))
310           }
311           else if (dt_algo[a] == 7) {
312            # C4.5
313            model <- "C4.5"
314            d3d <-
315             J48(EVENT_STATUS ~ .,
316               data = df_train,
317               control = Weka_control(R = TRUE))
318            #d3d <- train(EVENT_STATUS ~ ., method = "J48", data = df_train)
319            nrule <- width(as.party(d3d))
320           }
321           else if (dt_algo[a] == 8) {
322            # CART
323            model <- "CART"
324            control <-
325             rpart.control(
326              minsplit = minsplit,
327              minbucket = minbucket,
328              maxdepth = maxdepth,
329              cp = cp
330             )
331            d3d <-
```

```
332             rpart(
333               EVENT_STATUS ~ .,
334               data = df_train,
335               method = "class",
336               control = control,
337               parms = list(split = "gini")
338             )
339           nrule <-
340             nrow(as.data.table(rpart.rules(
341               d3d, cover = T, nn = T
342             )))
343         }
344
345         ### Test Model
346         print(
347           paste(
348             paste("Method A/C: ", model, sep = ""),
349             "FC:",
350             paste(k, " ", algo, sep = ""),
351             "ncol:",
352             top_n,
353             "K:",
354             K,
355             "Round:",
356             i,
357             "",
358             val_list[j]
359           )
360         )
361
362         try(
363           {
364             pred <- predict(d3d, newdata = df_test, type = 'class')
365             pred_prob <- predict(d3d, newdata = df_test, type = 'prob')
366             t <- table(df_test$EVENT_STATUS, pred)
367             df_softmax <- softmax(pred_prob)
368
369             df_pred_tmp <- cbind(K, df_test$EVENT_STATUS, pred, pred_prob,
    df_softmax)
370             colnames(df_pred_tmp) <- c('ROUND', 'REAL', 'PREDICT', 'P_FAIL',
    'P_PASS', 'S_FAIL', 'S_PASS')
371             df_pred <- rbind(df_pred, df_pred_tmp)
372           })
373         cat({
374         })
375
376         cf <- confusionMatrix(t)
377
378         df_cf <- as.data.frame(cf$overall)
```

```
            names(df_cf)[1] <- i
            df_cf <- as.data.frame(t(df_cf))
            df_cf$method <- model
            df_cf$fc <- paste(k, ".", algo, sep = "")
            df_cf$threshold <- val_list[j]
            df_cf$ncol <- top_n
            df_cf$iteration <- i
            df_cf$folds <- K
            df_cf$time <- Sys.time() - st_time
            df_cf_tmp <- as.data.frame(cf$table)
            df_cf$FF <- df_cf_tmp$Freq[[1]]
            df_cf$FP <- df_cf_tmp$Freq[[3]]
            df_cf$PP <- df_cf_tmp$Freq[[4]]
            df_cf$PF <- df_cf_tmp$Freq[[2]]

            df_cf$Positive_Predictive_Value <-
              df_cf_tmp$Freq[[1]] * 100 / (df_cf_tmp$Freq[[1]] +
df_cf_tmp$Freq[[3]])
            df_cf$False_Omission_Rate <-
              df_cf_tmp$Freq[[2]] * 100 / (df_cf_tmp$Freq[[2]] +
df_cf_tmp$Freq[[4]])

            df_param <- rbind(df_param, df_cf)
          }
        }
      }

      top_n_previous <- top_n
    }
  }

}
# Summarize CF

dt_model_result_sum <-
  ddply(
    df_param,
    c("method", " fc ", " ncol ", " threshold ", " iteration "),
    summarise,
    mean_acc = mean(Accuracy),
    sd = sd(Accuracy),
    n = length(Accuracy),
    mean_ppv = mean(Positive_Predictive_Value),
    sd_ppv = sd(Positive_Predictive_Value),
    mean_for = mean(False_Omission_Rate),
    sd_for = sd(False_Omission_Rate),
    ci = 1.96 * sd(Accuracy) / sqrt(length(Accuracy)),
    total_time = sum(time),
    mean_nrule = round(mean(nrule), 0),
```

```
426       mean_FF = mean(FF),
427       mean_FP = mean(FP),
428       mean_PP = mean(PP),
429       mean_PF = mean(PF)
430     )
431
432   # Select max Acc
433   dt_model_result_sum_final <- dt_model_result_sum %>%
434    group_by(method, fc) %>%
435    filter(mean_acc == max(mean_acc, na.rm = TRUE))
436
437   # Select min N Col
438   dt_model_result_sum_final <- dt_model_result_sum_final %>%
439    group_by(method, fc) %>%
440    filter(ncol == min(ncol, na.rm = TRUE))
441
442   write.csv(df_param, '1_Result.csv', row.names = FALSE)
443   write.csv(dt_model_result_sum,  '2_Result_Summary.csv',  row.names  =
    FALSE)
444   write.csv(dt_model_result_sum_final,
445         '3_Result_Summary_Final.csv',
446         row.names = FALSE)
447
448   end_time <- Sys.time()
449   run_time <- end_time - start_time
450   print(run_time)
```

# AUTHOR BIOGRAPHY

| | |
|---|---|
| Name | Mr. Suppakrit Kirdponpattara |
| Date of Birth | March 30, 1983, in Phangnga |
| Address: | 79/226, Permsin Road, Ao Ngoen, Sai Mai, Bangkok, 10220 |

Educational Background:

| | |
|---|---|
| 2005: | Bachelor of Engineering in Computer Engineering<br>Prince of Songkla University (PSU), Thailand. |
| 2010: | Master of Science in Computer Science<br>King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand. |

Research Area:

Data Mining

AI & Machine Learning

Work Experience:

| | |
|---|---|
| 2005 - 2007 | Computer Engineer at Charoen Pokphand Foods Public Company Limited, Thailand. |
| 2010 - Present | Staff Engineer at Seagate Technology (Thailand) Co., Ltd. Thailand. |

Honorary Awards:

| | |
|---|---|
| 2010 | Honor Pin Award with Certificate for the Excellent Master Degree in Science from The Professor Dr.Tab Nilanidhi Foundation, Thailand. |