APPLICATION OF DEFUZZIFICATION-FREE HIERARCHICAL FUZZY INFERENCE RULE GENERATION METHOD TO SOFTWARE FAULT PREDICTION PROBLEMS AND FUZZY RULE DISCUSSION

DURULAŞTIRMASIZ HİYERARŞİK BULANIK ÇIKARSAMA KURAL ÜRETME YÖNTEMİNİN YAZILIM HATA KESTİRİMİ PROBLEMLERİNE UYGULANMASI VE BULANIK KURAL TARTIŞMASI

NAZLI ECE UYKUR

PROF. DR. EBRU AKÇAPINAR SEZER

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Computer Engineering

2019

To my family…

ABSTRACT

APPLICATION OF DEFUZZIFICATION-FREE HIERARCHICAL FUZZY INFERENCE RULE GENERATION METHOD TO SOFTWARE FAULT PREDICTION PROBLEMS AND FUZZY RULE DISCUSSION

Nazlı Ece UYKUR

Master of Science, Department of Computer Engineering

Supervisor: Prof. Dr. Ebru AKÇAPINAR SEZER

January 2020, 90 pages

Fuzzy inference systems, a sub-branch of artificial intelligence, are decision support systems based on fuzzy set theory. The knowledge is transmitted to these systems through linguistic fuzzy rules. In general, the rule base is determined by expert opinion. However, the rules are generated by automatic rule generation methods when the problem domain becomes complex and expert knowledge is insufficient. In the studies conducted so far, the dataset used to produce the rules with automatic rule generation methods has not been analyzed in detail. The dataset may mislead the model and the ruleset obtained from the project may not always be the most accurate one. The most accurate and generalizable ruleset may be produced from another project. Thus, the rules can be transferred to another application for the same problem domain. The goal of this study is to solve new problems more effectively and faster way by using the ruleset of a different project without having to re-generate the rules when the dataset is changed. Therefore, unlike the data-driven approaches, the knowledge is transferred from the source project to the target project to make a predictive model in the current project. We investigate the portability of the rules by generating them from five different projects for Software Fault Prediction problem. The rulesets are generated by three automatic rule generation methods, namely "Wang-Mendel", "Interval-Valued Fuzzy Reasoning Method with Tuning and Rule Selection" and rule production approach of "Defuzzification-free Hierarchical Fuzzy System". In addition, automatic rule generation methods were compared with Artificial Neural Networks, which is a data-driven machine learning method to compare the success of the model. The results of the experiments indicate that the rules generated from more consistent datasets instead of own dataset of a project significantly improves the performance of existing fuzzy inference systems.

Keywords: Fuzzy rule learning, Portability of fuzzy systems, Software fault prediction, Fuzzy inference systems, Transfer learning

ÖZET

DURULAŞTIRMASIZ HİYERARŞİK BULANIK ÇIKARSAMA KURAL ÜRETME YÖNTEMİNİN YAZILIM HATA KESTİRİMİ PROBLEMLERİNE UYGULANMASI VE BULANIK KURAL TARTIŞMASI

Nazlı Ece UYKUR

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. Ebru AKÇAPINAR SEZER

Ocak 2020, 90 sayfa

Yapay zekanın bir alt dalı olan bulanık çıkarım sistemleri, bulanık küme teorisine dayanan karar destek sistemleridir. Bilgi bu sistemlere dilsel bulanık kurallarla iletilir. Genel olarak, kural tabanı uzman görüşü ile belirlenir. Ancak, problem alanı karmaşık hale geldiğinde ve uzman bilgisi yetersiz olduğunda kurallar otomatik kural üretme yöntemleriyle üretilir. Şimdiye kadar yapılan çalışmalarda, otomatik kural üretme yöntemleri ile üretilen kurallarda kullanılan veri seti ayrıntılı bir şekilde analiz edilmemiştir. Veri seti modeli yanlış yönlendirebilir ve projeden elde edilen kural kümesi her zaman en iyi kural kümesi olmayabilir. En doğru ve genellenebilir kural kümesi başka bir projeden üretilebilir. Böylece, kurallar aynı problem alanı için başka bir uygulamaya aktarılabilir. Bu çalışmanın amacı, veri seti değiştirildiğinde kuralları yeniden oluşturmak zorunda kalmadan farklı bir projenin kural kümesini kullanarak yeni sorunları daha etkili ve daha hızlı bir şekilde çözmektir. Bu nedenle, veri odaklı yaklaşımların aksine, bilgi, mevcut projede öngörücü bir model oluşturmak için kaynak alandan hedef alanına aktarılır. Kuralların taşınabilirliği, kuralları Yazılım Arıza Tahmin problemi için beş farklı projeden üreterek araştırıldı. Kural kümeleri, Wang-Mendel, Ayarlama ve Kural Seçimi ile Aralıklı Değerli Bulanık Muhakeme yöntemi ve Durulaştırmasız Hiyerarşik Bulanık Çıkarsama Sistemi'nin kural üretim yaklaşımı olan üç otomatik kural üretme yöntemi ile üretildi. Ayrıca, modelin başarısını karşılaştırmak için otomatik kural üretme yöntemleri, veri odaklı bir makine öğrenme yöntemi olan Yapay Sinir Ağları ile karşılaştırıldı. Deney sonuçları, bir projenin kendi veri kümesi yerine daha tutarlı veri setlerinden oluşturulan kuralların, mevcut bulanık çıkarım sistemlerinin performansını önemli ölçüde geliştirdiğini göstermektedir.

Anahtar Kelimeler: Bulanık kural öğrenme, Bulanık sistemlerin taşınabilirliği, Yazılım hata tahmini, Bulanık çıkarım sistemleri, Transfer öğrenimi

ACKNOWLEDGEMENTS

First of all, I would like to thank and express my gratitude to my thesis advisor, Prof. Dr. Ebru AKÇAPINAR SEZER, for her supervision, patience and encouragement throughout the research. She always keeps me motivated to explore new ideas and solutions in my research and contributed to this work with her experience, knowledge and energy.

In addition, I want to thank my company TÜBİTAK for supporting me to continue my studies. I would like to thank my colleagues in my project and especially to my old teammates, Ezgi Taşkomaz, Fatma Emül and Anıl Özberk who never stop supporting me and staying motivated at all times.

Finally, I would like to thank my family. Words cannot express how grateful I am for their support, love, and sacrifices that they have made throughout my life. They deserve special thanks for their limitless support and encouragement. This work would not have been possible without them.

TABLE OF CONTENTS

FIGURES

TABLES

SYMBOLS AND ABBREVIATIONS

Symbols

µ	Membership degree
f	Fuzzy sets
I	Input
O	Output

Abbreviations

FISs	Fuzzy inference systems
MD	Membership degree
MF	Membership function
SFP	Software fault prediction
WM	Wang-Mendel
IVTURS	Interval-Valued Fuzzy Reasoning Method with Tuning and Rule Selection
DF-HFS	Defuzzification-free Hierarchical Fuzzy System
MISO	Multi-input single-output
FIE	Fuzzy inference engine
COS	Center of sums
COG	Center of gravity
COA	Centroid of area
BOA	Bisector of area
IVFSs	Interval-valued fuzzy sets

IV-FRBCS	Interval-valued fuzzy rule based classification system
IV-REFs	Interval-valued restricted equivalence functions
RF	Rating factor
loc	Line of code
v(g)	Cyclomatic complexity
ev(g)	Essential complexity
iv(g)	Design complexity
v	Volume
d	Difficulty
i	Intelligence
n	Length
e	Effort
KEEL	Knowledge Extraction based on Evolutionary Learning
L	Low
M	Moderate
H	High
FR	Frequent rules
ROC-AUC	Area under receiver operating characteristic curve
TP	True positive
FP	False positive
TPR	True positive rate
FPR	False positive rate

1. INTRODUCTION

Fuzzy logic is a mathematical approach that provides simple solutions to the control of uncertain, time-varying, complex, not well-defined systems that cannot be solved by a classical logic method. In this approach, the classical set theory is replaced by fuzzy sets. Fuzzy inference systems (FISs) are the extension of rule-based inference by fuzzy set or fuzzy logic approach. These systems are used in the design of artificial intelligence based systems such as control systems, decision support systems, and expert systems, which are created for various problems. Although the success of FIS inevitably depends on the determination of membership function (MF) and the choice of fuzzy reasoning methods, the selection of fuzzy rules is always crucial to the accuracy of inference.

One of the most significant advantages of using FISs is linguistically expressible fuzzy rules, which are the main components of the system, are interpretable and portable. An interpretable FIS allows a user to give each fuzzy set a linguistic value and express the behavior of the system in an understandable manner by inspecting the fuzzy rules [1,2]. This strongly depends on model structure, such as the number of input variables and the complexity of fuzzy rules and sets, the shape of MFs, etc. Decision trees and fuzzy cognitive maps are also interpretable systems like FISs. These systems describe the model in the context of a cause-effect relationship. The domain expert does not know these relationships and it is not known whether a correct relationship is established or not. Machine learning algorithms may associate unrelated features. Since the internal structure of the model is unknown, it cannot be moved to another application field. In the field of data mining and information discovery, it is essential for decision support systems to extract knowledge from databases and represent it clearly or to make the process transparent to the user. Thus, it is easier for the user to validate the knowledge obtained from the FIS with the domain information, and to be convinced that the system behavior is reliable. This knowledge is provided to FISs by fuzzy rules.

Rules can be determined by the expert or can be obtained by learning algorithms. It may not always be possible to find experts in the field of study. In today's conditions, there are not enough experts in every field, and even if they exist, they cannot allocate sufficient time. As a result of different perspectives of experts on the same subject, the information that the experts have adopted and presented correctly may be inconsistent. In order to overcome such problems can only be achieved by generating rules with automatic rule generation methods. Therefore, we have created rules with three automatic rule generation methods namely “Wang-Mendel’s rule generation method” (WM) [3], “Interval-Valued Fuzzy Reasoning Method with Tuning and Rule Selection” (IVTURS) [4] and the expert-cooperated rule production scheme proposed in “Defuzzification-free Hierarchical Fuzzy System” (DF-HFS) [5] on different datasets. In order to compare the performance of FISs, widely used ANN is also implemented.

The rules do not always have to be generated from its data of the current project. We investigated the rules that were created for several datasets, namely, CM1, JM1, KC1, KC2, PC1 to discuss which dataset should be used to obtain the best set of rules for software fault prediction (SFP) problem by performing several FISs.

1.1. Motivation

The quality of rules is related to the quality of the dataset. Therefore, when the rule is generated from the data instead of expert knowledge, the dataset should be analyzed in detail. In addition, there will be information loss when the dataset is divided, and it directly affects the rule coverage of domain. In the literature, the rules are always produced from the target dataset (the dataset of the project itself). More specifically, both fuzzy applications [6–8] of SFP and the other studies [9–13] used the dataset of current projects while generating rules. However, the rules should be produced from the highest quality dataset to ensure that FIS has a better prediction on the problem domain.

Transfer learning focuses on solving a different but relevant problem in machine learning by using the knowledge obtained from a previously solved problem. Based on this approach, the best ruleset is determined from the source dataset, and it can be transferred to other problems with respect to dataset characteristics of the target domain. The portability property of FIS makes it possible to use pre-acquired knowledge to improve the effectiveness and learning accuracy in another project similar to the source project

[14,15]. Unlike machine learning algorithms that learn from domain-specific data, selected highest quality ruleset can be used on different domains for the same problem. We criticize that the ruleset of an investigated problem should always be produced from its dataset. In order to determine the best set of rules for a problem, we investigated whether it is better to use the corresponding project's dataset or to use another dataset that performs better for this problem domain.

The rule base, one of the main components of the FIS, is generally determined by domain experts who have comprehensive knowledge of the problem. The established rule base should bring a general solution to the problem and reflect the domain knowledge of the FIS. However, expressing this knowledge is a challenging task for the expert in real-world problems. In practice, the input space of the given data is usually high dimensional. In these circumstances, the expert cannot create consistent and complementary rules because s/he has difficulty relating expert system variables with the output. Because of this bottleneck, researchers prefer using automatic rule generation methods that work with previously acquired knowledge.

In the literature, there are several automatic rule generation algorithms were performed based on evolutionary optimization algorithms [4,16], genetic algorithms [17,18], clustering [19], linear solutions [3,12], data-driven machine learning algorithms such as decision tree [20] and association rule mining [21]. Although data-driven rule learning methods are widely used and produce reasonable rules, they are always domain-specific and limited with the dataset boundaries. Furthermore, data-driven approaches are insufficient to explain in a human-understandable manner since the decision is hidden within the network. On the other hand, the rules obtained by automatic rule generation methods are interpretable and represent the domain-related know-how with linguistic fuzzy rules.

In order to investigate the best ruleset to be used in the project, some automatic rule generation methods were used based on the SFP problem. The inter-module dependence and increasing complexity of the software have made it difficult to deliver high-quality and sustainable software at a low cost. Therefore, SFP is an essential activity to reduce maintenance efforts and improve software quality by locating faults and making improvements to the code before the software is released. In recent years, many studies have been made on the SFP problem [6,8,22–27]. This problem has an advantage in terms of the dataset since there are various datasets in this domain, and the selected datasets contain the same input linguistic variables. The utilized datasets CM1, JM1, KC1, KC2, PC1 [9,28,29] were obtained from PROMISE Software Engineering Repository [30]. We generate the rules by performing several automatic rule generation methods on these datasets, namely, WM, IVTURS, and rule generation scheme of DF-HFS. Several FISs were generated by different combinations of selected rule generation methods and datasets.

Misleading rules derived from the rule generation methods may be due to data corruption. The main argument of this thesis is the rule learning data investigation should be conducted according to its ability to provide a general solution that brings the subject to portability of the generated rules.

The set of rules produced for a specific problem does not have to be generated from the own dataset of the problem. FISs are created with a set of rules provided from a different project that can give more accurate results. At this point, the vital issue is that the selected dataset should be generalizable and reflects the problem domain in the most comprehensive way. Therefore, the rules should be produced from the most appropriate dataset by investigating the data of different projects.

1.2. Contributions

1. In the experiments conducted on SFP, the ruleset produced from each project was transferred between projects based on the transfer learning approach. We observed that the best ruleset is obtained from more consistent datasets instead of their project. This demonstrates that the portability feature of the rules between different projects.

2. We have observed that the FISs that we generate are more successful by comparing them with the ANN algorithm, which is highly preferred in classification problems.

3. In this study, the transfer learning approach was performed for the first time using FISs on the SFP problem. This approach protects the confidentiality of the source dataset since only fuzzy rules are transferred to other projects.

1.3. The Outline of the Thesis

Chapter 2 presents the background of fuzzy logic, FISs, rule generation algorithms, ANN, and SFP. Chapter 3 presents related works on automatic rule generation methods, transfer learning, and SFP. Chapter 4 presents comparisons of fuzzy rule generation algorithms and ANN for SFP problem. Chapter 5 presents the experiments and results on FISs. The last chapter, Chapter 6 presents the conclusion and future work.

2. BACKGROUND

2.1. Fuzzy Logic

Aristotle is the founder of the science of classical logic. Aristotle systematized the previous studies (384-322 BC) about logic and made it a branch of science. Formal logic is based on three main laws with respect to the law of identity, the law of non-contradiction, the law of impossibility of the third state [31]. In these laws, he argues that everything is identical to itself, that an object cannot be another object than itself, and finally, there is no in-between situation. Aristotle's understanding of logic has been recognized as an authority in all around the world for centuries.

In classical logic or Aristotle’s logic, when a question is asked to any proposition, the answer depends only on two parameters, true or false. In computer science, the value of the answer is “1” if it is true and “0” if it is false. Aristotle logic is not enough to explain the real-world uncertainties. In 1965, Zadeh [32] expanded the classical Aristotle logic and laid the foundations of fuzzy logic based on the fuzzy set theory. Fuzzy logic transforms the human knowledge experience into rule bases and draws conclusions that correspond to a specific mathematical function of each rule base. The basic idea behind fuzzy logic is expressed in linguistic variables such as a proposition tall, short, very tall, very short, very very short, very very tall, etc. Classical logic, which argues that an entity is only an element of a set, leaves its place in the fuzzy logic that an entity can be more than one element of the set according to the values that an asset takes.

The value that determines how much an input value belongs to a term of the linguistic variable is called the MD, which varies from “0” to “1”. MFs are the changes of membership degrees (MDs) within a subset. For example, let us define a subset of *TALL* that will answer the question “Who is tall?” according to both logic. As seen in Figure 2.1, according to the classical set logic, a person who is 160 cm is not in the tall people set, and even 169 cm person is not in the tall people set. However, according to the fuzzy logic, 160 cm person is not called as short. Because it is partly within the set of tall people. In fuzzy logic, 160 cm one can be tall with 0.6 MDs, one 170 cm tall with 0.7 MDs, and one 180 cm tall with 1.0 MDs. Larger MDs are considered less fuzzy, while smaller MDs are considered fuzzier.

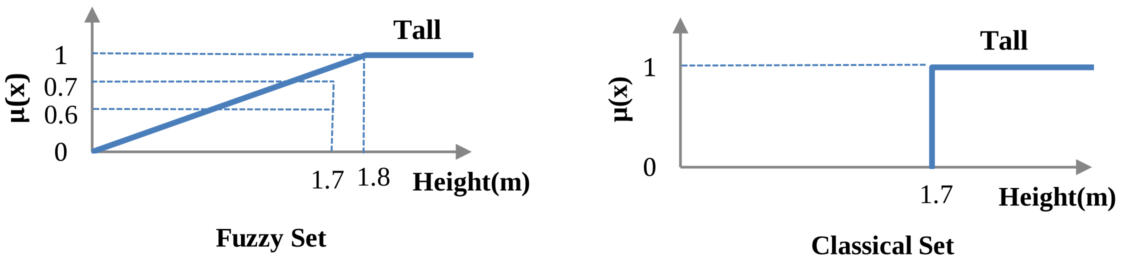


Figure 2.1 Comparison of the fuzzy set and the classical set

In Figure 2.2, the borders of the fuzzy set are shown by the Venn scheme. Here, the element “a” is the absolute element of the fuzzy set of “A”. The MD of this element is expressed as “1”. The MD is considered as “0” since element “b” does not belong to the fuzzy set “A”. The element “c” is in the fuzzy set “A” at a certain level. This is indicated by a MD in the range of [0, 1].

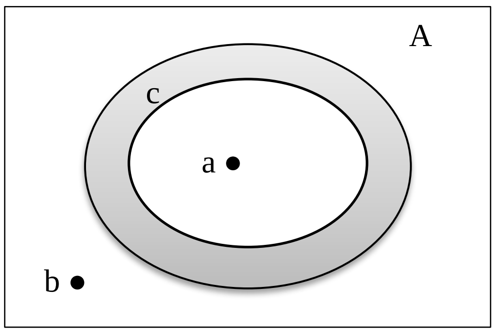


Figure 2.2 Borders of a fuzzy set

2.1.1. Membership Functions

The possible basic figures that MFs can take are shown in Figure 2.3. Each qualified definition used in the spoken language is written as an MF. Membership classes are determined at each point, and the boundaries applied.

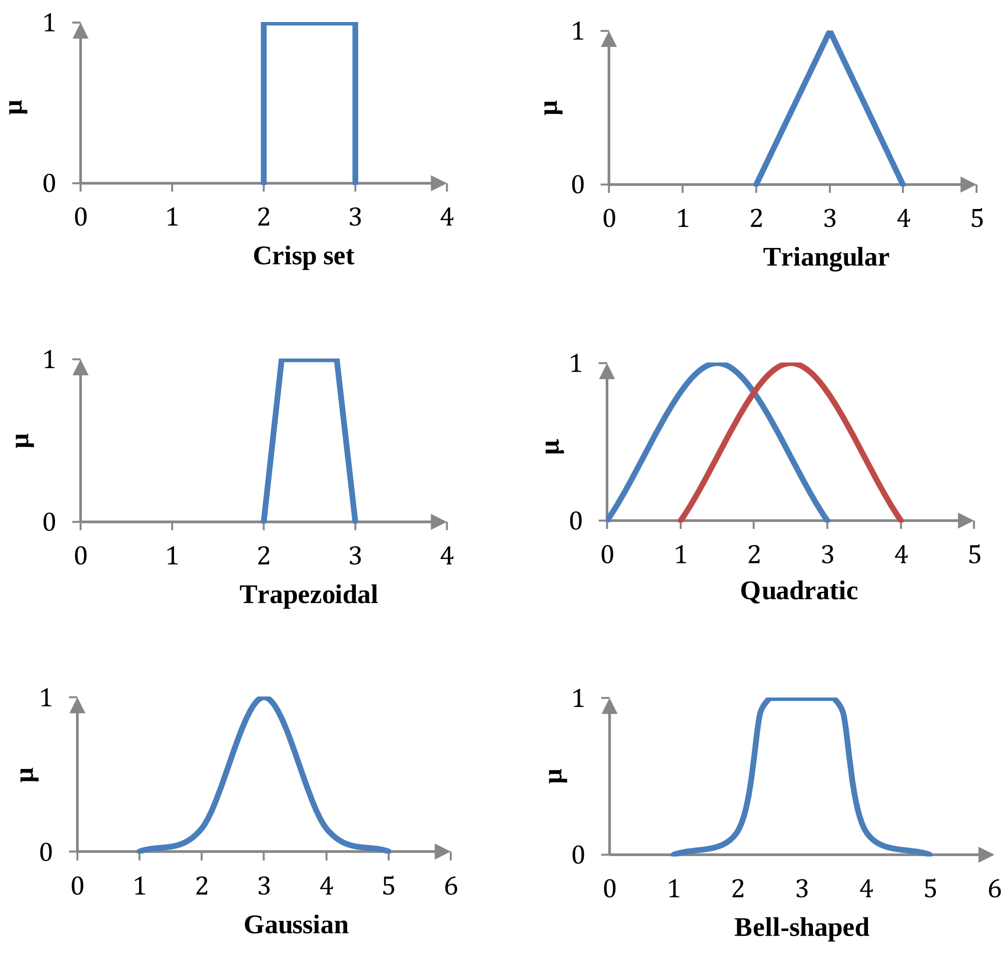


Figure 2.3 Several types of membership functions

In fuzzy logic, it is necessary to identify appropriate MFs to be used to specify the boundaries of regions that facilitate linguistic expression and to determine the membership weights of a real-world problem. MFs define system parameters. There is no restriction on the number and form of MFs, and it depends entirely on the desire and experience of the designer.

2.2. Fuzzy Inference

2.2.1. Fuzzy Inference Systems

A set of fuzzy rules from an expert is required to establish an FIS. The MDs of several fuzzy input and output sets are defined by a set of curves. Then, this relationship can be plotted between the input and output sets. "*If weather is cool then decrease the engine* *speed*” rule is given, the temperature and the speed of the motor represent the input and the output variables, respectively. These fuzzy sets will form a fuzzy region representing the set of all associations of the inputs and outputs of the rule. The size of the fuzzy region reflects the fuzziness of the rule. The area becomes smaller as the fuzzy set becomes certain.

FIS is the process of obtaining outputs from inputs by using fuzzy rules generated by using expert knowledge. In this sense, the fuzzy system approaches a mathematical cause-effect function.

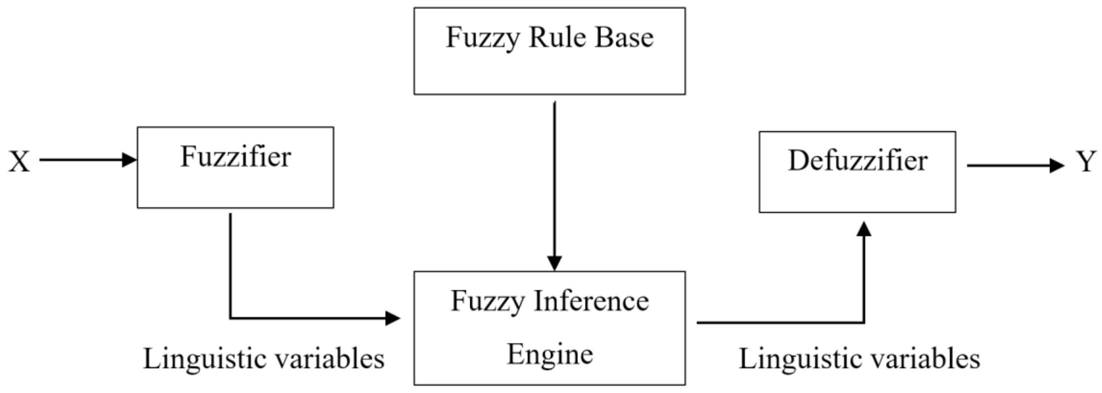


Figure 2.4 Fuzzy Inference System

Figure 2.4 represents the configuration of a simple FIS. Here, in the “*Fuzzifier*“ stage, the real-world values are converted to membership values in fuzzy sets. Then, “Fuzzy Inference Engine” (FIE) converts “IF-THEN” rules to a defined fuzzy relationship in the input-output space. In the “*Defuzzifier*” stage, the fuzzy set is converted to real-world values.

For example, let us consider a “fuzzy air conditioner” based on the four rules, hence four fuzzy sets to match the air temperature to the engine speeds. The temperature sets “*cold, cool, warm and hot”* and the engine speed sets “*very slow, slow, fast and very fast*” cover all fuzzy inputs and outputs. For example, the temperature of 20 degrees can be 30 percent cool and 70 percent warm. In addition, the weather is zero percent cool, warm, and hot. The “cool and warm” rules will activate both “slow and medium” engine speeds. Both rules contribute the same to the final speed of the engine. The fuzzy relationships derived from the defined rules form the fuzzy output set. This type of output cannot help controllers running in a binary system when in fuzzy form. The last step is a "*defuzzifier*" process that the fuzzy output value becomes a single numerical value. Thus, the system can be operated by a quantitative temperature input with the help of the temperature and engine speed sets to obtain a precise and convenient speed output.

Mostly, words are used to meet the variables that we use in our daily life. For example, in the sentence "it is very cold today", the *linguistic variable* is the air temperature of today, and the *output value* is very cold.

In fuzzy systems, fuzzy sets represent words in the area in which they are defined. When extracting a fuzzy model of the speed of a car, we first need to determine the area in which it is defined. This range is between the maximum and minimum speed a car can reach. After deciding this area, fuzzy sets are determined to represent linguistic variables such as "*very slow, slow, fast and very fast"* (Figure 2.5).

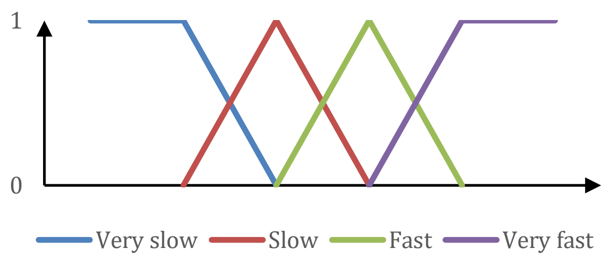


Figure 2.5 Speed of the car with fuzzy sets

The control of the systems can be provided by rules established using human knowledge and experience. The linguistic form of fuzzy rule is shown in Eq. (1) where *I* and *O* represent input and output linguistic variables, respectively, *m* is the number of input and f represents fuzzy sets.

$$IF\ I_1\ is\ f_1\ AND/OR\ I_2\ is\ f_2\ AND/OR\ \dots\ I_m\ is\ f_k\ THEN\ O\ is\ f_p \qquad (1)$$

2.2.2. Fuzzy Inference Models

The FIS combines the implications of each rule to determine how the entire system will output under the inputs. The main components of inference differ in some cases, and it leads to differentiation of consequent of fuzzy rules. Three main types of rule-based inference models were created based on the result consequent of fuzzy rules that are Mamdani, Takagi-Sugeno, and Tsukamoto models. The fuzzy systems used within the scope of the thesis were performed through the Mamdani type inference model.

2.2.2.1. Mamdani Model

The Mamdani model is a rule-based inference method that requires expert knowledge and can be applied to solve any problem. Therefore, it forms the baseline of other fuzzy inference models and is still widely used. Firstly, it was used for the control of a steam engine with the verbal rules produced with human experience by Mamdani and Assilian in 1975 [33]. Mamdani type fuzzy model consists of following these steps;

- The MD of the input variables varying in the range of [0, 1] is determined using fuzzy expressions in the fuzzification phase.
- The rules are determined using logical operators ("and", "or").
- The output fuzzy sets are aggregated for each rule.
- Fuzzy output sets are converted to crisp numbers in the defuzzification phase.

Mamdani has been proposed to express existing qualitative knowledge in the form of "IF-THEN" rules shown in Eq. 2.

$$\mathrm{K_i} = IF\ x\ is\ \mathrm{A_i}\ THEN\ y\ is\ \mathrm{B_i} \quad , i = 1,2,3,\dots,r \tag{2}$$

The "x" is the linguistic input variable, and "A_i" is the premise linguistic term. Likewise, "y" is the linguistic output variable, and "B_i" is the consequent linguistic term. A_i and B_i linguistic terms are fuzzy sets determined in their value set. MFs of the premise and consequent fuzzy sets are shown in Eq. 3.

$$\mu(x): X \rightarrow [0,1] \text{ } and \text{ } \mu(y): Y \rightarrow [0,1] \tag{3}$$

The rule base R = {K_i | i = 1, 2, 3…r} and "A" and "B" sets creates the information base of linguistic model.

2.2.2.2. Takagi-Sugeno Kang Model

"Takagi-Sugeno" fuzzy logic was first used in 1985 [34]. The fuzzification of input variables and the fuzzy logic operations are the same as for Mamdani fuzzy modeling. Two models differ in terms of the output MFs. In Sugeno type fuzzy modeling, output MFs are linear or constant, and for the input variable, it uses a mathematical function instead of fuzzy sets. They are called zero-order when output MFs are constant, and when they are in the form of linear equation, they are called first-order Sugeno fuzzy model. The first-order Sugeno type fuzzy rule form is shown in Eq. 4. The "a", "b" and "c" represent linguistic variables. The "X" and "Y" are fuzzy sets, and $f(a, b)$ is a mathematical function.

$$\text{IF a is X AND b is Y THEN c is } f(a, b) \tag{4}$$

The widely used "zero-order Sugeno fuzzy model" rule form is shown in Eq. 5. In this case, the output of each fuzzy rule is constant.

$$\text{IF a is X AND b is Y THEN c is } k\text{, where k is constant.} \tag{5}$$

The Sugeno model works well with the optimization and adaptive techniques and improves the results by optimizing the output parameters. It can use linear techniques to control nonlinear systems. On the other hand, high dimensional input parameters and subsets make it difficult to train the data and increase the number of output parameters that must be determined to obtain results.

2.2.2.3. Tsukamoto Model

In the “Tsukamoto model” [35], the consequent part of each “IF-THEN” fuzzy rule is represented by a fuzzy set with a monotone MF. Only monotone incremental or decreasing functions can be used for the output value. The “firing strength” of the rule creates the output as a crisp value for each rule. The crisp output is calculated by the “weighted average” in Tsukamoto FISs. The Tsukamoto rule form is represented in Eq.6.

$$IF\ x\ is\ \mathrm{A_i}\ THEN\ y\ is\ \mathrm{B_i} \quad , i = 1,2,3, \dots, n, \mathrm{B_i} \tag{6}$$

The overall output is always crisp value, even if the inputs are fuzzy. The output is calculated by the Eq. 7.

$$\mathrm{B(x)} = \frac{\sum_{i=1}^{n} \mathrm{B_i^{-1}} A_i(x)}{\sum_{i=1}^{n} A_i(x)} \tag{7}$$

This model is not used frequently because it is not as transparent as the other inference models. The differences between the inference models are represented in Table 2.1.

Table 2.1 Inference models for FIS

Inference Type	**Fuzzification**	**Inference process**	**Defuzzification**
Mamdani	Singleton	Min-max	Center of Gravity
Sugeno	Singleton	Min-product; Order 0	Weighted Average
Tsukamoto	Singleton	Min-max	Weighted Average

2.2.3. Mamdani Type Fuzzy Inference System Components

The critical point of FIS is to establish the knowledge, experience, intuition, and control strategy of a specialist system operator as a knowledge base in inference design. Inferences are not performed by complex and classical algorithms but by linguistic rules based on knowledge and experience. As shown in Figure 2.4, FIS contains four essential components that are "*fuzzifier*", "*fuzzy rule base*", "*fuzzy inference engine*" and "*defuzzifier*".

2.2.3.1. Fuzzification

"Fuzzification" is the converting process of numeric input variables (crisp data) into their corresponding fuzzy set. In other words, these variables, which are defined by fuzzy sets via MFs, are given a label to provide linguistic quantity.

In many applications, the inputs and outputs of fuzzy systems are crisp numbers. The inference mechanism models the thinking mechanism of human-based on fuzzy values and cannot be applied to the system because its output is fuzzy values. Therefore, crisp numbers are converted to linguistic variables, which is the name of one of the fuzzy sets, as shown in Figure 2.5.

2.2.3.2. Fuzzy Rule Base

The "fuzzy rule base" is the database in which the control rules are stored. When developing the rule base for a system, the input values that affect the output of the system should be determined. Fuzzy rules are generally created from expert knowledge. The rules are generated by automatic rule generation methods using dataset when expert knowledge is not sufficient.

The rule-base includes control rules that determine the behavior of the FIS. It consists of many parallel rules and audit variables used in decision-making. These rules explain the logical relationships between the inputs-outputs of the system. The output of the FIE is obtained by evaluating the rules describing the system. The rules are created with commands "***IF***" and "***THEN***" which are defined by system and control variables, respectively. The fuzzy rule forms for a multiple-input single-output (MISO) system is shown in Eq. 1. In a fuzzy rule "***IF*** x is T ***THEN*** y is Z", "x is T" and "y is Z" represent **premise** and **consequent** parts, respectively.

2.2.3.3. Fuzzy Inference Engine

The FIE is the leading block where the control algorithm is executed, and the decision-making stage takes place. The inputs of the defuzzification unit are the outputs of the FIE. FIE accesses the knowledge base and processes fuzzy rules and linguistic variables from the fuzzification interface. As a result of this process, it decides the control action. The rules in the fuzzy rule base are applied to the linguistic variables, and the control action is created by the chosen logical inference mechanism.

FIE architecture can be summarized as follows [33]:

- "Getting the fuzzy inputs."
- "Applying the fuzzy operators."
- "Applying the implication methods."
- "Aggregating the output fuzzy sets."
- "Creating the fuzzy outputs."

In the fuzzy rule base, the premise part of a rule is interconnected by the *AND* (min, prod) method or *OR* (max, probor) methods, and they reflect an MD to the output representing the rule degree. Then, each fuzzy rule consequent is determined by implication methods. Mamdani's minimum operator MAX-MIN or Larsen's product operator MAX-DOT implication methods are used to reshape the consequent of each fuzzy rule [33].

The MFs of all pre-clipped or scaled rule consequents are considered and combined into a single fuzzy set is called as an aggregation process (Figure 2.6).

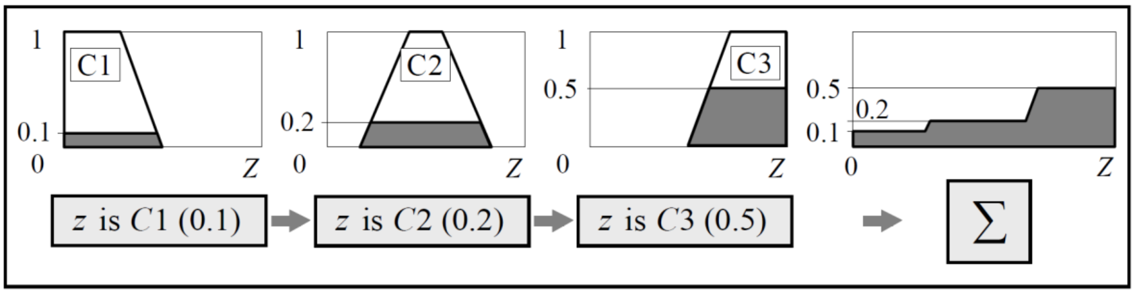


Figure 2.6 Aggregation of rule outputs

2.2.3.4. Defuzzification

The inference unit generates the fuzzy output values by evaluating the fuzzy inputs that come from the fuzzification interface under the fuzzy rule base. “Fuzziness” helps in evaluating rules, but the output of a system must be a “crisp” number. Fuzzy output values are scaled in the defuzzification interface and converted to real numbers. Each fuzzy output set of membership values for each rule used is determined in the output universal set. Then, one of the defuzzification methods is used on the logical combination set created by these sets and the defuzzification process is done by finding a single output value.

The most common defuzzification methods are “*center of sums (COS) method*”, “*center of gravity (COG)*” / “*centroid of area (COA) method*”, “*bisector of area (BOA) method*”, “*weighted average method*” and “*maxima methods*”.

COS is the fastest defuzzification method that uses algebraic sums instead of combining two output fuzzy sets. In COS, the overlapping areas are added twice. The defuzzified value is calculated by Eq. 8.

$$z^* = \frac{\int_z z \sum_{k=1}^{n} \mu_{C_k} z \, dz}{\int_z \sum_{k=1}^{n} \mu_{C_k} z \, dz} \tag{8}$$

COG or *COA* is one of the most widely used defuzzification techniques and evaluated by the center of gravity. The MF values and the scalar weights of fuzzy outputs of rules are multiplied, and their sums are calculated. The obtained value is divided by the sum of the MF values as in Eq.9.

$$\mathrm{COG} = \frac{\int_a^b \mu_A(x)x\,dx}{\int_a^b \mu_A(x)\,dx} \tag{9}$$

BOA method divides under the curve into two equal areas. The value that partitions both sides into equal regions is the output value by Eq. 10.

$$x^* = \int_{\alpha}^{x^*} \mu_A(x)\,dx = \int_{x^*}^{\beta} \mu_A(x)\,dx \tag{10}$$

In the weighted average method, all fuzzy values and MDs are used for defuzzification. The output value is calculated by weighting each MF with its maximum value by Eq. 11. Maxima method takes the element with the highest MD in the fuzzy set as the defuzzified value.

$$x^* = \frac{\sum \mu(x)x}{\sum \mu(x)} \tag{11}$$

2.3. Fuzzy Rule Generation Algorithms and Neural Network

2.3.1. Wang-Mendel Method

Wang and Mendel [3] proposed a common method to produce linguistic fuzzy rules from numerical data by designing a control system to replace the human controller. We need a human controller and input-output pairs to design the system. The human controller expresses his/her knowledge as linguistic "IF-THEN" rules. The rules are not sufficient for creating the system because the human controller will lose information when transmitting it. In addition, the input and output pairs cannot cover all possible conditions of the system and not sufficient to create a successful control system. Therefore, Wang and Mendel produced rules from numerical data by combining both information.

WM method consists of five steps that are "dividing input-output spaces into fuzzy regions, generation of the fuzzy rule, assigning degree to each rule, creating a combined fuzzy rule base and determining a mapping based on the integrated rule base". Let us assume a_1, a_2 are inputs, and b is the output. Domain intervals of inputs, and outputs are $[a_1^-, a_1^+]$, $[a_2^-, a_2^+]$ and $[b^-, b^+]$. The domain interval of a variable represents the interval between minimum and maximum value that a variable can take. Each domain interval is divided into "*2N + 1*" regions (value of *N* differentiates with respect to a variable and region length can be equal or not) and assigned a fuzzy MF. The shape of the MFs are triangular.

In the second step, the degrees of input and output variables are determined and assigned a region that has a maximum degree. As shown in Figure 2.7, $a_2^{(1)}$ belongs to region *S1*, similarly $b^{(1)}$ belongs to region *CE*. By assigning regions to the variables,

- Rule 1: IF a_1 is X1 and b_2 is S1 THEN c is CE
- Rule 2: IF a_1 is X1 and b_2 is CE THEN c is X1

rules are determined from input-output pairs. Inputs of all rules are connected with logical '*AND*' operator.

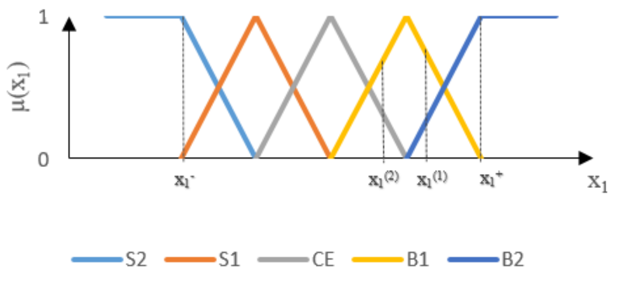
Since each data pair generates a rule, many rules are obtained. The resulting rules have conflicting rules with the same "IF" (premise) and different "THEN" (consequent) part. In the third step, the rule with the maximum degree is selected from the conflict group by assigning degrees to each rule to solve this conflict. Thus, the number of rules is reduced. The degree of a rule is calculated by Eq. 12.

$$\mathrm{D(rule)} = \mu_A(x_1)\mu_B(x_2)\mu_C(y) \tag{12}$$

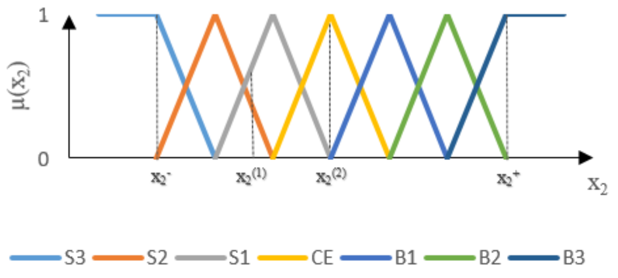
For instance, Rule 1 a has degree;

$$\begin{aligned} \text{D(rule)} &= \mu_{B1}(x_1)\mu_{S1}(x_2)\mu_{CE}(y) \\ &= 0.5 * 0.6 * 0.7 \\ &= 0.21 \end{aligned} \tag{13}$$

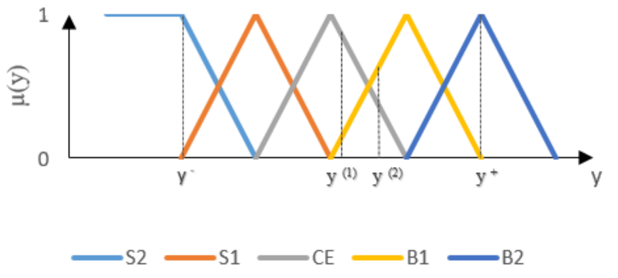
In the presence of an expert, the data pairs are examined and assigned a degree according to their usefulness. Therefore, the human controller determines the degree of the rules.



(a)



(b)



(c)

Figure 2.7 Fuzzy regions of input and output variables [3]

The rules derived from numerical data form a combined fuzzy rule basis. The rule with the highest degree is used when there are several rules in the resulting rule base. Thus, both numerical data and linguistic information are combined in the fourth step.

As a final step, Wang and Mendel determined the output for given x_1, x_2 inputs and began with identifying the output degree ($\mu^i_{O^i}$), using defuzzification by Eq. 14, where O^i and I^i_j represent the output and input regions of rule *i* respectively.

$$\mu^i_{O^i} = \mu_{I^i_1}(x_1)\mu_{I^i_2}(x_2) \tag{14}$$

They used "centroid defuzzification" to determine the output by Eq. 15, where $\bar{y}^i$ represents the center of a fuzzy region O^i and "K" is the number of rules of the obtained fuzzy rule base.

$$\mathrm{y} = \frac{\sum_{i=1}^{K} \mu^i_{O^i}\bar{y}^i}{\sum_{i=1}^{K} \mu^i_{O^i}} \tag{15}$$

WM is a method that very simple and easy to construct. Unlike the neural approach, the training process does not require time and effort. In addition, this approach can be extended to multi-input and multi-output problems. WM is independent of the number of inputs and outputs from the 1st step to the 4th step. For the last step, the only difference in Eq. 15 is the output degree should be ($\mu_{O^i_j}$) instead of ($\mu^i_{O^i}$), j denotes the *j*th output component.

If we consider the five-step procedure as a block, the input-output samples and the linguistic rules are inputs, and mapping from input to output space is the output to the block. This approach is a general method that can adapt when examples and linguistic rules change.

2.3.2. Interval-Valued Fuzzy Reasoning Method with Tuning and Rule Selection

The selection of MFs is one of the essential factors determining the success of fuzzy rule-based classification systems. This is a complex problem resulting from the uncertainty associated with definitions of linguistic terms. “Interval-valued fuzzy sets” (IVFSs) are used to define fuzzy terms with system uncertainties. IVFS defines an interval rather than a single number as the MD of each element. The IVTURS [4] is a new “interval-valued fuzzy reasoning method” by extending the classical fuzzy reasoning method.

This method called as IVTURS-FARC [4] since IVTURS is combined with FARC-HD [21] (“fuzzy association rule-based classification model for high dimensional problems”) algorithm which is used for rule learning process. IVTURS is composed of three steps:

1) “Interval-valued fuzzy rule-based classification system” (IV-FRBCS) is generated by the FARC-HD method, and linguistic labels are determined by IVFSs. Then interval-valued restricted equivalence functions (IV-REFs), that measure the equivalence between intervals, are generated for each variable.
2) The fuzzy reasoning method is extended on IVFSs.
3) Optimization is applied for genetic tuning and the rule selection process.

In order to learn the FRBCS, association rules are extracted by using a search tree, and rules, are selected using subgroup discovery that reduces the computational cost. After creating the initial FRBCS, linguistic labels are determined by generating the upper and lower bounds of each MF. Then, initial IV-REFs are generated for each variable.

A rule learning algorithm is used to generate a knowledge base. Fuzzy rule form is represented in Eq. 16, where an algorithm uses a set of patterns $(x_1, \dots., x_n)$, and R_j is the jth rule, A_{jn} is the fuzzy set that represents a linguistic term.

$$\text{Rule } \mathrm{R_j} = If\ x_1\ is\ A_{j1}\ \ldots and\ x_n\ is\ A_{jn}\ then\ Class = C_j\ with\ RW_j \tag{16}$$

Certainty factor is applied to compute rule weights, as shown in Eq. 17, where $\mu_{A_J}(x_p)$ is the matching degree of pattern x_p.

$$\mathrm{RW}_j = \mathrm{CF}_j = \frac{\sum_{x_p \in Class\ C_j} \mu_{A_J}(x_p)}{\sum_{p=1}^{P} \mu_{A_J}(x_p)} \tag{17}$$

IV-REFs compute how similar the interval MDs of each variable is to the ideal MD ([1,1]). Although initial IV-REF (a) is an interval MD, it varies according to the values of the parameters (b, c).

a) $\emptyset_1(x) = x\ and\ \emptyset_2(x) = x$:
   IV-REF ([0.6, 0.7], [1, 1]) = [0.6, 0.7]
b) $\emptyset_1(x) = x^2\ and\ \emptyset_2(x) = x$:
   IV-REF ([0.6, 0.7], [1, 1]) = [0.77, 0.84]
c) $\emptyset_1(x) = x^{0.5}\ and\ \emptyset_2(x) = x$:
   IV-REF ([0.6, 0.7], [1, 1]) = [0.36, 0.49]

In the final step, the genetic algorithm is used to set the values of the parameters used in the construction of the IV-REFs and to perform the selection process between obtained rules.

2.3.3. Rule Production Approach of Defuzzification-Free Hierarchical Fuzzy System

Increasing the number of input parameters makes the single fuzzy system inapplicable since it increases the computational cost and complexity of the rules. In this case, hierarchical fuzzy systems are highly preferred. Mutlu et al. [5] proposed a new strategy called as DF-HFS that preserves the fuzziness of data while transferring it between layers.

Data may not always be consistent, complete, or even exist. In the DF-HFS algorithm, Mutlu et al. [5] proposed a straightforward approach as a solution to these problems. The rule scheme of the DF-HFS algorithm does not need expert knowledge to produce rules. The DF-HFS generates the complete ruleset by using tiny information of data. In our study, we used only the rule production scheme of DF-HFS.

First, MFs are determined as whose boundaries of fuzzy sets equally partitioned. Each input-output variable has five fuzzy sets. The complete ruleset is determined as $R = [R^{ant}\ R^{cons}]$ where "R^{ant}" is rule antecedent, and "R^{cons}" is rule consequent. R^{ant} is $[n^m$ by $m]$ matrix that stores all variations of fuzzy sets for each input variable, where "n" is the number of fuzzy sets and "m" is the number of input variables. R^{cons} is a $[n^m$ by $1]$ matrix that contains the consequent fuzzy set of each rule. Score adjustment of these MFs determined by expert opinion in case of no data available or according to the rate of predetermined fuzzy sets. The rating factor (RF) is specified for each fuzzy set (f) of each input variable (i) and represented as $RT_{i,f}$. All RFs are normalized by dividing summation of the maximum RFs of each input variable, as shown in Eq. 18. The consequent of rule r is calculated by the summation of normalized RFs $RT_{(i,f)_r}$ for i^{th} input, and f^{th} fuzzy set as shown in Eq. 19.

$$\overline{RT_{i,f}} = \frac{RT_{i,f}}{\sum_{i=1}^{m} \max_{f=1 \to n} RT_{i,f}} \tag{18}$$

$$R_r^{cons} = \sum_{i=1}^{m} \overline{RT_{(i,f)_r}} \tag{19}$$

Finally, the fuzzy set of the rule consequents is determined by equation Eq. 19. The R^{cons} determines the rule consequent belong to which fuzzy set. For example, it is 0,82. The output space is divided into 5 equal intervals as K_1= [0.8–1.0], K_2= [0.6–0.8), K_3= [0.4–0.6), K_4= [0.2–0.4), K_5= [0–0.2). Then the output belongs to 1st fuzzy set since it is in interval K_1= [0.8–1.0].

2.3.4. Artificial Neural Network

ANN is inspired by the human brain through the mathematical modeling of the learning process. ANNs are composed of many neurons, and these neurons work simultaneously to perform complex functions. In other words, complex functions are created by the simultaneous operation of many neurons.

The task of the ANN is to produce an output in response to the information given as an input set [36]. A 3-layer (or layered) feedforward neural network model consists of input, hidden, and output layers are shown in Figure 2.8.

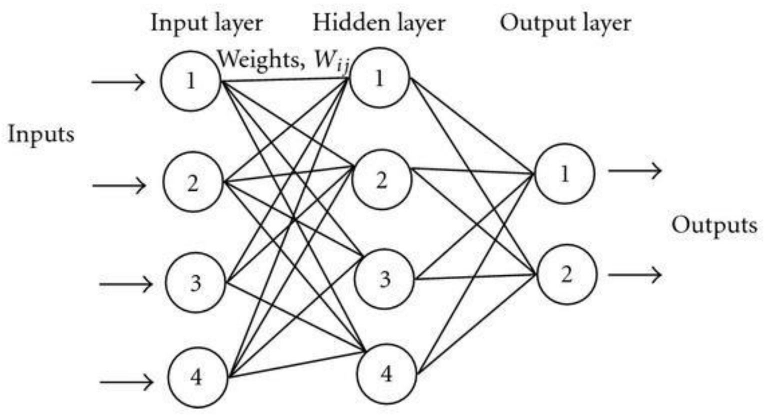


Figure 2.8 The ANN modeling

There are different types of ANN architectures, but the most popular ones are “feed-forward neural network” and “back-propagation neural network”.

- **“Feed-Forward Neural Network”**: There is a one-way flow of information, as shown in Figure 2.9. The information on the input layer is transmitted to the hidden layer. The output value is obtained by processing from the hidden layer and transmitting it to the output layer and then processing in the output layer.

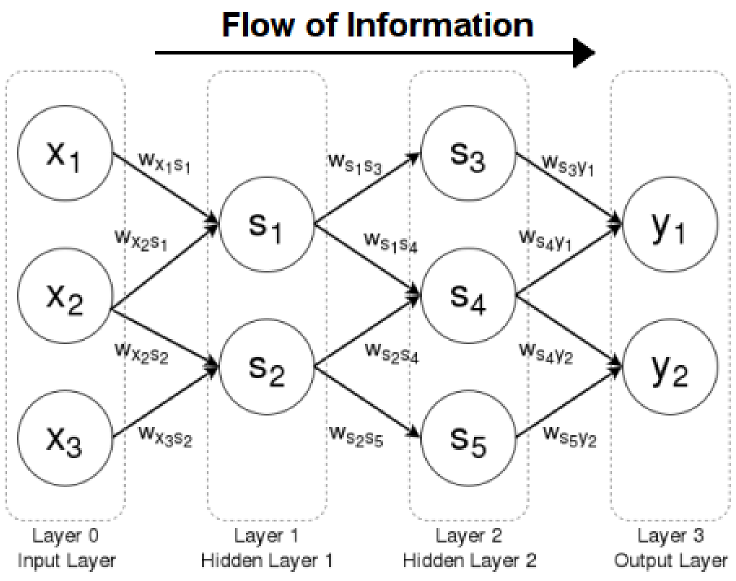


Figure 2.9 Feed-forward Neural Networks

- **"Feedback Neural Network"**: It is a network structure that the outputs in the output and hidden layers are fed back to the input layer or to the previous hidden layers, as shown in Figure 2.10. Thus, the inputs are transmitted both in the forward direction and in the reverse direction. This type of neural network has a dynamic memory that the current output reflects the current and previous inputs. They are especially suitable for prediction applications.

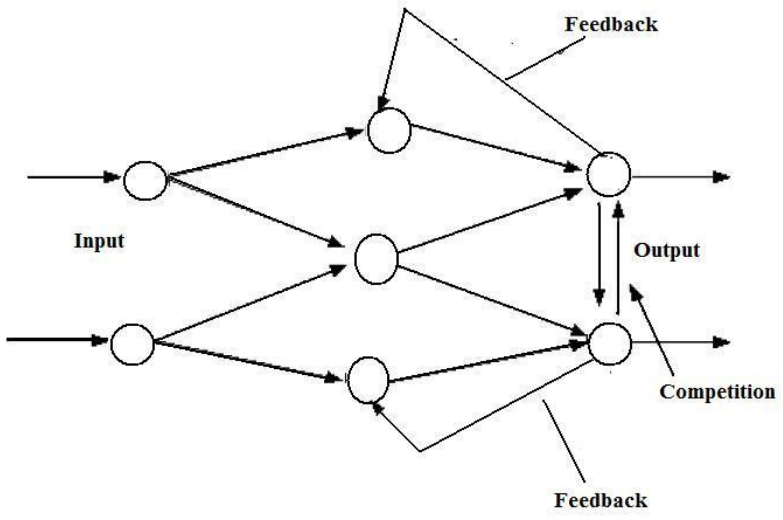


Figure 2.10 Feedback Neural Networks

In the process of developing ANNs, data is divided into two; one part trains the network and called as the training set and the other part measures the performance of the network other than the training data and is called the test set. ANN training is the process of optimizing the weights of the network. There are two conditions for changing the weights of the network. These are changing weights between the hidden layer and the output layer and changing the weights between the hidden layer and the input layer. The aim is to find the weight values that will produce the correct outputs for the training set. The process is repeated until the calculated error rate falls to the predefined threshold. As a result of the training process, the error calculated on the ANN is expected to decrease to an acceptable error rate. However, the decrease of the mean error does not always indicate that the ANN has reached generalization. The inputs in the test set are given to the ANN model, and the output value is compared with the desired output. The real purpose of the ANN reaches the generalization for the input-output samples.

2.4. Software Fault Prediction

The increasing size and complexity of the software make it difficult to offer high-quality software at a low cost. SFP plays a crucial role in improving software quality and reducing maintenance efforts. Early detection of failures enables early correction of these failures, cost reduction, and maintainable software delivery. In the literature, various software metrics are available. These metrics can be used to predict faulty modules in the early stages of the software development life-cycle.

SFP classifies modules or classes as faulty or non-faulty. Software metrics from similar projects or previous versions of the project can be used to generate a model that predicts fault. This model can be applied to the new project to find faulty modules or classes. Software practitioners can then find and correct existing faults in the fault-prone areas of the software at an early stage of software development. Therefore, higher quality and easy-to-maintain software can be produced at the given time and budget.

SFP metrics are classified as “method-level”, “class-level”, “component-level”, “file-level”, “process-level” and “quantitative-level” [37]. Halstead and McCabe method-level metrics are the most popular metrics for SFP problem. These metrics are included in public datasets that available in the PROMISE repository. There are several datasets for SFP problem classified as “public”, “private”, “partial” and “unknown”. The PROMISE and the NASA MDP repositories are commonly preferred to have public datasets.

In order to identify faults, there are several approaches, which we can group them as “statistical methods”, “machine learning-based methods”, “statistical methods combined with an expert opinion”, “statistical methods combined with machine learning-based methods”. Naïve Bayes, Random Forests, and J48 [37] machine learning algorithms are mostly used for SFP problems. According to reviews of Catal and Diri [37], distribution of methods is presented in Figure 2.11.

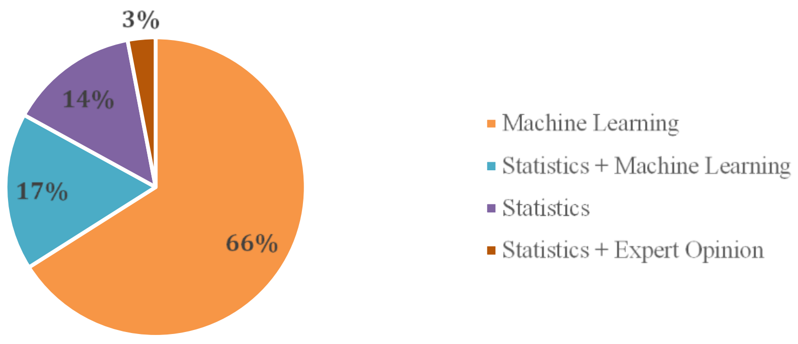


Figure 2.11 Distribution of Methods

3. RELATED WORKS

3.1. Fuzzy Rule Generation Algorithms

Fuzzy logic utilizes human knowledge, processing them as rule bases, and resulting in the conclusion of each rule base corresponding to a specific mathematical function. It requires expert knowledge to determine the rules accurately. Expert opinions may not be consistent when data complexity increases. In addition, to find an expert is usually a costly and time-consuming task. In order to overcome these challenges, several automatic fuzzy rule generation approaches have been proposed to generate human-understandable rules in recent decades [19,38–41]. Michalski and Chilausky [42] have done one of the first studies of creating IF-THEN rules from given data inductively instead of expert knowledge. They realized that inductively derived decision rules were performed better than the rules derived from the expert knowledge. Inductively derived rules diagnosed the 100% of diseases, while expert derived rules diagnosed 96.2%.

Fuzzy rules have an essential role in determining the robustness of fuzzy systems. There are various methods to extract rule base, such as genetic algorithms [6, 21], association rule mining [21,43,44], fuzzy clustering [19,45], linear solutions [3,12,46] and evolutionary rule learning methods [4,16,47].

Neural networks are widely preferred while dealing with prediction and classification problems. However, the main drawback is the rules produced by neuro-fuzzy systems are not human-understandable, that makes these systems lack interpretability [39,48]. Therefore, neuro-fuzzy systems were excluded from this study.

The selection of a ruleset is one of the main problems when the number of rules increases. Genetic algorithms [17] are used to maximize the number of accurately classified patterns and reduce the redundant fuzzy if-then rules. The ruleset represented as a string and treated individually. The rules are classified as dummy rules, relevant and irrelevant rules. The genetic algorithm assigns a degree to each rule, “0” to dummy rules, “1” to relevant, and “-1” to irrelevant rules randomly. After *n* (pre-defined stopping value) generations, the final ruleset is determined. The determined ruleset was tested with Iris data and obtained a 94.67% classification rate.

In high-dimensional problems, one of the main challenges is knowledge extraction. The fuzzy association rule-based classification method [21] is proposed to reduce the computational cost. Association rules extract interesting relations between values in a database. Rule interestingness measured with two factors, which are support and confidence. Different algorithms [44] are offered for the generation of association rules.

In fuzzy systems, the complexity of the model is essential in terms of the number of fuzzy rules. Fuzzy clustering methods [19,45] establishes a balance between model accuracy and complexity. During the clustering process, redundant and less critical clusters are removed with the orthogonal least-squares method [45]. “Fuzzy c-means clustering algorithm” [19] clusters the samples effectively by reducing the scale of samples. Then, the resulting samples are used to generate fuzzy rules.

For complex systems, the human controller is the most essential part of decision-making that expresses his knowledge by defining linguistic IF-THEN rules. In most of the cases, linguistic rules which represented by the human controller cannot describe all system. Numerical input-output data pairs and linguistic rules are together is sufficient for a successful design. In this circumstance, Wang and Mendel [3] propose a linear approach that generates fuzzy rules from numerical data by learning from examples. This method combines both numerical data and linguistic information then suggests a fuzzy rule base.

As an evolutionary rule learning method, IVTURS-FARC [4] is a trendy method based on the “interval-valued fuzzy reasoning method”. Computational cost is higher according to linear approaches because of tuning matching degrees and rule selection steps.

3.2. Transfer Learning

Data science analyzes massive amounts of data and extracts knowledge from them. Data-driven methods cannot produce meaningful information when data is corrupted or in cases of data deficiency. On the other hand, people adapt to changing environmental conditions by transferring information from one situation to another in the learning process [15]. They produce a solution based on previous information instead of learning from scratch when they encounter new tasks.

Transfer learning discusses how to exploit the pre-acquired knowledge in another project related to the original project to improve the accuracy of learning [49]. Several studies have been conducted in the field of transfer learning, and related studies can be divided into various categories based on problem identification: “multitasking learning” [50], “domain adaptation” [51], “cross-domain adaptation” [52] and “heterogeneous learning” [53]. The main research flows include “the transferring knowledge of samples [54,55], transferring knowledge of feature representations [56], transferring knowledge of model parameters [57] and transferring relational knowledge” [58] when categorized by the transferring approach. Existing studies are classified from the application point of view as classification [59,60], unsupervised learning [61,62] and regression [63,64].

Do and Ng [65] proposed an algorithm that finds a good parameter function to solve novel classifications tasks. The learning algorithm learned from 450 classification problems and outperforms the well-known naïve Bayes, softmax regression, and SVM algorithms. The proposed algorithm performed 0.459 accuracy while softmax achieved 0.376.

Quattoni et al. developed a new method by leveraging previous knowledge from the unlabeled data to learn a new visual category. They collected 10382 images that belong to 108 topics from “Reuters news web-site” and built a representation of image prototype. Experiments to estimate whether the image belongs to a news topic indicate that when only a few examples are available to train a target subject, the use of information learned from other subjects can significantly improve performance.

Zuo et al. [15] proposed a Takagi-Sugeno fuzzy regression model for transferring knowledge. The rules were constructed using source data and modified to reuse them in target data. In the experiments, they used syntactic and real-world datasets. They used the same feature dimensionality for both the source domain and target domain. The experiment results showed that the introduced fuzzy regression model effectively estimates the values of the target domain.

Shell and Coupland [14] combined the transfer learning and fuzzy logic in a novel framework. This framework learns from labeled data generates fuzzy rules, sets, and transferable FIS. It transfers the knowledge to unlabeled data using FIS. The framework was compared with K-Nearest Neighbour and Support Vector Machine and outperformed with a lower Root Mean Squared Error in 54.5% and 67.9%, respectively.

3.3. Software Fault Prediction

People work directly or indirectly in software-related jobs. This situation increases the dependence on software day by day. The software is used in several applications, so its quality and reliability are vital. Failures in software can lead to loss of labor, money, and even life. That is why software quality and reliability is a significant area of research.

The software is developed by the human beings and measured in man-month. This measure is the nature of the fuzzy approach. Software quality is measured by determining the defects/faults in the software. Several studies have been proposed in the field of SFP and estimation [6,8,66,67,9,22–27,37]. Catal and Diri [9,37] provided a comprehensive review of several SFP that is categorized as metrics, methods, and datasets.

In order to deliver reliable software, fault prediction is desirable in each phase of the software development life cycle [26,66]. The fault estimate made in the testing phase of the development can be quite costly, as it will cause changes in all previous phases. Therefore, SFP was performed in requirement analyses, design, coding, and testing phases. FIS was employed to develop the model.

Better design decisions can be made using object-oriented metrics for SFP, and more appropriate design alternatives can be easily selected. Ertürk and Sezer [23] were predicted the faults by using Mamdani FIS with class-level metrics that correspond to object-oriented metrics. In addition, they [6] made another study using both class-level and method-level metrics. FISs with method-level metrics were as successful as class-level metrics.

Marian et al. [25] proposed a new approach for SFP problem using fuzzy decision trees. They used JEdit and Ant datasets and reduced the software metric size to three as WMC, CBO, RFC. They compared their study with Weka – Decision Tree, Orange – Decision Tree, Logistic Regression, Bayesian networks and it outperforms all of them with the accuracy of 0.735 and 0.707 for JEdit and Ant datasets respectively.

Kaur et al. [68] investigated whether early lifecycle metrics (requirement metrics) and late lifecycle metrics (code metrics) can be used to identify faulty modules. They used JM1, PC1, and CM1 NASA software projects to test faultiness by using the "K-means clustering algorithm". The experiment results show that the best prediction model is identified by the combination of requirement and code metrics.

Menzies et al. [69] performed several data mining algorithms for SFP problem on NASA datasets by using method-level metrics. They showed that the Naïve Bayes algorithm outperformed the J48 algorithm, but changing dataset characteristics changed the best-performed algorithm. SFP problem should be examined with several datasets by using several learning algorithms.

4. COMPARISONS OF FUZZY RULE GENERATION ALGORITHMS AND ANN FOR SOFTWARE FAULT PREDICTION PROBLEM

FIS is a rule-based system that uses knowledge and experience to solve complex problems based on fuzzy IF-THEN rules. These rules play an essential role in determining the performance of fuzzy systems and the capabilities of the fuzzy system. The main approach of creating a fuzzy system is to extract fuzzy rules from samples. These rules are derived from several algorithms, such as heuristic and genetic algorithms, neural networks, fuzzy clustering, etc. However, these methods require continuous repetitive learning and are difficult to implement due to their complex mechanisms. In our study, three automatic rule generation algorithms are implemented, namely “Wang-Mendel”, “Interval-Valued Fuzzy Reasoning Method with Tuning and Rule Selection”, rule production approach of “Defuzzification-free Hierarchical Fuzzy System”. As seen in the study [38], these algorithms have been preferred in rule generation since they have an outstanding performance, and they all have some individual differences.

The WM [3] method can effectively produce fuzzy rules from sample data. The method uses a lookup table to extract the rules for each fuzzy subspace by dividing the input into equal fuzzy sets. The WM method is simple and practical because it does not require repeated learning steps.

A better fuzzy rule base produced from sample data can improve the success of the fuzzy system. As a result, the WM method provides the rules based entirely on the input-output relationships, so the dependence on samples is quite high. The fuzzy rule base extracted by the WM method will also not be consistent if the sample set is corrupted. Accordingly, the accuracy of the generated fuzzy system with the wrong set of rules will be poor. At this point, each method may have drawbacks in terms of applicability or accuracy.

Unlike the WM method, DF-HFS [5] combines some of the features that are gathered from expert knowledge and data. This allows DF-HFS to be applicable when there is not enough data or expert opinion, allowing existing information to replace the missing.

Because, unlike data-driven machine learning algorithms, this approach underlines that the fuzzy systems are also robust in the absence of data.

The rule production scheme of DF-HFS is based on the score values of each fuzzy input variable that is a measure of the effect of each fuzzy set. Unlike the original method, we determined three fuzzy sets for input and two fuzzy sets for the output variable. The score adjustment can be made by the expert in the absence of any data or can be calculated automatically, taking into account the regression between the input and output variable. Finally, after the scores have been set, the rule matrix is created to cover each possible input and output variations. In addition, since the DF-HFS method generates a complete set of rules, the system can be interpreted more easily, considering all metrics.

The IVTURS-FARC [4] algorithm optimizes and tunes the fuzzy rules based on MF definition with an evolutionary approach. This approach provides a fuzzy rule base, covering substantially the corresponding data. Nevertheless, this method is more expensive than linear approaches since they require extensive calculations during the learning and optimization processes. Since the algorithm may not include all metrics in the rule base it provides, the interpretation of the system will be difficult. On the other hand, the algorithm achieves very high classification success as it combines the equivalence setting with the rule selection to reduce the complexity of the system.

On the other hand, the ANN method was examined to compare the success of the classification of fuzzy systems with machine learning algorithms, which are preferred frequently. ANN algorithm is completely dependent on data and hides the learning mechanism in the inner layers of the network. Therefore, accurate results may not be obtained when the model is learned from a project and tested with a different project. Considering that “loc” is used as an input in a model established with ANN, only the relation of “loc” metric with the data format and output parameter is considered in this model. However, if the prediction model is FIS, as mentioned above, the classification process can be performed without being dependent on the data.

The rule base was created by learning from the source dataset with different rule generation algorithms for each project. A transferable FIS was created with the obtained rules. The generated FISs were tested with target projects, and the knowledge learned from the source project was transferred to the target project, and fault prediction was made, as shown in Figure 4.1.

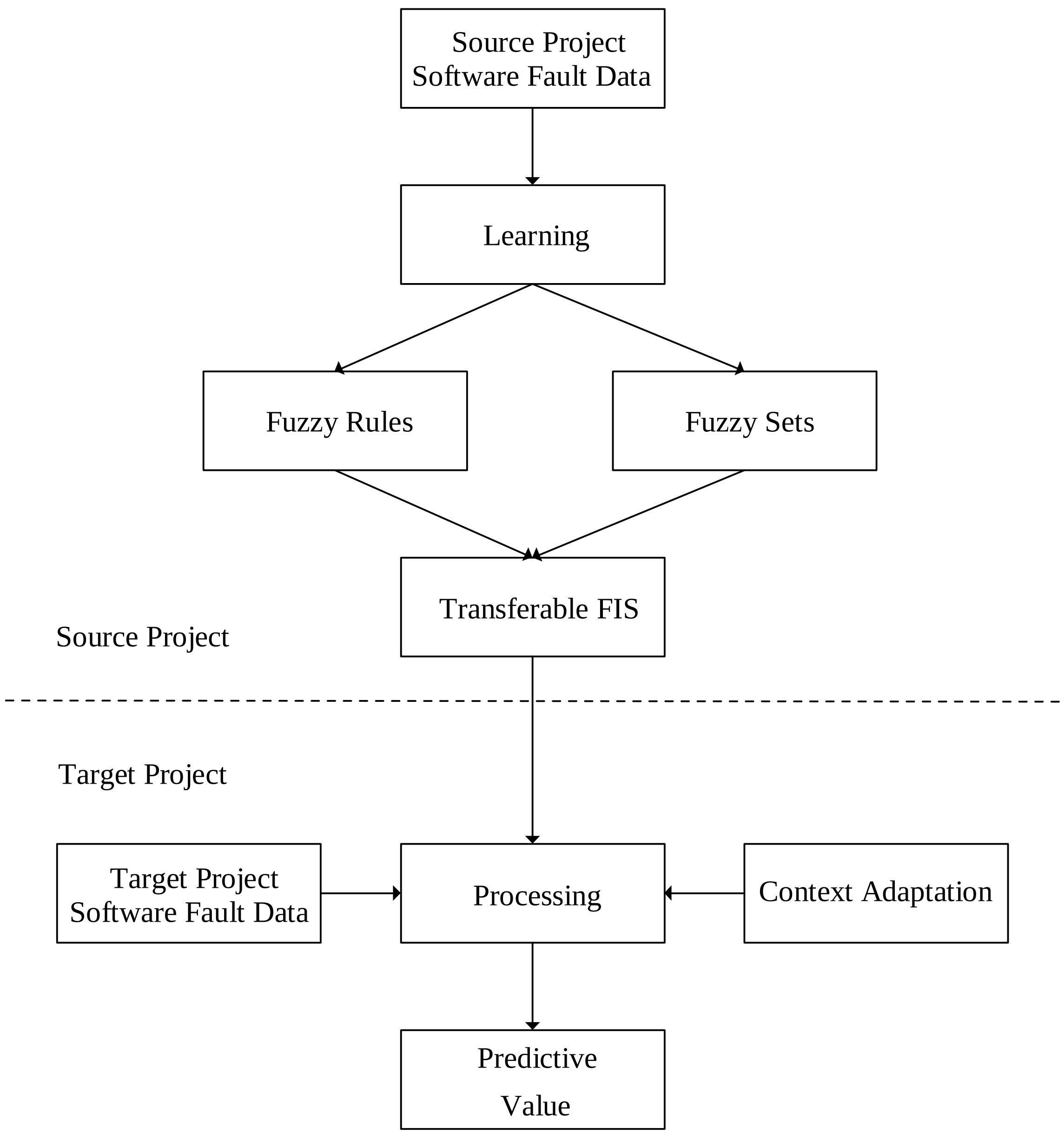


Figure 4.1 Overview of the Fuzzy Transfer Learning framework

4.1. Implementation Details

WM and the rule generation scheme of the DF-HFS algorithm were implemented with Java programming language on IntelliJ IDEA 2017.2.5. The IVTURS-FARC is included algorithm in open-source Java software tool “Knowledge Extraction based on Evolutionary Learning” (KEEL) [70]. KEEL provides a simple frontend that allows designing experiments with different datasets.

For all rule generation algorithms, triangular MFs are used, and fuzzy sets are divided equally. The generated rules were included in the FISs since the fuzzy rules define how the FIS can decide to classify an input and control an output. FIS properties are as follows:

- **Type of inference model**: Mamdani
- **Type of rules**: AND
- **Type of MF**: Triangular
- **Fuzzy reasoning method**: COA
- **Implication method**: Minimum
- **Aggregation method**: Maximum
- **Environment**: MATLAB 9.5.0 (R2018b)
- **Toolbox**: Fuzzy logic

FISs were created by using the Fuzzy Logic Toolbox provided by MATLAB. Here are several editors and viewers provided by the toolbox:

- **FIS editor**: represents the general information about the FIS
- **MF editor**: allows displaying and editing the MFs associated with the input and output variables
- **Rule editor**: allows viewing and editing the fuzzy rules
- **Rule viewer**: views detailed behavior of a FIS

The main parameters and implementation details of utilized ANN are as follows:

- **Type of architecture:** Feed-forward neural network
- **Train size**: 80% of data

- **Test size**: 20% of data
- **Number of epocs**: 100
- **Number of hidden layers**: 3
- **Number of neurons**: 48 in the first layer, 24 in the second layer, 1 in the third layer
- **Activation function**: Sigmoid
- **Training algorithm**: Gradient descent
- **Evaluation**: 5-fold cross-validation
- **Environment**: Python/Keras library [71]

5. EXPERIMENTS AND RESULTS

In this section, WM, IVTURS-FARC, rule production scheme of DF-HFS and ANN algorithms are compared based on their classification performance. FISs were created for different rulesets derived from these algorithms on several datasets.

5.1. Datasets

CM1, JM1, KC1, KC2, and PC1 publicly available SFP datasets obtained from PROMISE Software Engineering Repository [30]. These datasets are preferred since they are the most popular datasets in the SFP research area [6,9,28,29,38,72,73] and all contain the same metrics. Each dataset consists of 21 features that have 5 different lines of code measure, 4 McCabe metrics [20], 11 Halstead measures [16], 1 branch-count, and 1 class variable, which defines the instance whether defected or not. The specifications of the datasets are shown in Table 5.1.

Table 5.1 Dataset Specifications

Data	Language	# Modules	# Faulty Modules	Faulty Modules %
CM1	C	498	49	9.83 %
JM1	C	10885	8779	80.65 %
KC1	C++	2109	326	15.45 %
KC2	C++	522	105	20.5 %
PC1	C	1109	1032	93.05 %

During the data-preprocessing step, we transformed the class variables into a binary attribute, to denote whether defected or not. The value “1” represents defective entities, and the “0” is for non-defective ones. The number of inputs to be studied was reduced to nine that consist of McCabe and Halstead metrics. Although McCabe and Halstead’s metrics were introduced in the 1970s for fault prediction activities, they are still the most common metrics. According to Catal and Diri [37], predictive fault-prone modules may have faults during system testing or field testing. ***McCabe*** metrics used in the fault prediction are explained as follows:

- ***Line of code (loc)***: It measures the physical count of LOC.
- ***Cyclomatic complexity (v(g))***: It measures the number of linearly independent paths in a program module. It is one of the most common and accepted software metrics, independent of programming language and writing. The v(g) is the number of conditional branches. Increased v(g) leads to increasing processing paths in the function, and makes it hard to understand the program.
- ***Essential complexity (ev(g))***: It used to determine the amount of unstructured code and the quality of a program construct. Program is qualified as unstructured if an ev(g) is greater than "1" and far away from the well-structured programming.
- ***Design complexity (iv(g))***: It is proportional to the number of calls the method makes to other methods. The complexity of the code that occurs when the not-called code is eliminated.

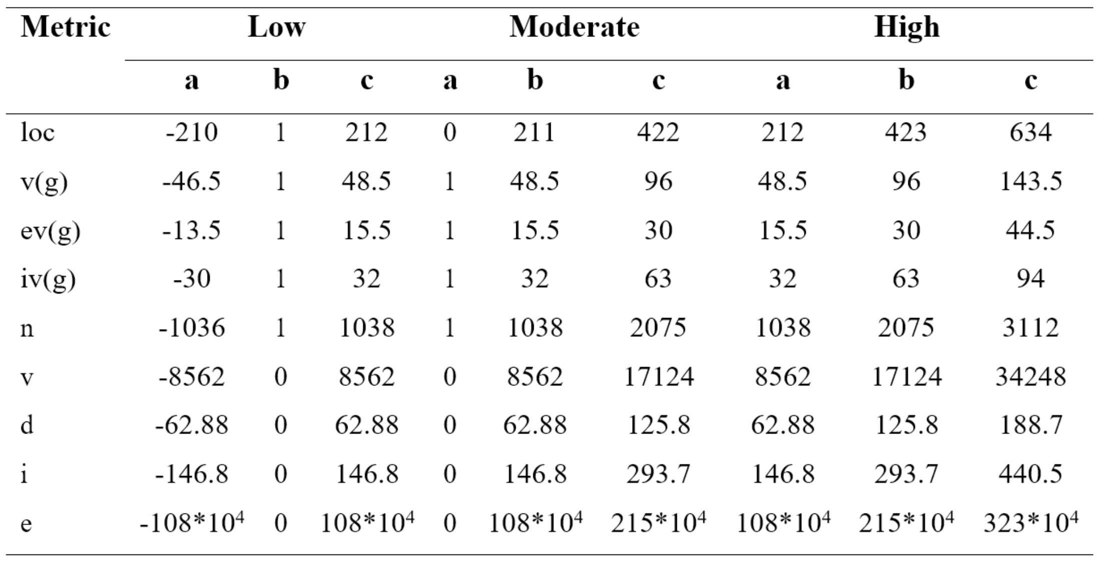
Halstead metrics are explained as follows:

- ***Volume (v)***: It represents the size of the space required to store the program in bits that proportional to the program size. This parameter depends on the specific algorithm implementation.
- ***Difficulty (d)***: It measures the difficulty level of a program. It is a metric that shows faults due to the overuse of the unique operands. Therefore, programming applications such as overuse of operands or wrong usage of higher-level control structures will be prone to increase difficulty.
- ***Intelligence (i)***: It provides a measurement of a particular algorithm complexity regardless of the implemented programming language.
- ***Length (n)***: It measures the total number of operators and operands that occurred in the module.
- ***Effort (e)***: It measures the mental activity amount required to convert the current algorithm into implementation in the specified program language.

5.2. Case Study: Software Fault Prediction

In order to perform our experiments, three fuzzy rule generation algorithms, namely WM, IVTURS-FARC, rule production scheme of DF-HFS methods, and ANN algorithm were implemented. First, we produced the rules in order to apply to FISs by using the aforementioned automatic rule generation algorithms. The experiments were performed separately for Halstead metrics and McCabe metrics for each of the five datasets. In the rule generation process, three and two fuzzy sets are used that are *Low (L), Moderate (M), High (H)* for corresponding input linguistic variable and *Low-Fault-Proneness (L), High-Fault-Proneness (H)* for corresponding output linguistic variables. Input and output fuzzy sets were divided into equal intervals. MF values were represented as seen in Table 5.2, Table 5.3, Table 5.4, Table 5.5, and Table 5.6 for each dataset.

Table 5.2 The MF values of Cm1 dataset



Metric	Low			Moderate			High		
	a	b	c	a	b	c	a	b	c
loc	-210	1	212	0	211	422	212	423	634
v(g)	-46.5	1	48.5	1	48.5	96	48.5	96	143.5
ev(g)	-13.5	1	15.5	1	15.5	30	15.5	30	44.5
iv(g)	-30	1	32	1	32	63	32	63	94
n	-1036	1	1038	1	1038	2075	1038	2075	3112
v	-8562	0	8562	0	8562	17124	8562	17124	34248
d	-62.88	0	62.88	0	62.88	125.8	62.88	125.8	188.7
i	-146.8	0	146.8	0	146.8	293.7	146.8	293.7	440.5
e	$-108*10^4$	0	$108*10^4$	0	$108*10^4$	$215*10^4$	$108*10^4$	$215*10^4$	$323*10^4$

Table 5.3 The MF values of Jm1 dataset

Metric	Low			Moderate			High		
	a	b	c	a	b	c	a	b	c
loc	-1720	1	1722	1	1722	3442	1722	3442	5163
v(g)	-233.5	1	235.5	1	235.5	470	235.5	470	704.5
ev(g)	-81	1	83	1	83	165	83	165	247
iv(g)	-199.5	1	201.5	1	201.5	402	201.5	402	602.5
n	-4221	0	4221	0	4221	8441	4221	8441	12662
v	$-404*10^5$	0	$404*10^5$	0	$404*10^5$	$808*10^5$	$404*10^5$	$808*10^5$	$121*10^6$
d	-209.1	0	209.1	0	209.1	418.2	209.1	418.2	627.3
i	-284.9	0	284.9	0	284.9	569.8	284.9	569.8	854.7
e	$-155*10^8$	0	$155*10^8$	0	$155*10^8$	$311*10^9$	$155*10^8$	$311*10^8$	$466*10^8$

Table 5.4 The MF values of Kc1 dataset

Metric	Low			Moderate			High		
	a	b	c	a	b	c	a	b	c
loc	-142.5	1	144.5	1	144.5	288	144.5	288	431.5
v(g)	-21	1	23	1	23	45	23	45	67
ev(g)	-11.5	1	13.5	1	13.5	26	13.5	26	38.5
iv(g)	-21	1	23	1	23	45	23	45	67
n	-553	0	553	0	553	1106	553	1106	1659
v	-3959	0	3959	0	3959	7919	3959	7919	$1188*10^4$
d	-26.88	0	26.88	0	26.88	53.75	26.88	53.75	80.63
i	-96.53	0	96.53	0	96.53	193.1	96.53	193.1	289.6
e	$-162*10^6$	0	$162*10^6$	0	$162*10^6$	$325*10^6$	$162*10^6$	$325*10^6$	$487*10^6$

Table 5.5 The MF values of Kc2 dataset

Metric	Low			Moderate			High		
	a	b	c	a	b	c	a	b	c
loc	-636	1	638	1	638	1275	638	1275	1912
v(g)	-88.5	1	90.5	1	90.5	180	90.5	180	270
ev(g)	-61	1	63	1	63	125	63	125	187
iv(g)	-70	1	72	1	72	143	72	143	214
n	-1990	1	1992	1	1992	3982	1992	3982	5973
v	$-169*10^5$	0	$169*10^5$	0	$169*10^5$	$338*10^5$	$169*10^5$	$338*10^5$	$507*10^5$
d	-51.77	0	51.77	0	51.77	103.5	51.77	103.5	155.3
i	-207.5	0	207.5	0	207.5	415.1	207.5	415.1	622.6
e	$-107*10^7$	0	$107*10^7$	0	$107*10^7$	$215*10^7$	$107*10^7$	$215*10^7$	$322*10^7$

Table 5.6 The MF values of Pc1 dataset

Metric	Low			Moderate			High		
	a	b	c	a	b	c	a	b	c
loc	-301	0	301	0	301	602	301	602	903
v(g)	-66.5	1	68.5	1	68.5	136	68.5	136	203.5
ev(g)	-60	1	62	1	62	123	62	123	184
iv(g)	-60	1	62	1	62	123	62	123	184
n	-1391	1	1393	1	1393	2785	1393	2785	4177
v	$-129*10^5$	0	$129*10^5$	0	$129*10^5$	$259*10^5$	$129*10^5$	$259*10^5$	$389*10^5$
d	-135.3	0	135.3	0	135.3	270.7	135.3	270.7	406
i	-299.2	0	299.2	0	299.2	598.3	299.2	598.3	897.5
e	$-214*10^6$	0	$214*10^6$	0	$214*10^6$	$428*10^6$	$214*10^6$	$428*10^6$	$642*10^7$

Rule weights are determined as “1” for all algorithms to make a fair comparison. Generated rule numbers are shown in Table 5.7 and Table 5.8, that varies depending on the algorithm.

Table 5.7 Generated rule numbers derived from McCabe metrics by Wang-Mendel, IVTURS-FARC, DF-HFS algorithms

Ruleset	**Wang-Mendel**	**IVTURS-FARC**	**DF-HFS**
Cm1's Rules	16	7	81
Jm1's Rules	13	4	81
Kc1's Rules	18	6	81
Kc2's Rules	4	4	81
Pc1's Rules	9	7	81
Frequent Rules	5	7	81

Table 5.8 Generated rule numbers derived from Halstead metrics by Wang-Mendel, IVTURS-FARC, DF-HFS algorithms

Ruleset	**Wang-Mendel**	**IVTURS-FARC**	**DF-HFS**
Cm1's Rules	15	4	243
Jm1's Rules	17	3	243
Kc1's Rules	14	8	243
Kc2's Rules	8	6	243
Pc1's Rules	11	6	243
Frequent Rules	10	6	243

In addition to generating rules for each dataset, a customized ruleset called as ***frequent rules*** (FR) was created. The rules generated from each dataset were combined, and the repeated rules in the resulting ruleset form the frequent ruleset. For example, regarding the WM algorithm, the rules derived from Cm1, Jm1, Kc1, Kc2, and Pc1 with McCabe metrics were combined (16+13+18+4+9) and obtained a total of 60 rules. The repetitive 5 rules in the combined ruleset, which consists of 60 rules, form the ***FR***. FR set of WM and IVTURS-FARC algorithms are represented in Table 5.9 and Table 5.10, respectively.

Although the number of rules generated from different datasets differs in WM and IVTURS-FARC algorithms, it is constant for the DF-HFS algorithm since DF-HFS's rule generation algorithm produces a complete ruleset that covers all input-output interactions. Therefore, the ruleset is the same for each dataset. The number of rules was calculated as 3^4 for 3 fuzzy sets and 4 McCabe metrics and 3^5 for 5 Halstead metrics. In the DF-HFS algorithm, the consequent part of the rules is updated according to the distribution of rules in the entire ruleset, whether the rules are defective or not.

Table 5.9 The Frequent Ruleset of WM algorithm

Metrics	Rules
McCabe	If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L then (faultiness is L)
	If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is L then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is M then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M then (faultiness is L)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is L then (faultiness is L)
Halstead	If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L)
	If (n is M) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
	If (n is M) and (v is M) and (d is L) and (i is H) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
	If (n is M) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is L)
	If (n is M) and (v is M) and (d is L) and (i is M) and (e is L) then (faultiness is H)
	If (n is L) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is M) and (v is M) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is L) and (v is L) and (d is L) and (i is H) and (e is L) then (faultiness is L)

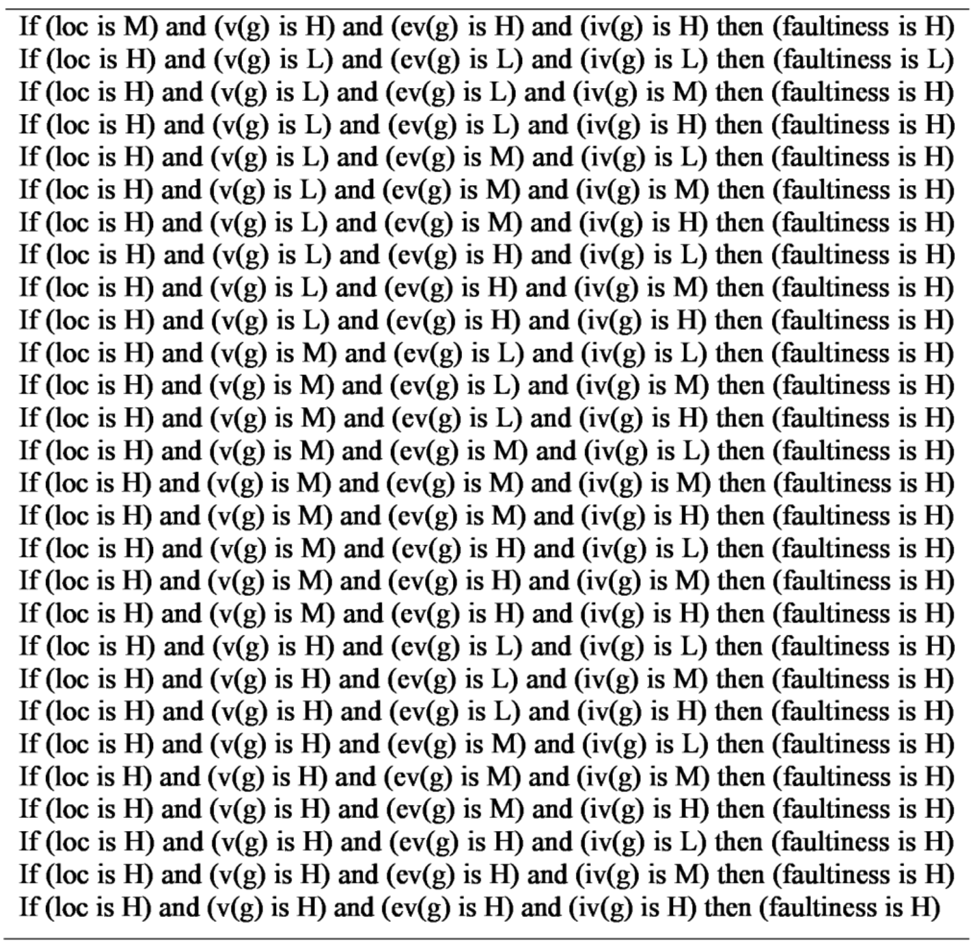
Table 5.10 The Frequent Ruleset of IVTURS-FARC algorithm

Metrics	Rules
McCabe	If (iv(g) is M) then (faultiness is H)
	If (iv(g) is L) then (faultiness is L)
	If (ev(g) is M) then (faultiness is H)
	If (ev(g) is L) then (faultiness is L)
	If (v(g) is L) then (faultiness is L)
	If (loc is M) then (faultiness is H)
	If (loc is L) then (faultiness is L)
Halstead	If (i is M) then (faultiness is L)
	If (i is L) then (faultiness is L)
	If (v is L) then (faultiness is L)
	If (d is L) then (faultiness is L)
	If (n is M) and (v is M) then (faultiness is H)
	If (n is L) then (faultiness is L)

The complete ruleset is represented in Table 5.11. All rules were different because they contained each possible state for this algorithm. Thus, the entire rules created the FR set (Table 5.11).

Table 5.11 The ruleset derived from DF-HFS algorithm with McCabe metrics

Rules
If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is M) then (faultiness is L)
If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is L) and (ev(g) is H) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is L) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is L) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is M) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is H) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is L) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
If (loc is M) and (v(g) is L) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is L) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is M) and (v(g) is H) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)



If (loc is M) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is H) and (v(g) is L) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is L) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is M) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is L) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is H)

Automatic rule generation algorithms have differences in certain aspects. The WM and DF-HFS algorithms produce complete rulesets, but the rules that IVTURS-FARC generates may not include all input variables. The rules generated by each algorithm for both McCabe and Halstead metrics. The rulesets that are generated from the WM algorithm are shown in Table 5.12 and Table 5.13. In addition, the IVTURS-FARC algorithm generates rules, as shown in Table 5.14 and Table 5.15.

Table 5.12 The ruleset derived from WM algorithm with McCabe metrics

Dataset	Rules
Cm1	If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is L)
	If (loc is H) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is L)
	If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
	If (loc is M) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
	If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
	If (loc is M) and (v(g) is L) and (ev(g) is M) and (iv(g) is M) then (faultiness is L)
	If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is L)
	If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
	If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is L)
	If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
	If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
	If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
	If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
Jm1	If (loc is M) and (v(g) is H) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
	If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
	If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
	If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
	If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
	If (loc is L) and (v(g) is H) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
	If (loc is H) and (v(g) is H) and (ev(g) is M) and (iv(g) is H) then (faultiness is H)
	If (loc is L) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is L)
	If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
	If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is M) then (faultiness is L)
Kc1	If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
	If (loc is H) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
	If (loc is H) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is H)
	If (loc is H) and (v(g) is H) and (ev(g) is L) and (iv(g) is H) then (faultiness is H)
	If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is H)
	If (loc is H) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
	If (loc is M) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is H)
	If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
	If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is L)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
	If (loc is H) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is L)
	If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
	If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
	If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is L)

Kc2	If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is H) If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L) If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is H) If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is H)
Pc1	If (loc is H) and (v(g) is H) and (ev(g) is M) and (iv(g) is M) then (faultiness is H) If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is H) If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is H) If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is L) If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is L) If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is L) If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L) If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L) If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)

Table 5.13 The ruleset derived from WM algorithm with Halstead metrics

Dataset	Rules
Cm1	If (n is H) and (v is H) and (d is H) and (i is M) and (e is M) then (faultiness is L) If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L) If (n is M) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L) If (n is M) and (v is M) and (d is M) and (i is M) and (e is L) then (faultiness is L) If (n is M) and (v is M) and (d is L) and (i is H) and (e is L) then (faultiness is L) If (n is M) and (v is L) and (d is H) and (i is L) and (e is L) then (faultiness is L) If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L) If (n is H) and (v is H) and (d is H) and (i is M) and (e is H) then (faultiness is L) If (n is M) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is L) If (n is L) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is L) If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is L) If (n is M) and (v is M) and (d is L) and (i is M) and (e is L) then (faultiness is H) If (n is M) and (v is M) and (d is M) and (i is M) and (e is M) then (faultiness is H) If (n is M) and (v is M) and (d is M) and (i is H) and (e is M) then (faultiness is H) If (n is M) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is H)
Jm1	If (n is M) and (v is M) and (d is M) and (i is L) and (e is L) then (faultiness is H) If (n is M) and (v is M) and (d is H) and (i is L) and (e is M) then (faultiness is H) If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is H) If (n is L) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is H) If (n is M) and (v is M) and (d is M) and (i is M) and (e is L) then (faultiness is H) If (n is M) and (v is M) and (d is L) and (i is H) and (e is L) then (faultiness is H) If (n is M) and (v is M) and (d is M) and (i is M) and (e is M) then (faultiness is H) If (n is M) and (v is M) and (d is M) and (i is L) and (e is M) then (faultiness is H) If (n is L) and (v is M) and (d is M) and (i is L) and (e is L) then (faultiness is H) If (n is H) and (v is H) and (d is H) and (i is M) and (e is H) then (faultiness is H) If (n is M) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L) If (n is L) and (v is L) and (d is L) and (i is H) and (e is L) then (faultiness is L) If (n is M) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L) If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L) If (n is M) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is L) If (n is M) and (v is L) and (d is H) and (i is L) and (e is M) then (faultiness is L) If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is L)

Kc1	If (n is M) and (v is L) and (d is H) and (i is L) and (e is L) then (faultiness is H)
	If (n is L) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is M) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is M) and (v is L) and (d is L) and (i is H) and (e is L) then (faultiness is H)
	If (n is M) and (v is M) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is M) and (v is M) and (d is H) and (i is M) and (e is M) then (faultiness is H)
	If (n is M) and (v is M) and (d is M) and (i is M) and (e is M) then (faultiness is L)
	If (n is L) and (v is L) and (d is L) and (i is H) and (e is L) then (faultiness is L)
	If (n is H) and (v is H) and (d is H) and (i is H) and (e is H) then (faultiness is L)
	If (n is M) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is H) and (i is L) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
Kc2	If (n is M) and (v is L) and (d is H) and (i is L) and (e is M) then (faultiness is H)
	If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L)
	If (n is H) and (v is H) and (d is H) and (i is H) and (e is H) then (faultiness is H)
	If (n is M) and (v is M) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is M) and (v is M) and (d is M) and (i is M) and (e is M) then (faultiness is H)
	If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is H)
	If (n is L) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is H)
	If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is H)
Pc1	If (n is H) and (v is H) and (d is M) and (i is M) and (e is M) then (faultiness is H)
	If (n is M) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is H)
	If (n is M) and (v is M) and (d is M) and (i is L) and (e is L) then (faultiness is L)
	If (n is M) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
	If (n is H) and (v is M) and (d is H) and (i is L) and (e is H) then (faultiness is L)
	If (n is M) and (v is M) and (d is L) and (i is M) and (e is L) then (faultiness is L)
	If (n is M) and (v is M) and (d is L) and (i is L) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L)
	If (n is M) and (v is M) and (d is L) and (i is H) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is L)
	If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)

Table 5.14 The ruleset derived from IVTURS-FARC algorithm for McCabe metrics

Dataset	Rules
Cm1	If (iv(g) is M) then (faultiness is H)
	If (iv(g) is L) then (faultiness is L)
	If (ev(g) is M) then (faultiness is H)
	If (ev(g) is L) then (faultiness is L)
	If (v(g) is L) then (faultiness is L)
	If (loc is M) then (faultiness is H)
	If (loc is L) then (faultiness is L)
Jm1	If (iv(g) is L) then (faultiness is L)
	If (ev(g) is L) then (faultiness is L)
	If (v(g) is L) then (faultiness is L)
	If (loc is L) then (faultiness is L)

Kc1	If (loc is M) and (v(g) is M) and (iv(g) is M) then (faultiness is H) If (loc is H) and (iv(g) is M) then (faultiness is H) If (v(g) is M) and (iv(g) is M) then (faultiness is H) If (ev(g) is M) then (faultiness is L) If (ev(g) is L) then (faultiness is L) If (loc is M) then (faultiness is L)
Kc2	If (v(g) is M) then (faultiness is H) If (loc is M) then (faultiness is H) If (loc is L) then (faultiness is L) If (loc is M) and (v(g) is L) and (iv(g) is L) then (faultiness is H)
Pc1	If (iv(g) is M) then (faultiness is H) If (iv(g) is L) then (faultiness is L) If (ev(g) is L) then (faultiness is L) If (ev(g) is M) then (faultiness is H) If (v(g) is L) then (faultiness is L) If (loc is M) then (faultiness is H) If (loc is L) then (faultiness is L)

Table 5.15 The ruleset derived from IVTURS-FARC algorithm for Halstead metrics

Dataset	Rules
Cm1	If (i is M) then (faultiness is L) If (i is L) then (faultiness is L) If (v is L) then (faultiness is L) If (i is H) then (faultiness is H)
Jm1	If (i is L) then (faultiness is L) If (d is M) then (faultiness is H) If (d is L) then (faultiness is L)
Kc1	If (d is H) then (faultiness is H) If (d is M) then (faultiness is H) If (n is L) and (v is L) then (faultiness is H) If (n is L) and (v is L) and (e is L) then (faultiness is H) If (d is M) and (e is L) then (faultiness is H) If (e is L) then (faultiness is L) If (v is L) and (i is L) then (faultiness is L) If (d is L) and (i is L) then (faultiness is L)
Kc2	If (i is L) then (faultiness is L) If (n is L) then (faultiness is L) If (v is M) then (faultiness is H) If (n is M) and (v is M) then (faultiness is H) If (d is M) and (i is M) then (faultiness is H) If (d is M) and (i is M) and (e is L) then (faultiness is H)
Pc1	If (i is M) then (faultiness is L) If (i is L) then (faultiness is L) If (v is M) then (faultiness is L) If (v is L) then (faultiness is L) If (d is L) then (faultiness is L) If (n is L) then (faultiness is L)

In the rule generation process, we also combine all datasets and generate the rules with each algorithm. The generated ruleset is called as a ***union*** ruleset. The union rulesets are shown in Table 5.16, Table 5.17, and Table 5.18. The union ruleset is the same as the ruleset in Table 5.11 for DF-HFS algorithm since it does not show any differences for different datasets.

Table 5.16 The union ruleset derived from WM algorithm for McCabe metrics

Rule set
If (loc is L) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is M) and (v(g) is L) and (ev(g) is L) and (iv(g) is L) then (faultiness is M)
If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is M)
If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is M) then (faultiness is L)
If (loc is M) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is H) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is L)
If (loc is M) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is L) and (ev(g) is M) and (iv(g) is L) then (faultiness is L)
If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is L) then (faultiness is M)
If (loc is M) and (v(g) is M) and (ev(g) is H) and (iv(g) is M) then (faultiness is M)
If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is M)
If (loc is L) and (v(g) is M) and (ev(g) is M) and (iv(g) is M) then (faultiness is L)
If (loc is L) and (v(g) is M) and (ev(g) is L) and (iv(g) is L) then (faultiness is L)
If (loc is H) and (v(g) is H) and (ev(g) is H) and (iv(g) is H) then (faultiness is M)

Table 5.17 The union ruleset derived from WM algorithm for Halstead metrics

Rule set
If (n is L) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is M)
If (n is M) and (v is M) and (d is M) and (i is L) and (e is L) then (faultiness is M)
If (n is M) and (v is M) and (d is H) and (i is L) and (e is M) then (faultiness is M)
If (n is L) and (v is L) and (d is M) and (i is M) and (e is L) then (faultiness is M)
If (n is M) and (v is M) and (d is M) and (i is M) and (e is L) then (faultiness is M)
If (n is M) and (v is M) and (d is L) and (i is H) and (e is L) then (faultiness is M)
If (n is M) and (v is M) and (d is M) and (i is M) and (e is M) then (faultiness is M)
If (n is M) and (v is M) and (d is M) and (i is L) and (e is M) then (faultiness is M)
If (n is L) and (v is M) and (d is M) and (i is L) and (e is L) then (faultiness is M)
If (n is H) and (v is H) and (d is H) and (i is M) and (e is H) then (faultiness is M)
If (n is L) and (v is L) and (d is L) and (i is H) and (e is L) then (faultiness is L)
If (n is M) and (v is L) and (d is L) and (i is M) and (e is L) then (faultiness is L)
If (n is L) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
If (n is M) and (v is L) and (d is M) and (i is L) and (e is L) then (faultiness is L)
If (n is M) and (v is L) and (d is H) and (i is L) and (e is M) then (faultiness is L)
If (n is L) and (v is L) and (d is L) and (i is L) and (e is L) then (faultiness is L)
If (n is M) and (v is M) and (d is L) and (i is M) and (e is L) then (faultiness is M)

Table 5.18 The union ruleset derived from IVTURS algorithm

Metric	Rules
McCabe	If (ev(g) is L) then (faultiness is L) If (v(g) is L) then (faultiness is L) If (loc is L) then (faultiness is L)
Halstead	If (v is L) then (faultiness is L) If (d is M) then (faultiness is L) If (d is L) then (faultiness is L) If (i is L) then (faultiness is L)

In order to evaluate the performance of FISs, "Area Under Receiver Operating Characteristic Curve" (ROC-AUC) was used. ROC-AUC is one of the most frequently used performance evaluation criteria for classification problems at all classification thresholds. ROC curve plots two parameters, which are true positive rate (TPR as seen Eq.20) and false positive rate (FPR as seen in Eq.21). The AUC is the area under the ROC curve, as shown in Figure 5.1.

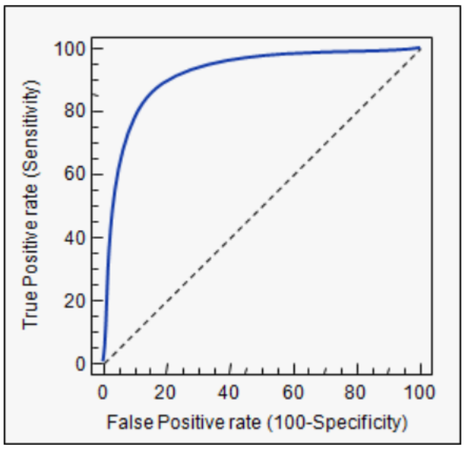


Figure 5.1 Area under the ROC Curve (AUC)

$$\text{True positive rate (TPR)} = \frac{TP}{TP + FP} \tag{20}$$

$$\text{False positive rate (FPR)} = \frac{FP}{FP + TN} \tag{21}$$

In order to make experiments on differently modeled FISs, seven FISs were generated for each software project in different scenarios. Each experiment was performed separately for the McCabe and Halstead metrics.

5.2.1. Ruleset of the current (target) project

In the first scenario, the FIS in a certain project was generated by using the data of the current project to determine the parameters of MFs and create the rule base. In order to explain in more detail, five FISs were produced for this scenario. The FIS generated with Cm1 data was tested with Cm1 rules, and the other four projects (Jm1, Kc1, Kc2, and Pc1) were tested with their own rulesets. Experiment results are shown in Table 5.19 and Table 5.20. Here, scenarios for each project are represented by presenting the name of the current project and the source project, that the rules were generated from, concatenated by ***"W"*** symbol. For this experiment, each case is denoted as CM1 W CM1, JM1 W JM1, KC1 W KC1, KC2 W KC2, and PC1 W PC1.

Table 5.19 Experiment results of projects tested with the ruleset of the current project regarding McCabe metrics

Ruleset / **Dataset**	**Wang-Mendel**	**IVTURS-FARC**	**DF-HFS**	**ANN**
CM1 W CM1	0,7003	0,7239	**0,7247**	0,676
JM1 W JM1	0,6888	0,6116	**0,6909**	0,60
KC1 W KC1	0,7897	0,7212	**0,7903**	0,793
KC2 W KC2	0,8416	**0,8422**	0,8416	0,823
PC1 W PC1	**0,6957**	0,6876	0,6704	0,614

Table 5.20 Experiment results of projects tested with the ruleset of the current project regarding Halstead metrics

Rule set / Dataset	Wang-Mendel	IVTURS-FARC	DF-HFS	ANN
CM1 W CM1	0,7236	0,7113	0,7056	**0,824**
JM1 W JM1	0,6202	0,6087	0,6192	**0,733**
KC1 W KC1	0,7865	**0,7951**	0,7932	0,783
KC2 W KC2	0,8249	**0,8446**	0,8254	0,815
PC1 W PC1	**0,6849**	0,6742	0,6709	0,614

5.2.2. Ruleset of the source project

In the second to fifth scenarios, MFs and FISs are determined by using the data of the current project the same as in the first scenario. On the other hand, the rule base of their FISs was generated from other data of projects instead of its own data. In this case, four separate FISs were created using the ruleset of the other four projects while each project was tested. For example, regarding the Cm1 project, the Cm1 dataset was used to determine MFs and FIS. The datasets of Jm1, Kc1, Kc2, and Pc1 were utilized for the rule generation. AUC values for these experiments are shown in Table 5.21, and Table 5.22 denoted as CM1 W JM1, CM1 W KC1, CM1 W KC2, and CM1 W PC1.

Table 5.21 Experiment results of projects tested with the ruleset of the source project regarding McCabe metrics

Ruleset / Dataset	Wang-Mendel	IVTURS-FARC	DF-HFS	ANN
CM1 W JM1	0,7202	0,5863	**0,7247**	**0,485**
CM1 W KC1	0,7183	0,6973	**0,7247**	0,279
CM1 W KC2	0,7118	**0,7514**	**0,7247**	0,298
CM1 W PC1	**0,7415**	0,7239	**0,7247**	0,483
JM1 W CM1	0,6817	0,6897	**0,6909**	0,485
JM1 W KC1	0,6889	0,6603	**0,6909**	0,463
JM1 W KC2	0,6882	**0,7087**	**0,6909**	0,484
JM1 W PC1	**0,7010**	0,6897	**0,6909**	**0,487**

KC1 W CM1	0,7891	**0,7914**	**0,7903**	0,689
KC1 W JM1	**0,7920**	0,6111	**0,7903**	0,608
KC1 W KC2	**0,7907**	0,7904	**0,7903**	**0,749**
KC1 W PC1	0,7908	**0,7914**	**0,7903**	0,632
KC2 W CM1	0,8438	**0,8441**	**0,8416**	**0,651**
KC2 W JM1	**0,8417**	0,7073	**0,8416**	0,581
KC2 W KC1	0,8398	0,8184	**0,8416**	0,440
KC2 W PC1	0,8430	**0,8441**	**0,8416**	0,486
PC1 W CM1	0,6660	**0,6885**	**0,6704**	0,653
PC1 W JM1	**0,6797**	0,5828	**0,6704**	0,621
PC1 W KC1	0,6648	0,6292	**0,6704**	**0,759**
PC1 W KC2	0,6591	0,6872	**0,6704**	0,711

Table 5.22 Experiment results of projects tested with the ruleset of the source project regarding Halstead metrics

Ruleset / **Dataset**	**Wang-Mendel**	**IVTURS-FARC**	**DF-HFS**	**ANN**
CM1 W JM1	0,7274	0,7064	**0,7056**	0,292
CM1 W KC1	**0,7283**	0,7345	**0,7056**	0,212
CM1 W KC2	0,7180	**0,7491**	**0,7056**	0,164
CM1 W PC1	0,6998	0,7360	**0,7056**	**0,294**
JM1 W CM1	0,6210	0,6254	**0,6192**	0,7071
JM1 W KC1	**0,6252**	**0,6277**	**0,6192**	0,783
JM1 W KC2	0,6198	0,6203	**0,6192**	**0,847**
JM1 W PC1	0,6208	0,6271	**0,6192**	0,719
KC1 W CM1	**0,7916**	0,7737	**0,7932**	0,704
KC1 W JM1	0,7847	0,7829	**0,7932**	0,690
KC1 W KC2	0,7499	0,7799	**0,7932**	**0,835**
KC1 W PC1	0,7840	**0,7899**	**0,7932**	0,660
KC2 W CM1	0,8287	0,8269	**0,8254**	0,772
KC2 W JM1	0,8355	0,8311	**0,8254**	0,697
KC2 W KC1	**0,8371**	**0,8414**	**0,8254**	0,753
KC2 W PC1	0,8301	0,8413	**0,8254**	**0,834**
PC1 W CM1	0,6867	0,6852	**0,6709**	0,653
PC1 W JM1	0,7126	0,6323	**0,6709**	0,621
PC1 W KC1	0,6888	0,6914	**0,6709**	**0,759**
PC1 W KC2	**0,7134**	**0,6986**	**0,6709**	0,711

5.2.3. Ruleset of the frequent rules

In the sixth scenario, the rules produced in the first five scenarios were combined, and the rules that were included in more than one ruleset constituted the frequent ruleset. As in other scenarios, MFs and FISs were created using the data of the current project. Five different FISs were created since the FR set was tested with each project. Experiments are represented as CM1 W FR, JM1 W FR, KC1 W FR, KC2 W FR, PC1 W FR in Table 5.23, and Table 5.24.

Table 5.23 Experiment results of projects tested with the frequent ruleset regarding McCabe metrics

Ruleset / Dataset	Wang-Mendel	IVTURS-FARC	DF-HFS	ANN
CM1 W FR	**0,7383**	0,7221	0,7247	NA
JM1 W FR	0,7010	0,6899	**0,6909**	NA
KC1 W FR	**0,7905**	0,7897	0,7903	NA
KC2 W FR	0,8431	**0,8445**	0,8416	NA
PC1 W FR	**0,6944**	0,6876	0,6704	NA

Table 5.24 Experiment results of projects tested with the frequent ruleset regarding Halstead metrics

Ruleset / Dataset	Wang-Mendel	IVTURS-FARC	DF-HFS	ANN
CM1 W FR	0,7295	**0,7379**	0,7056	NA
JM1 W FR	0,6252	**0,6276**	0,6192	NA
KC1 W FR	0,7856	**0,7933**	0,7932	NA
KC2 W FR	0,8395	**0,8410**	0,8254	NA
PC1 W FR	0,6890	**0,6929**	0,6709	NA

5.2.4. Ruleset of the union data

In the last experiment on FIS, a new dataset is created called *"**union**"* by combining the data of five projects. For the union dataset, new MFs and FIS are determined. The rules produced from this dataset are tested with FIS. Experiment results are shown in Table 5.25.

Table 5.25 Experiment results of projects tested with the union of datasets

Metric	Wang-Mendel	IVTURS-FARC	DF-HFS	ANN
McCabe	0,7076	0,6136	0,7092	**0,737**
Halstead	0,6483	**0,6526**	0,6458	0,641

5.2.5. Portability of the rules

In order to investigate the portability of the rulesets obtained from rule generation algorithms, we assign weights between 0-6 to the rulesets. The most accurate ruleset gets "6", and the least accurate one gets "1" for the corresponding project. If the AUC value of the ruleset tested with a project is less than the AUC value received when tested with the project's own ruleset, it takes the "0". For example, since the CM1 project receives the highest AUC value when it is tested with the ruleset of PC1 project, the weight of the PC1 ruleset is given as "6", as seen in Table 5.26. On the other hand, when JM1 project is tested with CM1 and KC2 rulesets, it gets "0" since AUC values of CM1 (0,6817) and KC2 (0,6882) is lower with respect to tested with its own data (0,6888). Total weights of rules are shown in Table 5.26, Table 5.27, Table 5.28, Table 5.29, and Table 5.30.

Table 5.26 Total rule weights obtained from the Wang-Mendel method regarding McCabe metrics

Ruleset	CM1	JM1	KC1	KC2	PC1	Total Rule Weights
CM1's Rule	1	0	0	6	5	12
JM1's Rule	4	4	6	3	0	17
KC1's Rule	3	5	2	0	0	10
KC2's Rule	2	0	4	2	0	8
PC1's Rule	6	6	5	4	0	**21**
Frequent Rules	5	6	3	5	0	19

Table 5.27 Total rule weights obtained from the Wang-Mendel method regarding Halstead metrics

Rule Set	**CM1**	**JM1**	**KC1**	**KC2**	**PC1**	**Total Rule Weights**
CM1's Rule	3	5	6	2	2	18
JM1's Rule	4	3	0	4	5	16
KC1's Rule	5	6	5	5	3	**24**
KC2's Rule	0	0	0	1	6	7
PC1's Rule	0	4	0	3	1	8
Frequent Rules	6	6	0	6	4	22

Table 5.28 Total rule weights obtained from the IVTURS-FARC method regarding McCabe metrics

Ruleset	**CM1**	**JM1**	**KC1**	**KC2**	**PC1**	**Total Rule Weights**
CM1's Rule	5	4	6	5	6	**26**
JM1's Rule	0	2	0	0	5	7
KC1's Rule	0	3	3	0	0	6
KC2's Rule	6	6	5	3	0	20
PC1's Rule	5	4	6	5	0	20
Frequent Rules	0	5	4	6	5	20

Table 5.29 Total rule weights obtained from the IVTURS-FARC method regarding Halstead metrics

Ruleset	**CM1**	**JM1**	**KC1**	**KC2**	**PC1**	**Total Rule Weights**
CM1's Rule	2	3	0	0	3	8
JM1's Rule	0	1	0	0	0	1
KC1's Rule	3	6	6	0	4	19
KC2's Rule	6	2	0	6	6	**20**
PC1's Rule	4	4	0	0	2	10
Frequent Rules	5	5	0	0	5	15

Table 5.30 Total rule weights obtained from the DF-HFS method regarding McCabe/Halstead metrics

Ruleset	**CM1**	**JM1**	**KC1**	**KC2**	**PC1**	**Total Rule Weights**
CM1's Rule	6	6	6	6	6	**30**
JM1's Rule	6	6	6	6	6	**30**
KC1's Rule	6	6	6	6	6	**30**
KC2's Rule	6	6	6	6	6	**30**
PC1's Rule	6	6	6	6	6	**30**
Frequent Rules	6	6	6	6	6	**30**

5.2.6. Experiments with the ANN algorithm

Regarding ANN, experiments were performed differently from fuzzy rule generation algorithms. Five different scenarios were defined for ANN. In the first scenario, ANN was trained with 80% of data and tested with 20% of data for each project as a regular machine learning evaluation process. These experiments are shown as CM1 W CM1, JM1 W JM1, KC1 W KC1, KC2 W KC2, PC1 W PC1 in Table 5.31 and Table 5.32 since the train and test data belong to the same project.

In the other four experiments, ANN was trained with 100% data of the current project and tested with 100% data of source project. For the CM1 project, ANN was trained with the CM1 project and tested with four other projects. Four different experiments were performed for each project called as CM1 W JM1, CM1 W KC1, CM1 W KC2, and CM1 W PC1 in Table 5.31 and Table 5.32.

Table 5.31 AUC values of FISs and ANN regarding McCabe metrics

Ruleset / **Dataset**	**Wang-Mendel**	**IVTURS-FARC**	**DF-HFS**	**ANN**
CM1 W CM1	0,7003	0,7239	**0,7247**	**0,676**
CM1 W JM1	0,7202	0,5863	**0,7247**	0,485
CM1 W KC1	0,7183	0,6973	**0,7247**	0,279
CM1 W KC2	0,7118	**0,7514**	**0,7247**	0,298
CM1 W PC1	**0,7415**	0,7239	**0,7247**	0,483
CM1 W FR	0,7383	0,7221	**0,7247**	NA
JM1 W JM1	0,6888	0,6116	**0,6909**	**0,60**
JM1 W CM1	0,6817	0,6897	**0,6909**	0,485
JM1 W KC1	0,6889	0,6603	**0,6909**	0,463
JM1 W KC2	0,6882	**0,7087**	**0,6909**	0,484
JM1 W PC1	**0,7010**	0,6897	**0,6909**	0,487
JM1 W FR	**0,7010**	0,6899	**0,6909**	NA
KC1 W KC1	0,7897	0,7212	**0,7903**	**0,793**
KC1 W CM1	0,7891	**0,7914**	**0,7903**	0,689
KC1 W JM1	**0,7920**	0,6111	**0,7903**	0,608
KC1 W KC2	0,7907	0,7904	**0,7903**	0,749
KC1 W PC1	0,7908	**0,7914**	**0,7903**	0,632
KC1 W FR	0,7905	0,7897	**0,7903**	NA

KC2 W KC2	0,8416	0,8422	**0,8416**	**0,823**
KC2 W CM1	**0,8438**	0,8441	**0,8416**	0,651
KC2 W JM1	0,8417	0,7073	**0,8416**	0,581
KC2 W KC1	0,8398	0,8184	**0,8416**	0,440
KC2 W PC1	0,8430	0,8441	**0,8416**	0,486
KC2 W FR	0,8431	**0,8445**	**0,8416**	NA
PC1 W PC1	**0,6957**	0,6876	**0,6704**	0,614
PC1 W CM1	0,6660	**0,6885**	**0,6704**	0,653
PC1 W JM1	0,6797	0,5828	**0,6704**	0,621
PC1 W KC1	0,6648	0,6292	**0,6704**	**0,759**
PC1 W KC2	0,6591	0,6872	**0,6704**	0,711
PC1 W FR	0,6944	0,6876	**0,6704**	NA
UNION	0,7076	0,6136	0,7092	**0,737**

Table 5.32 AUC values of FISs and ANN regarding Halstead metrics

Ruleset **Dataset**	**Wang-Mendel**	**IVTURS-FARC**	**DF-HFS**	**ANN**
CM1 W CM1	0,7236	0,7113	**0,7056**	**0,824**
CM1 W JM1	0,7274	0,7064	**0,7056**	0,292
CM1 W KC1	0,7283	0,7345	**0,7056**	0,212
CM1 W KC2	0,7180	**0,7491**	**0,7056**	0,164
CM1 W PC1	0,6998	0,7360	**0,7056**	0,294
CM1 W FR	**0,7295**	0,7379	**0,7056**	NA
JM1 W JM1	0,6202	0,6087	**0,6192**	0,733
JM1 W CM1	0,6210	0,6254	**0,6192**	0,7071
JM1 W KC1	**0,6252**	**0,6277**	**0,6192**	0,783
JM1 W KC2	0,6198	0,6203	**0,6192**	**0,847**
JM1 W PC1	0,6208	0,6271	**0,6192**	0,719
JM1 W FR	**0,6252**	0,6276	**0,6192**	NA
KC1 W KC1	0,7865	**0,7951**	**0,7932**	0,783
KC1 W CM1	**0,7916**	0,7737	**0,7932**	0,704
KC1 W JM1	0,7847	0,7829	**0,7932**	0,690
KC1 W KC2	0,7499	0,7799	**0,7932**	**0,835**
KC1 W PC1	0,7840	0,7899	**0,7932**	0,660
KC1 W FR	0,7856	0,7933	**0,7932**	NA
KC2 W KC2	0,8249	**0,8446**	**0,8254**	0,815
KC2 W CM1	0,8287	0,8269	**0,8254**	0,772
KC2 W JM1	0,8355	0,8311	**0,8254**	0,697
KC2 W KC1	0,8371	0,8414	**0,8254**	0,753
KC2 W PC1	0,8301	0,8413	**0,8254**	**0,834**
KC2 W FR	**0,8395**	0,8410	**0,8254**	NA

PC1 W PC1	0,6849	0,6742	**0,6709**	0,614
PC1 W CM1	0,6867	0,6852	**0,6709**	0,653
PC1 W JM1	0,7126	0,6323	**0,6709**	0,621
PC1 W KC1	0,6888	0,6914	**0,6709**	**0,759**
PC1 W KC2	**0,7134**	**0,6986**	**0,6709**	0,711
PC1 W FR	0,6890	0,6929	**0,6709**	NA
UNION	0,6483	**0,6526**	0,6458	0,641

5.3. Experimental Results

The experimental results are:

1) Although the Kc2 dataset commonly studied with machine learning algorithms, the fuzzy approach used in our study is also quite successful and competitive with other approaches. Indeed, our study outperforms all other algorithms except the study of Kalsoom et al. [74].

2) In the experiments with the WM algorithm, FISs achieve better AUC values when tested with rules created using datasets of other projects instead of their project. The FIS for Pc1 is the only project that disrupts this conclusion by giving the most successful AUC value with the ruleset generated from its data. This emphasizes that the Pc1 project is the most consistent dataset.

3) In the experiments with the IVTURS-FARC algorithm, FISs are more accurate when tested with the rules generated from other projects that consist of McCabe metrics. In contrast, Kc1 and Kc2 projects consisting of Halstead metrics are more accurate with their own ruleset.

4) In the experiments conducted with the DF-HFS algorithm, the same AUC value is taken in different scenarios created for a project since the algorithm generates the same set of rules for each project.

5) In projects that consist of McCabe metrics, ANN usually performs better when tested with its dataset. On the other hand, for the projects that include Halstead metrics, ANN is more accurate when tested with the dataset of other projects except for the Cm1 and Kc2 projects. It concludes that the minimum and maximum values of the dataset used to train the model covers the dataset used for the test.

6) In the experiments performed on the Cm1 project with the ANN algorithm, there are huge differences in performance values. AUC values decrease dramatically when the Cm1 project is trained with its data and tested with the data of other projects. Since Cm1 is a small dataset, it cannot exhibit a general behavior when it is learned from its data. However, FISs tested with rules generated by Cm1 data gives highly accurate results.

7) In experiments with the union dataset, a dataset consisting of McCabe metrics gives the most accurate performance result with the ANN algorithm. In contrast, the dataset consisting of Halstead metrics gives the best AUC value with the WM algorithm.

8) The most vital result of our study, FIS does not have to provide the most accurate performance results while tested with its own ruleset. As shown in the experiments, FIS presents the best AUC values with the rulesets produced from the more consistent datasets.

In addition to the experiments performed, the results of AUC values in our study were compared with other studies in Table 5.33.

Table 5.33 Comparison of the performance results with other studies

Study	**Year**	**Method**	**Dataset**	**AUC**
Catal and Diri [9]	2008	Random Forest	Kc2	0,79
Riquelme et al. [72]	2008	Naïve Bayes	Kc2	0,83
Mende et al. [29]	2009	Random Forest	Kc2	0,84
De Carvalho et al. [28]	2010	SVM	Kc2	0,6460
Erturk and Sezer [6]	2016	FRBS	Kc2	0,7304
Yohannese et al. [73]	2017	Ensemble Learning Algorithms Information Gain	Kc2	0,801
Kalsoom et al. [74]	2018	SMOTE + Random Forest	Kc2	0,93
Mutlu et al. [38]	2018	FRBS	Kc2	0,8272
This Study	**2020**	**FRBS**	**Kc2**	**0,8446**

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

FISs are based on two main components, “the inference system” and “the fuzzy rule base”. The inference system that implements the fuzzy inference process necessary to obtain an output from the FIS when an input is specified. The fuzzy rule base, a collection of fuzzy rules that represents knowledge about the problem. The linguistic fuzzy rule base is determined by domain experts. Setting the rules is a very time consuming and difficult task for the experts. The problem domain may not always be simple and straightforward, and when the problem is complicated, it may not be possible to determine the rule base for the expert. In such cases, the rule base is generated using automatic rule generation methods.

In literature, researchers have always been produced the rules using their own project data. In order to solve similar problems more quickly and effectively, the previously acquired information should be transferred to the new problem domain. In this study, the knowledge acquired from previous studies is transferred through the interpretable rules to the new problem domains. The investigated project is tested with FISs created with the rules produced from its own project and the datasets of other projects. We drew attention to the rules that are produced by using datasets other than their own dataset, and we proved that better performance could be achieved with these rulesets.

In the rule generation process, three automatic rule generation approaches are employed, namely WM, IVTURS-FARC, and the rule generation scheme of DF-HFS. In order to demonstrate that the portability of the rules is not only related to rule generation algorithms but also the dataset used, we perform experiments on the five different projects (CM1, JM1, KC1, KC2, and PC1) in which each process corresponded to metrics of a software module on the scope of SFP.

The rules are created for both McCabe and Halstead's metrics of each project. In the experiments with McCabe metrics, the ruleset obtained from the Pc1 project is more accurate than the ruleset of the corresponding project. On the other hand, for Halstead metrics, we realized that the ruleset produced from the Kc1 project is more consistent instead of the ruleset of the corresponding project in most of the circumstances.

We also investigate ANN for validation purposes. From the point of view of ANN, it is quite unsuccessful when it cannot generalize the knowledge it learns from its own dataset and is tested with the dataset of other projects. However, it is successful when training with the most comprehensive dataset and tested with other projects. Furthermore, it is difficult to understand and interpret information learned from neural networks. The information cannot be transferred to another project since the internal structure of the neural networks is unknown.

On the other hand, fuzzy rule-based models are easy to understand since they use linguistic variables and interpretable rulesets. Although neural models are frequently used in prediction problems, the results show that FRBSs are highly acceptable and competitive with ANN. In ANN, reasonable accuracy can only be achieved by a complete dataset of a project, which is not possible in the early phases of the software development life cycle. On the contrary, FISs can be used at any stage of the development, because as we see in this study, having a complete dataset for the corresponding project is not vital for the reasonable behavior of the inference system.

Experiment results conclude that the dataset of a project may not provide the best-performed rules. In fact, the most accurate rules can be derived from another dataset corresponding to the project in SFP. This proves that the portability of the rulesets between different projects.

6.2. Future Work

As we pointed out in the experiments and results section, we have only considered the cases where the input metrics of the source project and the target project have the same dimensions. In future studies, it will be useful to address the more challenging problem where the two projects have different feature sizes.

The Halstead and McCabe features were defined in the 1970s, and since then, technology has considerably improved. In addition to the Halstead and McCabe metrics, which are intramodule metrics, it should be possible to define new intermodule metrics that give better predictors of the defect.

REFERENCES

[1] M.J. Gacto, R. Alcalá, F. Herrera, Inf. Sci. (Ny). 181 (2011) 4340–4360.

[2] R. Mikut, J. Jäkel, L. Gröll, Fuzzy Sets Syst. 150 (2005) 179–197.

[3] L.-X. Wang, J.M. Mendel, IEEE Trans. Syst. Man. Cybern. 22 (1992) 1414–1427.

[4] J.A. Sanz, A. Fernandez, H. Bustince, F. Herrera, IEEE Trans. Fuzzy Syst. 21 (2013) 399–411.

[5] B. Mutlu, E.A. Sezer, H.A. Nefeslioglu, Fuzzy Sets Syst. 307 (2017) 50–66.

[6] E. Erturk, E.A. Sezer, in: Int. J. Data Anal. Tech. Strateg., Inderscience Publishers, 2016, pp. 14–28.

[7] M. Reformat, in: Lect. Notes Artif. Intell. (Subseries Lect. Notes Comput. Sci., Springer Verlag, 2003, pp. 644–651.

[8] P. Singh, N.R. Pal, S. Verma, O.P. Vyas, IEEE Trans. Syst. Man, Cybern. Syst. 47 (2017) 826–837.

[9] C. Catal, B. Diri, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2008, pp. 244–257.

[10] O. Cordón, M.J. Del Jesus, F. Herrera, Int. J. Approx. Reason. 20 (1999) 21–45.

[11] H. Ishibuchi, T. Yamamoto, IEEE Trans. Fuzzy Syst. 13 (2005) 428–435.

[12] B. Mutlu, E.A. Sezer, M.A. Akcayol, IEEE Int. Conf. Fuzzy Syst. 2018-July (2018) 1–9.

[13] L.H. Yang, J. Liu, Y.M. Wang, L. Martínez, Knowl. Inf. Syst. 60 (2019) 837–878.

[14] J. Shell, S. Coupland, Inf. Sci. (Ny). 293 (2015) 59–79.

[15] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, J. Lu, IEEE Trans. Fuzzy Syst. 25 (2017) 1795–1807.

[16] O. Cordón, M.J. Del Jesus, F. Herrera, M. Lozano, Int. J. Intell. Syst. 14 (1999) 1123–1153.

[17] H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka, IEEE Trans. Fuzzy Syst. 3 (1995) 260–270.

[18] E.Y. Nagai, L.V.R. De Arruda, IFAC Proc. Vol. 15 (2002) 385–390.

[19] J. Gou, F. Hou, W. Chen, C. Wang, W. Luo, Neurocomputing 151 (2015) 1293–1304.

[20] J. Abonyi, J. Abonyi, J.A. Roubos, F. Szeifert, Int. J. Approx. Reason. 32 (2002) 1--21.

[21] J. Alcalá-Fdez, R. Alcalá, F. Herrera, IEEE Trans. Fuzzy Syst. 19 (2011) 857–872.

[22] V.D. Anezakis, M.M. Ozturk, 2018 IEEE Int. Conf. Innov. Intell. Syst. Appl. INISTA 2018 1 (2018) 1–8.

[23] E. Erturk, E.A. Sezer, Proc. IASTED Int. Conf. Softw. Eng. SE 2014 (2014) 101–108.

[24] L. Liu, K. Li, M. Shao, W. Liu, Proc. - 2015 Int. Conf. Cloud Comput. Big Data, CCBD 2015 (2016) 93–96.

[25] Z. Marian, I.G. Mircea, I.G. Czibula, G. Czibula, Proc. - 18th Int. Symp. Symb. Numer. Algorithms Sci. Comput. SYNASC 2016 (2017) 240–247.

[26] H.B. Yadav, D.K. Yadav, Inf. Softw. Technol. 63 (2015) 44–57.

[27] C. Catal, Expert Syst. Appl. 38 (2011) 4626–4636.

[28] A.B. de Carvalho, A. Pozo, S.R. Vergilio, J. Syst. Softw. 83 (2010) 868–882.

[29] T. Mende, R. Koschke, in: ACM Int. Conf. Proceeding Ser., 2009.

[30] Univ. Ottawa (2004).

[31] (n.d.).

[32] L.A. Zadeh, Inf. Control 8 (1965) 338–353.

[33] I.J. Eyoh, U.A. Umoh, Br. J. Psychiatry 111 (1965) 1009–1010.

[34] T. Takagi, M. Sugeno, IEEE Trans. Syst. Man Cybern. SMC-15 (1985) 116–132.

[35] Y. TSUKAMOTO, in: Readings Fuzzy Sets Intell. Syst., Elsevier, 1993, pp. 523–529.

[36] D.C. Park, R.J. Marks, L.E. Atlas, M.J. Damborg, IEEE Transadions Power Syst. 6 (1991) 442–449.

[37] C. Catal, B. Diri, Expert Syst. Appl. 36 (2009) 7346–7354.

[38] B. Mutlu, E.A. Sezer, M.A. Akcayol, in: UBMK 2018 - 3rd Int. Conf. Comput. Sci. Eng., 2018, pp. 209–214.

[39] S. Mitra, Y. Hayashi, IEEE Trans. Neural Networks 11 (2000) 748–768.

[40] F. Pourpanah, C.P. Lim, J.M. Saleh, Expert Syst. Appl. 49 (2016) 74–85.

[41] E.I. Papageorgiou, Appl. Soft Comput. J. 11 (2011) 500–513.

[42] R.S. Michalski, R.L. Chilausky, Int. J. Hum. Comput. Stud. 51 (1999) 239–263.

[43] M. Kulkarni, S. Kulkarni, Int. J. Comput. Appl. 143 (2016) 30–35.

[44] J. Nahar, T. Imam, K.S. Tickle, Y.P.P. Chen, Expert Syst. Appl. 40 (2013) 1086–1093.

[45] M. Setnes, IEEE Trans. Fuzzy Syst. 8 (2000) 416–424.

[46] K. Nozaki, H. Ishibuchi, H. Tanaka, Fuzzy Sets Syst. 86 (1997) 251–270.

[47] A. González, R. Pérez, IEEE Trans. Fuzzy Syst. 7 (1999) 176–191.

[48] G. Castellano, A.M. Fanelli, C. Mencar, Cogn. Syst. Res. 3 (2002) 125–144.

[49] S.J. Pan, Q. Yang, IEEE Trans. Knowl. Data Eng. 22 (2010) 1345–1359.

[50] Q. Liu, X. Liao, L. Carin, in: Adv. Neural Inf. Process. Syst., 2008, pp. 937–944.

[51] H. Zuo, G. Zhang, V. Behbood, J. Lu, X. Meng, in: Proc. - 2015 10th Int. Conf. Intell. Syst. Knowl. Eng. ISKE 2015, Institute of Electrical and Electronics Engineers Inc., 2016, pp. 183–188.

[52] J. Yang, R. Yan, A.G. Hauptmann, in: Proc. ACM Int. Multimed. Conf. Exhib., 2007, pp. 188–197.

[53] Q. Yang, Y. Chen, G.-R. Xue, W. Dai, Y. Yu, Heterogeneous Transfer Learning for Image Clustering via the Social Web, 2009.

[54] S. Amos, in: Dataset Shift Mach. Learn., 2013, pp. 2–28.

[55] W. Dai, Q. Yang, G.R. Xue, Y. Yu, in: ACM Int. Conf. Proceeding Ser., 2007, pp. 193–200.

[56] H. Zuo, G. Zhang, V. Behbood, J. Lu, in: 16th World Congr. Int. (IFSA)/9th Conf. Eur., 2015.

[57] L. Duan, I.W. Tsang, D. Xu, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2012)

465–479.

[58] T.G. Dietterich, P. Domingos, L. Getoor, S. Muggleton, P. Tadepalli, Mach. Learn. 73 (2008) 3–23.

[59] J. Huang, A.J. Smola, A. Gretton, K.M. Borgwardt, B. Schölkopf, in: Adv. Neural Inf. Process. Syst., 2007, pp. 601–608.

[60] S. Bickel, M. Brückner, T. Scheffer, in: ACM Int. Conf. Proceeding Ser., 2007, pp. 81–88.

[61] Z. Wang, Y. Song, C. Zhang, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2008, pp. 550–565.

[62] W. Dai, Q. Yang, G.R. Xue, Y. Yu, in: Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 200–207.

[63] L. Borzemski, G. Starczewski, in: Proc. - 2009 1st Asian Conf. Intell. Inf. Database Syst. ACIIDS 2009, 2009, pp. 28–33.

[64] P. Yang, Q. Tan, Y. Ding, in: Proc. - Int. Conf. Comput. Sci. Softw. Eng. CSSE 2008, 2008, pp. 62–65.

[65] C.B. Do, A.Y. Ng, in: Adv. Neural Inf. Process. Syst., 2005, pp. 299–306.

[66] S. Chatterjee, B. Maji, Soft Comput. 20 (2016) 4023–4035.

[67] I.G. Czibula, G. Czibula, Z. Marian, V.S. Ionescu, Stud. Informatics Control 25 (2016) 207–216.

[68] A. Kaur, P.S. Sandhu, A.S. Brar, in: 2009 2nd Int. Conf. Mach. Vision, ICMV 2009, 2009, pp. 242–245.

[69] T. Menzies, J. Greenwald, A. Frank, IEEE Trans. Softw. Eng. 33 (2007) 2–13.

[70] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, Soft Comput. 13 (2009) 307–318.

[71] (n.d.).

[72] J.C. Riquelme, R. Ruiz, D. Rodríguez, J. Moreno, in: ADIS 2008 - Apoyo a La Decis. En Ing. Del Software, Even. Realiz. En El Marco Las 13th Jornadas Ing. Del Softw. y Bases Datos, JISBD 2008, 2008, pp. 67–74.

[73] C.W. Yohannese, T. Li, M. Simfukwe, F. Khurshid, in: Proc. 2017 12th Int. Conf. Intell. Syst. Knowl. Eng. ISKE 2017, 2018, pp. 1–6.

[74] A. Kalsoom, M. Maqsood, M.A. Ghazanfar, F. Aadil, S. Rho, J. Supercomput. 74 (2018) 4568–4602.

APPENDICES

APPEXDIX 1 – Proceeding that has been accepted for the publication

Assessment of Fuzzy Rule Generation Methods by Considering Portability

Nazli Ece Uykur[1][0000−0001−5740−0070], Begum Mutlu[2][0000−0003−1960−2143], and
Ebru Akcapinar Sezer[3][0000−0002−9287−2679]

1 Software Technologies Research Institute, The Scientific and Technological Research Council of Turkey (TUBITAK), Ankara, Turkey
nazli.uykur@hacettepe.edu.tr
2 Department of Computer Engineering, Gazi University, Ankara, Turkey
begummutlu@gazi.edu.tr
3 Department of Computer Engineering, Hacettepe University, Ankara, Turkey
ebruakcapinarsezer@gmail.com

Abstract. Interpretability is one of the most essential reasons for the common use of fuzzy rule-based systems, compared with the black-box machine learning models. This interpretability comes from the linguistically expressible fuzzy sets and fuzzy rules. In general approach, the rules are generated by human experts; however, in case of non-existence of expert opinion, or investigated problem is getting complex to be handled by a human, existing datasets can be used to provide fuzzy rules by automatic rule generation approaches. The existing studies have been focused on rule generation methods to provide more accurate rules. However, they have not deeply analyzed the utilized datasets for this purpose. The dataset for a real-world problem may mislead the model. Perhaps a more consistent dataset exists to generate rules that can be easily applied to new datasets with the same characteristics, without having to re-generate rules every time when the dataset is changed. However, the standard process of train-test set partitioning makes this situation stay latent. This study was empirically addressed this issue for the software fault prediction problem. We investigated the portability of the ruleset by generating rules from different datasets for a specific project. The datasets of five projects were acquired and fuzzy systems obtained from using several combinations of these datasets were evaluated by their accuracy. The results show that more accurate rules can be obtained from other projects instead of their own projects. In addition, it emphasizes that the resulting ruleset can be transferred to other projects by the portability property of the rules.

Keywords: Fuzzy rule learning · Portability of fuzzy systems · Software fault prediction.

INTRODUCTION

Fuzzy rule-based systems (FRBSs) have been successfully employed in numerous studies on several research fields [1,8,15,32,41]. Even though the success of the applications inevitably depend on the membership function (MF) modeling and choice of fuzzy reasoning methods, the selection of the fuzzy ruleset has always been vital in inference accuracy. One of the most important advantages of using FRBSs is the interpretability and portability of these rules. Interpretability is an understanding of the real system behavior by human beings by inspecting the fuzzy rules [15,26]. It depends on the model structure such as the number of input variables, the number of fuzzy rules, the complexity of each individual rule and the shape of MFs. There are other interpretable systems such as decision trees and fuzzy cognitive maps just like FRBSs. In these systems, the model establishes a cause-effect relationship within itself. Since these relationships are not known in classical machine learning algorithms, it is not known whether the correct causes are associated with the correct effects. Machine learning algorithms can also establish a natural relationship with unrelated features. Since the established relationships are not known by the domain expert, it is not clear how to move the model to another application field. Interpretable models are transparent to the domain expert. On the other hand, non-interpretable models must be retrained when investigating a new domain. Thus, new relationships are established in the new domain. It is not known whether the new cause-effect relationships are the same as those in the old domain. In FRBSs, these relations are obtained through rules.

The quality of the rules depends on the quality of the dataset used to produce the rule. If the rules are obtained from data rather than the expert, certain information is discarded and the quality of the data should be investigated. In addition, when the dataset is divided, in fact, more information is lost. It is clear that the quality of the data space affects the quality of the generated rules. In the literature, the rules have been produced from the source dataset so far. More specifically, both the fuzzy applications of software fault prediction (SFP) [14,31,37] and the other studies [9,19,28,39] produced the rules using their own datasets. However, rules should be produced using the highest quality dataset as much as possible. Thus, we obtain the best ruleset and make the resulting FRBS have a better insight about the corresponding problem. By portability property of FRBSs, the information is transferred from an existing similar but not identical source project to the target project to improve the learning [36,41]. Subsequently, once the ruleset is transferred to another domain, it becomes possible to use it as a rule template, and apply it according to the characteristics (MFs) of different target domains. This means that the best performing fuzzy ruleset can be used in different domains for the same research problem, contrary to machine learning approaches that are strictly dependent on domain-specific data. In this study, we criticize that the set of rules to be used in the investigated problem is always produced from its own dataset. In order to determine the best ruleset of a problem, we investigated whether it would be better to use the dataset of the corresponding project itself or to use another dataset that performs better for this problem domain.

Ideally, the rules should be provided from a domain expert who can comprehensively consider the problem and the influencing factors of it. The resulting system provides a general solution in situations where human perspective and domain knowledge are reflected to the fuzzy rules. However, this knowledge is difficult to obtain from most of the real-world problems. Because the problems usually influenced by several factors. The use of several factors as independent variables of an inference problem makes the targeted rule base high dimensional. The increase in the number of these variables causes both a linear increase in the complexity of each rule and an exponential increase in the total number of fuzzy rules to be determined. In these circumstances, the domain expert may be inadequate to define the relationship of all system variables to the output and generate rules over these relationships. Therefore, rule production becomes very challenging for the expert in such a high dimensional variable set, since rules present the know-how of the domain. Because of this bottleneck in human-based rule determination, the researchers have transferred their effort in another direction, automatic rule generation from historical knowledge, i.e. existing datasets.

The automatic generation of rules can be performed by data-driven methods [4,16,30,35] or expert-cooperated procedures [28,29]. The model produced from data-driven approaches consists entirely of input-output relationships in the data, while expert-cooperated procedures use some general information obtained from data or expert during the rule generation procedure.

The majority of recent researches on automatic rule generation has been focused on evolutionary optimization algorithms [10,34], clustering [16,35], genetic algorithms [6,18], linear solutions [28,38], machine learning approaches such as decision tree [2] and association rule mining [4,30]. Although data-driven rule learning methods are widely used and serve reasonable rules to FRBSs, these systems are always domain-specific and restricted with the limits of the dataset. This is the natural outcome of data-dependent machine learning models. An obstacle to the more widespread acceptance of data-driven methods (such as artificial neural networks) is incapable of explaining how the network has reached a particular decision in a human-understandable manner. On the other hand, FRBSs are easy to understand since they use linguistic fuzzy $IF\text{-}THEN$ rules.

In order to perform this investigation, some automatic rule learning methods were performed on the SFP problem. The error rate increases as the size of the software grow. In order to improve software quality, the error rate should be decreased. Fault prediction activities also serve to this purpose, it aims to detect faults in the software automatically. In recent years, there are many studies for SFP problem [5,13,14,22,23,29]. The SFP problem has an advantage based on the dataset since there are a lot of datasets from various projects on the scope of SFP. CM1, JM1, KC1, KC2, PC1 datasets, which are commonly used in SFP [7,12,25], were obtained from PROMISE Software Engineering Repository [11]. The rules are generated using these datasets by automatic rule generation methods, which are Wang-Mendel's rule generation method (WM) [38], Interval Valued Fuzzy Reasoning Method with Tuning and Rule Selection (IVTURS) [34] and the expert-cooperated rule production scheme proposed in Defuzzification-free Hierarchical Fuzzy System (DF-HFS) [29]. Subsequently, several FRBSs were generated and the performance evaluation of each resulting FRBS was employed based on the accuracy of the inference tendency of system.

The misleading rules obtained from the rule learning methods may be caused by misleading situations of data. The main argument of the paper is that the dataset of rule learning should be investigated according to its ability of providing a general solution, which brings the subject into portability capability. The most leading dataset should be used during the rule generation process. The best ruleset may be obtained from different datasets since there is no strict rule that the ruleset can only be produced by using its own dataset. In this paper, the experiments on SFP presented that the automatically generated rules obtained from the datasets from other projects can be more accurate since they contain more consistent data for that project. Through this empirical study, it was also presented that the rules are portable if the underlying dataset can be selected accurately.

1 FUZZY RULE GENERATION ALGORITHMS

1.1 Wang-Mendel

Being one of the most pioneering rule generation methods in this domain, WM is a widely used solution since it is not a problem-dependent but a general solution and it does not contain exhaustive learning/optimization iterations. By these properties it is being straightforward and easy to understand [38]. It combines both linguistic and numerical information by producing fuzzy rules from numerical data. The rule learning process of WM consists of three steps. First, it divides the input and output spaces into fuzzy regions. Second, fuzzy rules are generated from given each transaction of data. Since the number of transactions will be very high, the number of rules produced is also high. This strategy may cause conflicting rules, which consist of the same antecedent but different consequent. This conflict is solved in the third step by determining a degree to each candidate rule and selecting the rule with the greatest degree as the rule in the final ruleset. The final linguistic rule form is shown in Eq.(1) where I and O represent input and output linguistic variables respectively, m is the number of input and f corresponds fuzzy sets. By WM, each rule contains all input variables which are connected with logical AND operator.

$$IF\ I_1\ is\ f_i\ AND\ I_2\ is\ f_j\ AND\ \dots\ I_m\ is\ f_k\ THEN\ O\ is\ f_p \qquad (1)$$

1.2 IVTURS-FARC

IVTURS is an interval-valued fuzzy rule learning classification method [34]. The method is called IVTURS-FARC, because FARC-HD [9] algorithm is based in the rule learning process. IVTURS built with three steps. Firstly, the algorithm initializes the interval-valued FRBSs by using FARC-HD algorithm [9] and generates an interval-valued equivalence function for every single variable. The second step creates the new fuzzy reasoning method by extending it which is based on interval-valued fuzzy sets. The final step is composed of optimization tasks, which corresponds tuning of the MFs and selecting the rule base to reduce computational cost.

1.3 Rule Generation Scheme of DF-HFS

In decision-making mechanisms, data may not always be complete, consistent, or be even exist. Mutlu et al. [29] propose a very simple but effective method to overcome these issues. DF-HFS generates a complete ruleset that contains every possible situation of inputs and outputs. The method does not require expert knowledge, but small information that how much the input MFs affect the output. The researchers proposed a simple procedure to measure this effect by using little information of data without implementing a whole learning process. This effect called as *rating factor*. The degree of a consequent rule (R^{cons}) is formed by a simple summation operation based on the rating factors (see Eq.(2)). In this study, the output divided into two equal intervals as K_1 = [0-0.5), K_2 = [0.5-1]. The fuzzy set which is triggered by (R^{cons}) at the highest level is selected as the consequent fuzzy set of corresponding rule. For instance, if it is calculated as 0.32, then output belongs to K_1 which is in the first fuzzy set. The complete ruleset is rearranged by determining the output fuzzy sets for each rule.

$$R_{cons} = \sum_{i=1}^{m} RatingFactor_{(i,f)} \tag{2}$$

At the end of the rule generation procedure of DF-HFS, (n^m) rules obtained in resulting ruleset, where n is the number of fuzzy sets and m is the number of inputs.

2 EXPERIMENTS

2.1 Datasets

CM1, JM1, KC1, KC2, PC1 software defect prediction datasets were obtained from PROMISE software engineering repository [11]. These datasets commonly preferred in the software engineering research area [7,12,27,33] and also they all have the same software metrics. Each dataset contains total 22 metrics belongs to the software modules which has 5 different lines of code measure, 3 McCabe metrics [24], 4 base Halstead measures [17], 8 derived Halstead measures [17], 1 branch-count and 1 output variable which defines the instance whether has defects or not. Table 1 shows the characteristics of the selected datasets.

Input metrics were reduced to 9 that consists of 4 McCabe (*loc, v(g), ev(g), iv(g)*) and 5 Halstead metrics (*v, d, i, n, e*) and explained as follows:

- **Line count of code** (loc) : Measures the line count of code.
- **Cyclomatic complexity** (v(g)) : Measures the number of linearly independent paths.
- **Essential complexity** (ev(g)) : Measures the degree of unstructured constructs.
- **Design complexity** (iv(g)) : Measures the amount of integration between modules.
- **Volume** (v) : The number of bits required for storing the program.
- **Difficulty** (d) : Measures the ability of a program to be written or understood.
- **Intelligence** (i) : Measures the complexity of a particular algorithm, regardless of the language used.
- **Length** (n) : The total number of operator and operand appearances.
- **Effort** (e) : The estimated time required to implement the program.

Table 1. Dataset specifications

Data	Language	# Modules	# Faulty Modules	Faulty Modules %
CM1	C	498	49	9.83%
JM1	C	10885	8779	80.65%
KC1	C++	2109	326	15.45%
KC2	C++	522	105	20.5%
PC1	C	1109	1032	93.05%

2.2 Case Study: Software Fault Prediction

In the experiments, three fuzzy rule generation algorithms (WM, IVTURSFARC, and DF-HFS) and an artificial neural network (ANN) were implemented which make a difference in certain aspects. WM and IVTURS-FARC methods are based on the relations of all input-output data, while DF-HFS approach generates rules based on few information of data. On the other hand, ANN is a data-driven approach that requires data to learn and completely dependent on data.

WM and DF-HFS rule generation methods were implemented with Java programming language. Knowledge Extraction based on Evolutionary Learning (KEEL) open source Java software tool [3] was used for IVTURS-FARC algorithm. Regarding ANN, implementation details are as follows:

- Number of neurons: 48 in the first layer, 24 in the second layer, 1 in the third layer
- Number of epochs: 100
- Activation function: Sigmoid
- Training algorithm: Gradient descent
- Environment: Python/Keras library [21]
- 5-fold cross validation: 80%-20% of data for training and evaluation

In the rule generation process, three and two fuzzy sets are used for each input and output linguistic variables respectively. These fuzzy sets were divided into equal intervals. In the implementation of rule generation approaches, rules were generated from McCabe and Halstead metrics of each dataset with all algorithms. Rule weights were set to 1 for each algorithm to make a fair comparison. Table 2 presents the number of fuzzy rules generated by using McCabe and Halstead metrics.

Table 2. The number of rules generated by WM, IVTURS-FARC, DF-HFS.

Metric	**McCabe**						**Halstead**					
Ruleset	**Cm1**	**Jm1**	**Kc1**	**Kc2**	**Pc1**	**FR**	**Cm1**	**Jm1**	**Kc1**	**Kc2**	**Pc1**	**FR**
WM	16	13	18	4	9	5	15	17	14	8	11	10
IVTURS-FARC	7	4	6	4	7	7	4	3	8	6	6	6
DF-HFS	81	81	81	81	81	81	243	243	243	243	243	243

Rule generation algorithm of DF-HFS produces a complete ruleset, the number of resulting rules was constant (3^4 and 3^5 for McCabe and Halstead metrics respectively) in these tables. The rules generated from each dataset were combined and the repeated ones form a customized ruleset called as *frequent rules (FR)*. For example, the total number of rules generated by the WM algorithm using McCabe metrics were 60 (rule numbers of Cm1 + Jm1 + Kc1 + Kc2 + Pc1 projects). A total of 5 rules out of 60 rules form a set of FR since they are generated from more than one project. The FR contains all the rules generated by DF-HFS rule generation scheme since the rules generated for each dataset were the same.

The ruleset was presented in Table 3 that was generated from Pc1 and Kc1 projects by IVTURS-FARC algorithm.

Table 3. Rules belong to Pc1 and Kc1 datasets which consist of McCabe and Halstead metrics respectively

Metrics	**Dataset**	**Rules**
McCabe	Pc1	If (iv(g) is M) then (faultiness is H)
		If (iv(g) is L) then (faultiness is L)
		If (ev(g) is L) then (faultiness is L)
		If (ev(g) is M) then (faultiness is H)
		If (v(g) is L) then (faultiness is L)
		If (loc is M) then (faultiness is H)
		If (loc is L) then (faultiness is L)

Halstead	Kc1	If (d is H) then (faultiness is H) If (d is M) then (faultiness is H) If (n is L) and (v is L) then (faultiness is H) If (n is L) and (v is L) and (e is L) then (faultiness is H) If (d is M) and (e is L) then (faultiness is H) If (e is L) then (faultiness is L) If (v is L) and (i is L) then (faultiness is L) If (d is L) and (i is L) then (faultiness is L)

After the rule evaluation phase, the resulting rulesets were included in FRBSs. The common properties of FRBSs are as follows:

- Type of inference: Mamdani
- Implication-aggregation-defuzzification: minimum-maximum-centroid
- Environment: MATLAB 9.5.0 (R2018b)

In order to test the modeled FRBSs, several scenarios were defined that led to the creation of six FRBSs for each software project. In the first scenario, the FRBS in a certain project was generated by using the data of the current project to determine the MF parameters and generate the rules. In the other four scenarios, the MFs of an FRBS created for a specific project were determined by using the data of current project similar to the first scenario. On the other hand, the ruleset was generated from other projects instead of its own dataset. In the example, the FRBSs of CM1 were modeled by using its own project for the determination of MF parameters. On the other hand, the datasets of JM1, KC1, KC2, and PC1 projects were utilized for rule generation which led to four FRBSs to be obtained. Unlike other scenarios, in the sixth scenario, *FR* is used as a ruleset. In the last scenario, all acquired datasets were combined to create a *union* dataset, the rules and the MF parameters of an FRBS were generated from this dataset.

Only five scenarios were defined for ANN and they performed differently from fuzzy rule generation algorithms. As a regular machine learning evaluation process, in the first scenario, the ANNs were trained with 80% and tested with 20% of the data respectively for each project. In the other four scenarios, the ANNs were trained with the data (100%)

of current projects and tested with the data (100%) of source projects. In the example, CM1 was trained with the CM1 and tested with the other four datasets. Here, each aforementioned scenario corresponds to an experiment that is performed for both McCabe and Halstead metrics respectively. The performance results of FRBSs and ANNs were presented in Table 4 and Table 5.

Area Under Receiver Operating Characteristic Curve (ROC-AUC) values were measured to evaluate the resulting experiments. Here the scenarios for each project were addressed by presenting the name of the current project and the source project concatenated by W to generate fuzzy rules together. For example, $CM1\ W\,JM1$ means, the FRBS was created for $CM1$, while its rules were generated by using $JM1$ dataset. The results obtained from the experiments based on several scenarios are as follows:

- Regarding the FRBSs whose rules were obtained from WM, FRBSs which were evaluated with the rulesets of other projects achieved better AUC values. The only exception is the Pc1 dataset which has performed better with its own ruleset. This means that the best ruleset belongs to the Pc1 dataset.

  Regarding the FRBSs whose rules were obtained from IVTURS-FARC, for the projects that are composed of Halstead metrics, better AUC values were obtained with datasets of other projects, while Kc1 and Kc2 were given the most successful performance results with their own set of rules.
- The same AUC values were taken since DF-HFS rule generation algorithm produced the same complete ruleset for each project.
- ANN performed mostly better when tested with their own projects which are composed of McCabe metrics. On the other hand, projects that consist of Halstead metrics achieved better AUC values when tested with other projects except Cm1 and Kc2 projects. This means that the minimum and maximum values of the dataset used to train the model cover the dataset used in the test.
- Regarding Cm1 project, experiments on ANN showed that a considerable difference appeared between the evaluation results of models. Specifically, basing the AUC

values obtained from $CM1\ W\,CM1$, these values decreased dramatically when the model was tested with other projects. This situation can be explained by the fact that Cm1 is a small dataset, and ANN can not provide a general solution when it learned from Cm1.

Table 4. ROC-AUC values of FRBSs and ANN regarding McCabe metrics.

Dataset W/ Ruleset	**WM**	**IVTURS FARC**	**DF-HFS**	**ANN**
CM1 W CM1	0,7003	0,7239	**0,7247**	**0,676**
CM1 W JM1	0,7202	0,5863	**0,7247**	0,485
CM1 W KC1	0,7183	0,6973	**0,7247**	0,279
CM1 W KC2	0,7118	**0,7514**	**0,7247**	0,298
CM1 W PC1	**0,7415**	0,7239	**0,7247**	0,483
CM1 W FR	0,7383	0,7221	**0,7247**	NA
JM1 W JM1	0,6888	0,6116	**0,6909**	**0,60**
JM1 W CM1	0,6817	0,6897	**0,6909**	0,485
JM1 W KC1	0,6889	0,6603	**0,6909**	0,463
JM1 W KC2	0,6882	**0,7087**	**0,6909**	0,484
JM1 W PC1	**0,7010**	0,6897	**0,6909**	0,487
JM1 W FR	**0,7010**	0,6899	**0,6909**	NA
KC1 W KC1	0,7897	0,7212	**0,7903**	**0,793**
KC1 W CM1	0,7891	**0,7914**	**0,7903**	0,689
KC1 W JM1	**0,7920**	0,6111	**0,7903**	0,608
KC1 W KC2	0,7907	0,7904	**0,7903**	0,749
KC1 W PC1	0,7908	**0,7914**	**0,7903**	0,632
KC1 W FR	0,7905	0,7897	**0,7903**	NA
KC2 W KC2	0,8416	0,8422	**0,8416**	**0,823**
KC2 W CM1	**0,8438**	0,8441	**0,8416**	0,651
KC2 W JM1	0,8417	0,7073	**0,8416**	0,581
KC2 W KC1	0,8398	0,8184	**0,8416**	0,440
KC2 W PC1	0,8430	0,8441	**0,8416**	0,486
KC2 W FR	0,8431	**0,8445**	**0,8416**	NA
PC1 W PC1	**0,6957**	0,6876	**0,6704**	0,614
PC1 W CM1	0,6660	**0,6885**	**0,6704**	0,653
PC1 W JM1	0,6797	0,5828	**0,6704**	0,621
PC1 W KC1	0,6648	0,6292	**0,6704**	**0,759**
PC1 W KC2	0,6591	0,6872	**0,6704**	0,711
PC1 W FR	0,6944	0,6876	**0,6704**	NA
UNION	0,7076	0,6136	0,7092	**0,737**

Table 5. ROC-AUC values of FRBSs and ANN regarding Halstead metrics.

Dataset W/ Ruleset	**WM**	**IVTURS FARC**	**DF-HFS**	**ANN**
CM1 W CM1	0,7236	0,7113	**0,7056**	**0,824**
CM1 W JM1	0,7274	0,7064	**0,7056**	0,292
CM1 W KC1	0,7283	0,7345	**0,7056**	0,212
CM1 W KC2	0,7180	**0,7491**	**0,7056**	0,164
CM1 W PC1	0,6998	0,7360	**0,7056**	0,294
CM1 W FR	**0,7295**	0,7379	**0,7056**	NA
JM1 W JM1	0,6202	0,6087	**0,6192**	0,733
JM1 W CM1	0,6210	0,6254	**0,6192**	0,7071
JM1 W KC1	**0,6252**	**0,6277**	**0,6192**	0,783
JM1 W KC2	0,6198	0,6203	**0,6192**	**0,847**
JM1 W PC1	0,6208	0,6271	**0,6192**	0,719
JM1 W FR	**0,6252**	0,6276	**0,6192**	NA
KC1 W KC1	0,7865	**0,7951**	**0,7932**	0,783
KC1 W CM1	**0,7916**	0,7737	**0,7932**	0,704
KC1 W JM1	0,7847	0,7829	**0,7932**	0,690
KC1 W KC2	0,7499	0,7799	**0,7932**	**0,835**
KC1 W PC1	0,7840	0,7899	**0,7932**	0,660
KC1 W FR	0,7856	0,7933	**0,7932**	NA
KC2 W KC2	0,8249	**0,8446**	**0,8254**	**0,827**
KC2 W CM1	0,8287	0,8269	**0,8254**	0,751
KC2 W JM1	0,8355	0,8311	**0,8254**	0,698
KC2 W KC1	0,8371	0,8414	**0,8254**	0,539
KC2 W PC1	0,8301	0,8413	**0,8254**	0,710
KC2 W FR	**0,8395**	0,8410	**0,8254**	NA
PC1 W PC1	0,6849	0,6742	**0,6709**	0,815
PC1 W CM1	0,6867	0,6852	**0,6709**	0,772
PC1 W JM1	0,7126	0,6323	**0,6709**	0,697
PC1 W KC1	0,6888	0,6914	**0,6709**	0,753
PC1 W KC2	**0,7134**	**0,6986**	**0,6709**	**0,834**
PC1 W FR	0,6890	0,6929	**0,6709**	NA
UNION	**0,6483**	0,6526	0,6458	0,641

a NA represents not applicable.

However, FRBSs can present more reasonable behaviors even if the fuzzy rules were generated by Cm1.

- In the experiment conducted with a union of datasets, the dataset consisting of McCabe metrics achieved the best AUC values with the ANN algorithm. On the other hand, for the experiment with Halstead metrics, FRBS with WM rules provided the best AUC value.

In addition to these experiments, a comparison of the performance results of our study and some previous studies on fuzzy logic was listed in Table 6. It is clearly seen according to the table, the performance result obtained in this study is quite competitive compared to other studies.

Table 6. Comparison of the results with other studies

Study	**Year**	**Method**	**Dataset**	**AUC**
Catal and Diri [7]	2008	Random Forest	Kc2	0,79
Riquelme et al. [33]	2008	Naive Bayes	Kc2	0,83
Mende et al. [25]	2009	Random Forest	Kc2	0,84
De Carvalho et al. [12]	2010	SVM	Kc2	0,6460
Erturk and Sezer 14]	2016	FRBS	Kc2	0,7304
Yohannese et al.[40]	2017	Ensemble Learning Algorithms + Information Gain	Kc2	0,801
Kalsoom et al. [20]	2018	SMOTE + Random Forest	Kc2	0,93
Mutlu et al. [27]	2018	FRBS	Kc2	0,8272
This Study	**2019**	**FRBS**	**Kc2**	**0,8446**

As a result, an FRBS does not have to give the most accurate results when tested with its own set of rules. The rulesets from other projects provide very successful results.

3 CONCLUSION

The most essential problem of FRBS modeling has been determining the FRBS rules accurately. Ideally, domain experts have been expected to be the main contributor of rule generation. On the other hand, obtaining knowledge may be very challenging in some complex problems. Therefore, automatic rule generation methods have been recommended in numerous studies in the existence of data rather than expert knowledge. In the literature, rules are always generated by using the dataset of the corresponding project. On the other hand, more accurate rules can be produced by using other datasets that belong to different projects. Thus, we emphasized the portability of the rules by using the most accurate ruleset in different projects.

In order to present that the portability is not strongly related with the rule generation algorithm, but the utilized dataset, three different rule generation approaches (WM, IVTURS-FARC and rule generation scheme of DF-HFS) were employed. Experiments were performed on the dataset of five different projects on the scope of SFP. For each project, the rules were both generated by using its own dataset, as usual, and by using other projects' datasets. In order to evaluate these rule bases, FRBS implementation was performed for each project. Provided results show that when working with McCabe metrics, in most of the circumstances more successful performance results were obtained when FRBSs were evaluated with the rules that were generated from Pc1 dataset instead of its own dataset. Regarding the use of Halstead metrics, the rules that were generated from Kc1 dataset were more accurate with respect to the own dataset of a project.

Furthermore, ANN was investigated for validation purposes. Regarding the neural network point of view, they provide higher accuracy results for datasets composed of Halstead metrics. However, it is difficult to understand and interpret the knowledge learned from neural networks. In contrast, fuzzy rule-based models are easy to understand since they use linguistic variables and interpretable rulesets. Though neural models have a great advantage in having several iterations to perform better on corresponding data, provided results show that FRBSs have performed quite acceptable and competitive with ANN. On the other hand, there is a serious trade-off between the accuracy and applicability of ANNs for fault prediction. Because a good accuracy can only be achieved by a complete dataset of its own project, which is highly difficult to obtain in the early stages of the software development life-cycle. Contrary, fuzzy systems can be utilized at any stage of a life-cycle because as shown in this study having a dataset for the corresponding project is not vital for fuzzy reasoning to behave plausibly. The results of experiments conclude that the dataset of a project may not lead to the most accurate rules to be generated. Indeed, the best ruleset may be obtained from other data acquired from a different project.

REFERENCES

1. Aarabi, A., Fazel-Rezai, R., Aghakhani, Y.: A fuzzy rule-based system for epileptic seizure detection in intracranial eeg. Clinical Neurophysiology **120**(9), 1648–1657 (2009). https://doi.org/10.1016/j.clinph.2009.07.002
2. Abonyi, J., Roubos, J.A., Szeifert, F.: Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. International journal of approximate reasoning **32**(1), 1–21 (2003). https://doi.org/10.1016/s0888-613x(02)00076-2
3. Alcal´a-Fdez, J.: Keel: A software tool to assess evolutionary algorithms for data mining problems (regression, classification, clustering, pattern mining and so on). http://www.keel.es/. Accessed: 2019-05-24
4. Alcala-Fdez, J., Alcala, R., Herrera, F.: A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. IEEE Transactions on Fuzzy systems **19**(5), 857–872 (2011). https://doi.org/10.1109/tfuzz.2011.2147794
5. Anezakis, V.D., Oztu¨rk, M.M.: Verification of the effectiveness of fuzzy rule-based¨ fault prediction: A replication study. In: 2018 Innovations in Intelligent Systems and Applications (INISTA), pp. 1–8. IEEE (2018)
6. Angelov, P.P., Buswell, R.A.: Automatic generation of fuzzy rule-based models from data by genetic algorithms. Information Sciences **150**(1-2), 17–31 (2003). https://doi.org/10.1016/s0020-0255(02)00367-5
7. Catal, C., Diri, B.: A fault prediction model with limited fault data to improve test process. In: International Conference on Product Focused Software Process Improvement, pp. 244–257. Springer (2008). https://doi.org/10.1007/978-3-54069566-0 21
8. Chang, P.C., Liu, C.H.: A tsk type fuzzy rule based system for stock price prediction. Expert Systems with applications **34**(1), 135–144 (2008). https://doi.org/10.1016/j.eswa.2006.08.020
9. Cordo´n, O., del Jesus, M.J., Herrera, F.: A proposal on reasoning methods in fuzzy rule-based classification systems. International Journal of Approximate Reasoning **20**(1), 21–45 (1999). https://doi.org/10.1016/s0888-613x(00)88942-2
10. Cord´on, O., del Jesu´s, M.J., Herrera, F., Lozano, M.: Mogul: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. International Journal of Intelligent Systems **14**(11), 1123–1153 (1999). https://doi.org/10.1002/(sici)1098-111x(199911)14:11¡1123::aid-int4¿3.0.co;2-6
11. Dataset, P.: Promise datasets page. http://promise.site.uottawa.ca/SERepository/datasets-page.html. Accessed: 2019-04-04
12. De Carvalho, A.B., Pozo, A., Vergilio, S.R.: A symbolic fault-prediction model based on multiobjective particle swarm optimization. Journal of Systems and Software **83**(5), 868–882 (2010). https://doi.org/10.1016/j.jss.2009.12.023
13. Erturk, E., Sezer, E.A.: Software fault prediction using fuzzy inference system and object-oriented metrics. In: Proceedings of the 13th IASTED International

Conference on Software Engineering Austria, pp. 101–108 (2014). https://doi.org/10.2316/P.2014.810-004

14. Erturk, E., Sezer, E.A.: Software fault prediction using mamdani type fuzzy inference system. International Journal of Data Analysis Techniques and Strategies **8**(1), 14–28 (2016). https://doi.org/10.1504/ijdats.2016.075971
15. Gacto, M.J., Alcala´, R., Herrera, F.: Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. Information Sciences **181**(20), 4340–4360 (2011). https://doi.org/10.1016/j.ins.2011.02.021
16. Gou, J., Hou, F., Chen, W., Wang, C., Luo, W.: Improving wang– mendel methodperformance in fuzzy rules generation using the fuzzy c-means clustering algorithm. Neurocomputing **151**, 1293–1304 (2015). https://doi.org/10.1016/j.neucom.2014.10.077
17. Halstead, M.H., et al.: Elements of software science, vol. 7. Elsevier New York (1977)
18. Ishibuchi, H., Nozaki, K., Yamamoto, N., Tanaka, H.: Selecting fuzzy if-then rules for classification problems using genetic algorithms. IEEE Transactions on fuzzy systems **3**(3), 260–270 (1995). https://doi.org/10.1109/91.413232
19. Ishibuchi, H., Yamamoto, T.: Rule weight specification in fuzzy rule-based classification systems. IEEE transactions on fuzzy systems **13**(4), 428–435 (2005)
20. Kalsoom, A., Maqsood, M., Ghazanfar, M.A., Aadil, F., Rho, S.: A dimensionality reduction-based efficient software fault prediction using fisher linear discriminant analysis (flda). The Journal of Supercomputing **74**(9), 4568–4602 (2018). https://doi.org/10.1007/s11227-018-2326-5
21. Keras.io: Home - keras documentation. https://keras.io/. Accessed: 2019-05-04
22. Liu, L., Li, K., Shao, M., Liu, W.: Fuzzy integral based on mutual information for software defect prediction. In: 2015 International Conference on Cloud Computing and Big Data (CCBD), pp. 93–96. IEEE (2015)
23. Marian, Z., Mircea, I.G., Czibula, I.G., Czibula, G.: A novel approach for software defect prediction using fuzzy decision trees. In: 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 240–247. IEEE (2016)
24. McCabe, T.J.: A complexity measure. IEEE Transactions on software Engineering (4), 308–320 (1976). https://doi.org/10.1109/tse.1976.233837
25. Mende, T., Koschke, R.: Revisiting the evaluation of defect prediction models. In: Proceedings of the 5th International Conference on Predictor Models in Software Engineering, p. 7. ACM (2009). https://doi.org/10.1145/1540438.1540448
26. Mikut, R., Ja¨kel, J., Gro¨ll, L.: Interpretability issues in data-based learning of fuzzy systems. Fuzzy sets and systems **150**(2), 179–197 (2005). https://doi.org/10.1016/s0019-9958(65)90241-x
27. Mutlu, B., Sezer, E.A., Akcayol, M.A.: Automatic rule generation of fuzzy systems: A comparative assessment on software defect prediction. In: 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 209–214. IEEE (2018). https://doi.org/10.1109/ubmk.2018.8566479

28. Mutlu, B., Sezer, E.A., Akcayol, M.A.: End-to-end hierarchical fuzzy inference solution. In: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–9. IEEE (2018). https://doi.org/10.1109/fuzz-ieee.2018.8491481
29. Mutlu, B., Sezer, E.A., Nefeslioglu, H.A.: A defuzzification-free hierarchical fuzzy system (df-hfs): Rock mass rating prediction. Fuzzy Sets and Systems **307**, 50–66 (2017). https://doi.org/10.1016/j.fss.2016.01.001
30. Nahar, J., Imam, T., Tickle, K.S., Chen, Y.P.P.: Association rule mining to detect factors which contribute to heart disease in males and females. Expert Systems with Applications **40**(4), 1086–1093 (2013). https://doi.org/10.1016/j.eswa.2012.08.028
31. Reformat, M.: A fuzzy-based meta-model for reasoning about the number of software defects. In: International Fuzzy Systems Association World Congress, pp. 644–651. Springer (2003)
32. Rezaee, B., Zarandi, M.F.: Data-driven fuzzy modeling for takagi– sugeno–kang fuzzy system. Information Sciences **180**(2), 241–255 (2010). https://doi.org/10.1016/j.ins.2009.08.021
33. Riquelme, J., Ruiz, R., Rodr´ıguez, D., Moreno, J.: Finding defective modules from highly unbalanced datasets. Actas de los Talleres de las Jornadas de Ingenier´ıa del Software y Bases de Datos **2**(1), 67–74 (2008)
34. Sanz, J.A., Fernandez, A., Bustince, H., Herrera, F.: Ivturs: A linguistic fuzzy rulebased classification system based on a new interval-valued fuzzy reasoning method with tuning and rule selection. IEEE Transactions on Fuzzy Systems **21**(3), 399– 411 (2013). https://doi.org/10.1109/tfuzz.2013.2243153
35. Setnes, M.: Supervised fuzzy clustering for rule extraction. IEEE transactions on Fuzzy Systems **8**(4), 416–424 (2000). https://doi.org/10.1109/91.868948
36. Shell, J., Coupland, S.: Fuzzy transfer learning: methodology and application. Information Sciences **293**, 59–79 (2015). https://doi.org/10.1016/j.ins.2014.09.004
37. Singh, P., Pal, N.R., Verma, S., Vyas, O.P.: Fuzzy rule-based approach for software fault prediction. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(5), 826–837 (2016)
38. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Transactions on systems, man, and cybernetics **22**(6), 1414–1427 (1992). https://doi.org/10.1109/21.199466
39. Yang, L.H., Liu, J., Wang, Y.M., Mart´ınez, L.: New activation weight calculation and parameter optimization for extended belief rule-based system based on sensitivity analysis. Knowledge and Information Systems **60**(2), 837–878 (2019)
40. Yohannese, C.W., Li, T., Simfukwe, M., Khurshid, F.: Ensembles based combined learning for improved software fault prediction: A comparative study. In: 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pp. 1–6. IEEE (2017). https://doi.org/10.1109/iske.2017.8258836
41. Zuo, H., Zhang, G., Pedrycz, W., Behbood, V., Lu, J.: Fuzzy regression transfer learning in takagi–sugeno fuzzy models. IEEE Transactions on Fuzzy Systems **25**(6), 1795–1807 (2016). https://doi.org/10.1109/tfuzz.2016.2633376