Feature Selection for Fuzzy Models

Krijgsman, S.R.

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Industrial Engineering & Innovation Sciences

In partial fulfillment of the requirements for the degree of

Master of Science in Operations Management and Logistics

Feature Selection for Fuzzy Models

Steffi Krijgsman

1369555

Supervisors:

dr. M.S. Nobile

dr. L. Genga

i.r. C.E.M. Fuchs

Version 1.0

Eindhoven, 23 July 2021

Executive summary

Introduction and purpose

Nowadays, more and more decisions are taken using artificial intelligence (AI) systems. These systems often use high dimensional datasets, and the application possibilities are endless. When decisions are made based on AI-powered systems, it is important to understand the reasons for the decision, especially when the decision affects for instance someone's health or has a legal impact. Therefore, there is a high need for being able to understand the rationale of the AI method, in other words, there is a high need for interpretable AI. Some AI systems, for example rule-based models, are already easier to interpret because of their design. However, the complexity of these systems could still get in the way of the interpretability of the model. In order to maximize human understanding of the rationale of these AI systems, the complexity of these AI systems should be mitigated. Therefore, reducing the complexity and increasing interpretability of machine learning models is an increasingly important research topic.

For this project, the specific AI model that must be increased in interpretability, is a fuzzy model. Fuzzy models are rule-based models which use fuzzy logic, which is based on the observation that people make decisions based on vague, imprecise, non-numerical information (Zadeh, 1965). The rule based fuzzy models can also be called fuzzy inference systems. In general, rule-based models are already considered easier to interpret than other AI systems, because the rules indicate how certain variables influence the outcome of the model. However, implementing a rule-based model is no guarantee for creating an interpretable model. Another factor that influences the interpretability of a model is the number of input features. When the input data has high dimensionality, as a result, the rules become very long and difficult to interpret (Antonelli et al., 2016).

This is where feature selection comes in. Feature selection is a technique to reduce the dimensionality of the input data. The idea is to select a subset of input features by eliminating the irrelevant and redundant features from the dataset. The reduction of input data dimensionality could increase the fuzzy model's interpretability because the rules become shorter as the number of features decrease. To summarize, the main research question is formulated as following:

What is the most appropriate feature selection method to reduce complexity and improve interpretability of a fuzzy model?"

Fuzzy modeling

The Python library pyFUME is used for the implementation of fuzzy models during this project. pyFUME estimates fuzzy inference systems automatically from data and can be used to find the best fitting fuzzy inference system to describe some phenomenon. pyFUME was introduced in July 2020, and is a relatively new Python library (Fuchs et al., 2020). Currently, one of the limitations of pyFUME is the lack of good feature selection options to perform on the dataset. To evaluate the model created by pyFUME 10 fold-cross validation is used. The

mean absolute error (MAE) is obtained as the average MAE from the 10 folds. The average MAE is used as the model's performance indicator throughout this study.

Feature selection

Seven feature selection methods were tested in this research, which are briefly discussed below.

Mutual information (MI)

As defined in the research of Chen et al. (2018), mutual information (MI) measures the amount of information obtained about one random variable, through another random variable. MI can be applied for feature selection purposes, by measuring the MI between input variables and the output variable. The variables are ranked based on the MI, meaning this is a filter method. MI does not assume linearity or normality and can measure any kind of relationship between random variables.

F-score

A feature selection method based on F-test statistics is proposed by Elssied et al. (2014). The F-test is utilized to test the linear relation between the features and the target variable. This is done by calculating the Pearson's correlation between each input variable and the target variable. And then the converting the correlation values in to F-scores. High F-scores indicate a strong relation between two variables.

Fisher score

Fisher interclass separability method (FISM) was originally proposed by Abonyi et al. (2001), and is a filter feature selection method. The FISM is based on the concept of the Fisher score (Fisher, 1936). The working of the FISM is to find the features for which the between-cluster covariance is large, and for which the within-cluster covariance is small. If the between-cluster covariance is large, this means the clusters are far apart. If the within-cluster covariance is small, this means the cluster is compact. The features with the worst covariance values are eliminated from the dataset.

Genetic algorithm (GA)

The genetic algorithm (GA) was originally proposed by Holland et al. (1992) and was used for feature selection for the first time by Siedlecki & Sklansky (1989). The GA is an optimization algorithm which does not optimize a single solution, but it modifies a population of individuals simultaneously to find the optimal solution. The GA applies the principle of survival of the fittest. Each solution is represented by a binary vector, where the features with a 1 are selected and the features with a 0 are not selected. In each generation the best solutions of the population are selected based on the fitness value, which in this case is the MAE. The selected solutions are updated with crossover and mutation and continue to the next generation. This is a wrapper feature selection method because it involves the performance of the fuzzy model.

NSGA-II, NSGA-III, and SPEA2

NSGA-II, NSGA-III, and SPEA2 are multi-objective evolutionary algorithms (MOEAs) and are used to solve multi objective optimization problems. These algorithms are utilized for

feature selection to optimize the objectives: minimize the number of features and to minimize the MAE. NSGA-II, NSGA-III, and SPEA2 are wrapper methods since the performance of the fuzzy model is involved in the feature selection algorithm. The MOEAs work with the same principle of the GA. The difference is in the selection process at the end of every generation. Also, the MOEAs do not return a single optimal solution, but they identify a Pareto front of multiple optimal solutions. The difference between the NSGA-II, NSGA-III, and SPEA2 is in the selection process at the end of every generation.

Experimental set-up and data

The seven feature selection methods are tested on two datasets: an artificial dataset and the COVID-19 dataset. The purpose of the artificial dataset is to test whether the methods can select the relevant features from a simple dataset with linear relations between the features and the target variable. The COVID-19 dataset is retrieved from the "European registry of patients with COVID-19, cardio-vascular risk and complications", and is more complex than the artificial dataset. After preprocessing, the COVID-19 dataset consists of 39 features and one target variable: 'worst PaO2/FiO2 ratio', which is the worst measured value for PaO2/FiO2 during the hospitalization period of a patient.

Main conclusions

The seven feature selection methods are tested on the COVID-19 dataset and the least performing feature selection methods are the Fisher score and the GA. The MI and F-score feature selection methods already perform better than the GA and the Fisher score method. Because, both methods managed to rank the features of the COVID-19 dataset such that the most relevant features were selected. The best performing feature selection methods are the three multi-objective wrapper methods: NSGA-II, NSGA-III, and SPEA2. Since the multi-objective wrapper methods return multiple optimal solutions in a Pareto front, there is a great chance of returning at least one solution that supports the goal of this research. Therefore, making a comparison between the returned solutions of the filter methods and the wrapper methods is not really fair.

The recommended feature selection method to use in combination with pyFUME for the COVID-19 dataset is the NSGA-II, NSGA-III, or SPEA2. Since there was no significant difference between the performances of these methods, there is no recommendation for specifically one of them. The disadvantage of these methods is the long computational time, therefore, the second best method that is recommended is the MI method. This method can be used in situations where a long computational time is undesirable.

The feature selection methods aimed to select the most relevant features from the COVID-19 dataset. The three features that were selected by all multi-objective wrapper methods are 'Platelets_value', 'pO2', and 'pO2_FiO2_ratio'. It is recommended to keep at least these three features in the dataset to build the fuzzy model. Building a fuzzy model with only these three features causes a large reduction in model complexity, comparing to a fuzzy model with all 39 features. This reduction in model complexity contributes to better interpretability of the model.

Content

List of Figures

List of Tables

List of Abbreviations

ACO	Ant colony optimization
AI	Artificial intelligence
BPSO	Binary particle swarm optimization
CLI	Command line interface
CPU	Central processing unit
ENORA	Evolutionary non dominated sorting with radial slots based algorithm
FIS	Fuzzy inference system
FISM	Fisher interclass separability method
FST-PSO	Fuzzy self-tuning particle swarm optimization
FS	Feature selection
GA	Genetic algorithm
Gen	Number of generations
ICPSO	Integer and Categorical Particle Swarm Optimization
LDA	Linear Discriminant Analysis
MAE	Mean absolute error
MSE	Mean squared error
MOEA	Multi objective evolutionary algorithm
MUTPB	The probability that an offspring is produced by mutation
NSGA	Non-dominated sorting genetic algorithm
Pop	Population size
PSO	Particle swarm optimization
RMSE	Root mean square error
RST	Rough set theory
SFS	Sequential forward selection
SPEA	Strength Pareto evolutionary algorithm
SSH	Secure Shell
XCPB	The probability that an offspring is produced by crossover

1 Introduction

1.1 Problem formulation

Nowadays, more and more decisions are taken using artificial intelligence (AI) systems. These systems often use high dimensional datasets, and the application possibilities are endless. When decisions are made based on AI-powered systems, it is important to understand the reasons for the decision, especially when the decision affects for instance someone's health or has a legal impact. Therefore, there is a high need for being able to understand the rationale of the AI method, in other words, there is a high need for interpretable AI. Currently, many machine learning models are seen as black box models, because most humans do not understand what this model bases its predictions on. Stakeholders in AI are demanding more transparency and understanding of the rationale of the models, because these machine learning models are increasingly used for critical decision making. An example of these stakeholders are medical doctors. The regulations do not allow medical doctors to make automated decisions and they are required by regulations to be able to explain the rationale for their decisions. Therefore, when an AI system is utilized to support a certain decision (for example to make a prognosis), the medical doctor must understand and agree with the rationale of the AI system to be allowed to use the information provided by the AI system for the decision.

Some AI systems, for example rule-based models, are already easier to interpret because of their design. However, the complexity of these systems could still get in the way of the interpretability of the model. In order to maximize human understanding of the rationale of these AI systems, the complexity of these AI systems should be mitigated. Therefore, reducing the complexity and increasing interpretability of machine learning models without affecting the models performance negatively is an increasingly important research topic.

For this project, the specific AI system that must be increased in interpretability, is a fuzzy inference system. In this work, the terms 'fuzzy model' and 'fuzzy inference system' are used interchangeably and both refer to fuzzy rule-base systems. Fuzzy models are rule-based models which use fuzzy logic, which is based on the observation that people make decisions based on vague, imprecise, non-numerical information. Fuzzy logic was introduced by Zadeh (1965). In general, rule-based models are already considered easier to interpret than other AI systems, because the rules indicate how certain variables influence the outcome of the model. Fuzzy models are based on IF-THEN rules, which make the model very transparent about its rationale. Also, these rules are written in natural language, which makes it more understandable for humans. However, implementing a rule-based model is no guarantee for creating an interpretable model. Another factor that influences the interpretability of a model is the number of input features. When the input data has high dimensionality, as a result, the rules become very long and difficult to interpret (Antonelli et al., 2016). The problem of high dimensional data tends to grow since the data collected across fields is growing in dimensionality.

This is where feature selection comes in. Feature selection is a technique to reduce the dimensionality of the input data. The idea is to select a subset of input features by eliminating the irrelevant and redundant features from the dataset. The reduction of input data dimensionality could increase the fuzzy model's interpretability because the rules become shorter as the number of features decrease. There is a wide variety of feature selection algorithms suggested in the literature. There is no 'best' feature selection algorithm for all cases. Therefore, it is interesting to compare several feature selection algorithms, and see which algorithm perform well in combination with fuzzy models.

The Python libraries Simpful and pyFUME are used for the implementation of fuzzy models during this project. Simpful, introduced in June 2020, was designed to facilitate the definition, analysis and interpretation of fuzzy inference systems. Simpful allows to define fuzzy sets and fuzzy rules, and to perform fuzzy inference (Spolaor et al., 2020). Simpful is employed within pyFUME, meaning that pyFUME estimates Simpful models (i.e. fuzzy inference systems) automatically from data and can be used to find the best fitting fuzzy inference system to describe some phenomenon. pyFUME was introduced in July 2020, and is a relatively new Python library (Fuchs et al., 2020). Currently, one of the limitations of pyFUME is the lack of good feature selection options to perform on the dataset.

Which has led to the formulation of the objective of the dissertation, which is as follows: "Improve interpretability of fuzzy models (that use high dimensional data and automatically generated rules) by implementing the most appropriate feature selection method with pyFUME such that the complexity of the model is reduced and the performance of the model is maintained in order to increase user's understandability of the model's rationale." The following research question has been formulated which corresponds to the objective:

What is the most appropriate feature selection method to reduce complexity and improve interpretability of a fuzzy model?"

And the following sub research questions:

1. What are existing methods of feature selection that are appropriate for a fuzzy model?
2. What method is used to measure the model's interpretability and complexity?
3. What method is used to measure the model's performance?
4. How to implement the chosen feature selection method?
5. How to evaluate and compare the feature selection methods?

1.2 Research design

To provide structure to the research process, the engineering cycle methodology (Wieringa, 2014) is utilized as research method. The engineering cycle maps a problem-solving process consisting of five phases, shown in Figure 1.1. This method was chosen because it provides a clear structure which suits the purpose of this research very well. As described before, the purpose of this research is to determine the most appropriate feature selection method for fuzzy models to increase model interpretability. In the engineering cycle the feature selection method

will be the artifact which is designed, validated, implemented and evaluated. The engineering cycle methodology is used as guidance and inspiration for the research process. It is not used as a strict procedure which must be followed exactly. The five phases are described below.

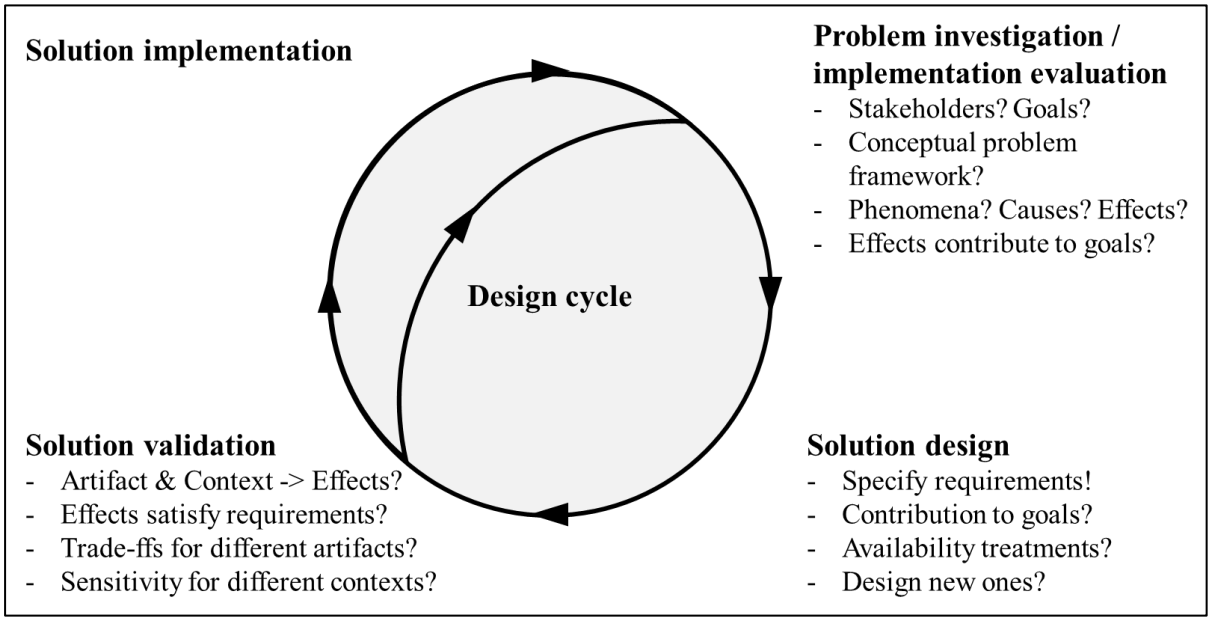


Figure 1.1: The engineering cycle adapted from Wieringa (2014)

Phase 1: Problem investigation

This phase serves as the orientational phase of the research. The objective of this phase is to define the problem, the research goals, the research context, and the define the phenomena that must be improved.

Phase 2: Solution design

In this phase the goal is to design one or more solutions (in other words artifacts). In the design science research field there are four types of artifacts defined: constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems) (Hevner et al., 2004). In this research the artifact is a feature selection method or algorithm, therefore, the artifact is referred to as a method after this section.

The first step for designing the artifact is to explore the existing feature selection methods through a literature review. The conducted literature review is focused on feature selection methods for fuzzy models which have to potential of increasing model interpretability. The next step is to evaluate the found feature selection methods and decide whether these methods could be used as artifacts. The found artifacts could be used as they are, or could be modified or combined with other artifacts. It is also possible to design totally new artifacts. However, the designed or selected artifacts must have the potential to contribute to the research objective.

Phase 3: Solution validation

The goal of this phase is to test whether the selected artifacts will contribute to the overall research objective. The performance of the artifact should be in line with the research objective, otherwise it does not make sense to continue to the next phase with the artifact. If the artifact is not valid, other artifacts should be considered before continuing the research process. Validating the selected artifact (i.e. feature selection method) is done by testing the method on an artificial dataset. This dataset was created to validate the feature selection methods before implementation. More details about this artificial dataset are given in section 3.5.1. The feature selection method is considered valid when it meets the requirements regarding model's interpretability and the model's performance. The specifics about the requirements are defined in section 3.5.1.

Phase 4: Artifact implementation

During this phase the valid artifacts are implemented. Therefore, validation should be performed before implementation. However, performing phase 3 and 4 in this order is not completely possible in this research. Validity can only be argued by testing the method on the artificial dataset, which is only possible by (partially) implementing the method. Therefore, phase 3 and phase 4 cannot be seen completely separate from each other.

The artifacts are implemented in Python with the support of several Python libraries. pyFUME is utilized in the implementation to test the model's performance and interpretability. However, please note that the artifact is not implemented within the pyFUME library, but only with the support of pyFUME. In the future, successful artifacts can be implemented within pyFUME, but this is out of scope for this project. When an artifact is considered valid, it moves on to next dataset: the COVID-19 dataset. This implementation is almost similar to the previous implementation for the artificial dataset. The difference is that the second dataset is much higher in dimensionality and demands much more computational time. Multi-processing is implemented to cope with the demand for high computational capacity.

Phase 5: Implementation evaluation

When the implementation is finished, the solution is evaluated. In this phase it becomes clear how successful each artifact has been. Artifacts are compared with each other and criticized. This phase leads to the conclusion and the answer to the main research question. In addition, during this phase new problems could come to light. Meaning this phase could possibly start a new iteration through the engineering cycle, which will result in recommendations for future research.

1.3 Reading guide

This report consists of five chapters that altogether represent the process of this project. The first chapter, the Introduction, provides an introduction to the research topic, the research question, and the research design.

In chapter 2, more background information about the research topic is given. This includes, information on fuzzy logic, fuzzy models, model interpretability, and feature selection. A literature study was conducted to find feature selection methods for fuzzy models. The results of this literature study are also shown in this chapter.

In the next chapter, chapter 3, all methods that are utilized throughout the process are described. These methods include: the feature selection methods that are tested, the methods and tools that are used for implementation, the methods for measuring model interpretability and model performance, and the experimental setup.

In chapter 4, the results are given. First the results of the feature selection methods on the artificial dataset are given. Then the results on the COVID-19 dataset are given, and the feature selection methods are compared.

In the final chapter, chapter 5, the discussion is given. The discussion consists of the answer to the research question, conclusions, and suggestions for future research.

A description of the Python scripts for the implementations and the files which store the results is given in Appendix A.

2 Background and related literature

2.1 Fuzzy logic

2.1.1 The idea of fuzzy logic

Classical logic is the science of formal principles of reasoning. Fuzzy logic a form of logic which is focused on the formal principles of approximate reasoning. Unlike classical logic, fuzzy logic aims to model the imprecise reasoning that is performed by humans to make rational decisions based on vague, imprecise, non-numerical information (Zadeh, 1988).

Classical logic cannot account for proportionate values for the truth, since it can only handle Boolean values for the truth. In fuzzy logic, the truth of the variables may be any real number in the interval [0,1]. Fuzzy logic has the ability to handle terms from natural language such as warm water, tall people, and fast cars. In these terms, the words warm, tall, and fast can represent a degree of truth about a variable. For instance, there is more than one truth about the warmth of water. Some people will consider water of 25 °C as warm, while other people will consider this water as cold. Therefore, there is no Boolean truth about the warmth of the water, since not all people have the same truth about this. In this case the temperature is called a linguistic variable.

2.1.2 Fuzzy sets

The concept of fuzzy sets was introduced by Zadeh (1965). A fuzzy set is a set of points where each point is associated with a real number in the interval of [0,1], which represents the membership of a point. This means it is possible to have a partial membership to the set. This is in contrast to crisp sets, where the membership of each point is defined by a 0 or a 1, meaning the point is either a member of the set or it is not. A visualization of this concept is shown in Figure 2.1. The concept of fuzzy sets can be applied to the example of warm water. The warmth water is not defined the same by everyone. If the temperature of the water is 0 °C most people will agree that the water is not warm (or cold). However, when the temperature of the water is 25 °C, some will say the water is warm and some will say it is cold. In a crisp set this will cause problems, because crisp sets require a Boolean and precise value for the membership value. However, in fuzzy logic the imprecise values can be handled, because the water of 25 °C can have for example a 0.8 membership in the set ‘warm’ and a 0.2 membership in the set ‘cold’.

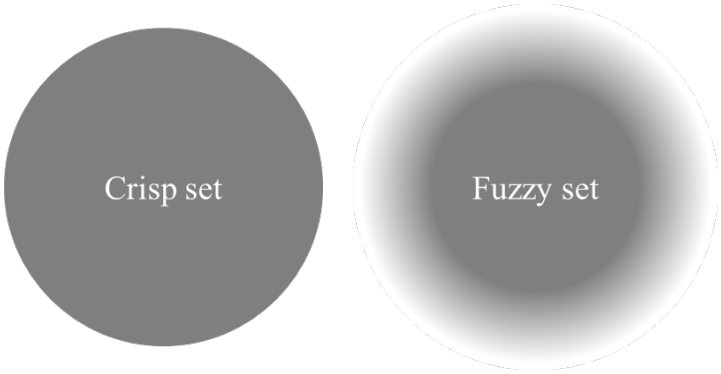


Figure 2.1: Visualization of a crisp set and a fuzzy set

2.1.3 Membership functions

The fuzzy sets associate each datapoint with a membership to a certain set. This membership is represented by a number in the interval [0,1]. A membership function helps define the membership value of a datapoint. Membership functions can have multiple shapes, two examples of common shapes are trapezoidal and Gaussian. A visual example of the trapezoidal and Gaussian membership functions are presented in Figure 2.2. However, in practice many more shapes can form a membership function.

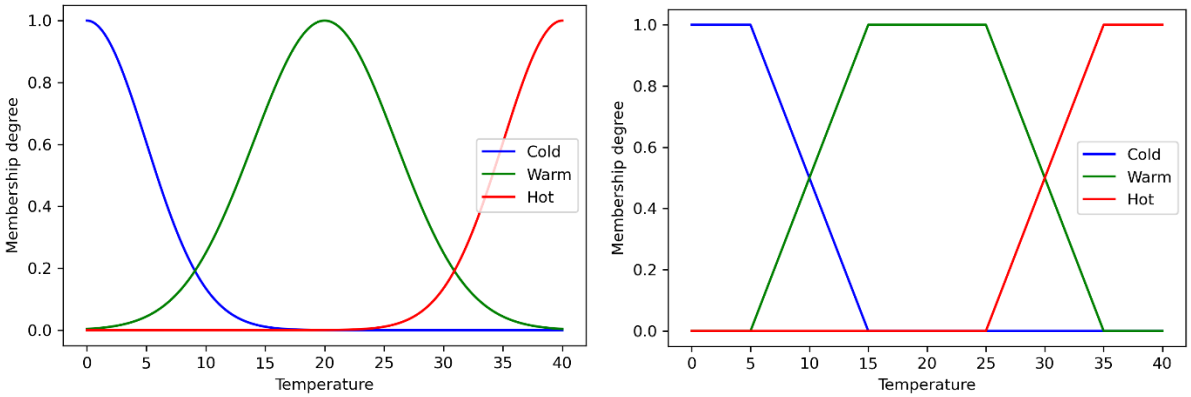


Figure 2.2: Membership function shapes (left: Gaussian shape, right: trapezoidal shape)

2.1.4 Fuzzy rules

When all datapoint are associated with a membership values to all defined sets, the next step is to define fuzzy rules. Generally, fuzzy systems use “if-then” rules to model the reasoning process of humans. When a human is reasoning, this often goas like this: ‘if I do this, than that will happen’ or ‘if the water of the sea is below 20 °C, then it will not be crowded at the beach’. Fuzzy systems aim to replicate this process with fuzzy rules which look like this: IF x is A THEN y is B. Here x and y are linguistic variables, and A and B are linguistic values that are

determined by the fuzzy sets and membership functions. An example of a fuzzy rule is: 'IF the water is warm THEN the beach is crowded'. In this example, the terms 'warm' and 'crowded' are the linguistic values. These linguistic values are chosen based on the membership values. For example, if the temperature of the water is 20 °C, this could be described with one of these linguistic values: 'cold', 'warm', or 'hot'. The linguistic value with the highest membership value, which is 'warm' according to Figure 2.2, is chosen to represent a temperature of 20 °C.

The premise in the example is 'the water is warm'. If the rules have more than one premise before the THEN operator, the fuzzy rules can also include logic operators such as AND, OR, and NOT. An example of a fuzzy rule with a logic operator is: 'IF the water is warm AND the weather is sunny THEN the beach is crowded'. If there is more than one premise, then these premises are called 'partial premises'.

2.1.5 Fuzzy inference systems

A fuzzy inference system (FIS) is a system that uses input data to derive output with the support of fuzzy rules. In general, the process of fuzzy inference can be described by three steps (Hong & Lee, 1996): 1) Fuzzify the input data, in other words, convert the input data to linguistic values with the support of the membership functions, 2) define the output groups by matching the linguistic values with the fuzzy rules, and 3) defuzzify the output groups to the final output. The structure of a FIS is visualized in Figure 2.3.

The purpose of the fuzzifier is to be the interface between the real world and the fuzzy system. In the fuzzification process, the input data is converted to linguistic values by matching the linguistic variables with the membership function. The input data points receive a membership value for each linguistic variable. The knowledge base consists of the fuzzy rules and the database that was conducted by the fuzzifier. The knowledge base holds information about the relation between the input and output. The inference engine is the reasoning mechanism of the fuzzy system. It performs the reasoning with the support of the knowledge base, and it produces the output. The defuzzifier converts the output of the inference engine to a crisp output, which can represent for example a final decision. The defuzzifier functions as an interface between the FIS and the real world.

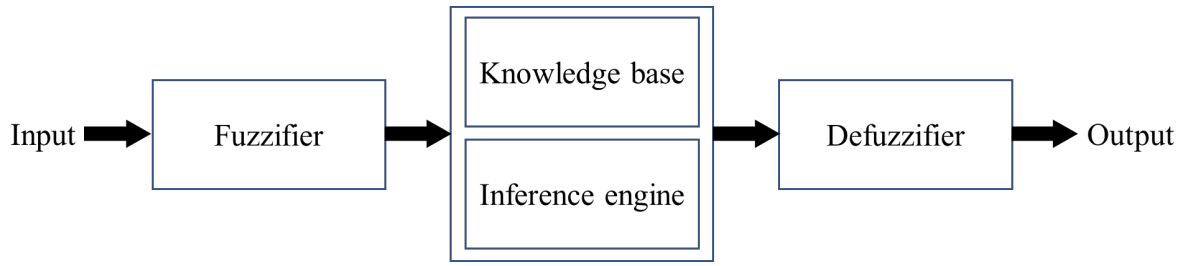


Figure 2.3: Structure of a Fuzzy Inference System

There are two main inference methods to use in the FIS, which are Mamdani method and the Takagi-Sugeno method. In this research, first order Takagi-Sugeno is applied, which is also the default setting in pyFUME. The fuzzy rules in Takagi-Sugeno inference are in the following format (Takagi & Sugeno, 1985):

$$IF\ x\ is\ A\ and\ y\ is\ B\ THEN\ Z\ =\ f(x,y)$$

where, A and B are the fuzzy sets and Z is a linear function of x and y. The obtained value of Z is the output of the model, thus defuzzification is not needed for this method.

2.1.6 Simpful and pyFUME

A FIS can be brought into practice with the support of Python libraries Simpful and pyFUME. Simpful was introduced in June 2020 and is therefore fairly new and up to date. The library was introduced to address the need for having a lightweight, open-source, Python library to support the creation of readable FISs. Simpful was designed to facilitate the definition, analysis and interpretation of FISs. Simpful allows to define fuzzy sets and fuzzy rules, and to perform fuzzy inference. Currently, Simpful supports the following processes: defining membership functions, defining fuzzy rules, and performing inference with Mamdani and Takagi-Sugeno-Kang methods (Spolaor et al., 2020).

In Simpful the user is required to define the fuzzy sets, linguistic variables, membership functions and fuzzy rules manually. If one desires to generate these automatically, one can utilized pyFUME. Simpful is employed within pyFUME, meaning that pyFUME estimates Simpful models (i.e. FISs) automatically from data and can be used to find the best fitting FIS to describe some phenomenon. pyFUME was introduced in July 2020, therefore this Python library is relatively new. pyFUME aims to create a first order Takagi-Sugeno-Kang fuzzy model from data. The user only has to provide the dataset and define the number of clusters (or number of rules) that should be identified in the data, and pyFUME will automatically generate a fuzzy model. If the user wishes to have more control, one can change the default settings fairly easily. For instance, enable k-fold cross validation in the testing phase. In addition, it is possible for users to model their own pipeline, however, this is more complicated than utilizing the option described before. Also, pyFUME has a feature to measure the performance of the generated fuzzy model with the following metrics: Root Mean Squared Error (RMSE), Mean Squared Error (MSE), or Mean Absolute Error (MAE) (Fuchs et al., 2020).

The function of pyFUME is to automatically generate and execute a FIS with the support of Simpful. Besides performing fuzzy inference, pyFUME also creates an output file with the code of the Simpful model that was created. This provides insight in the fuzzy sets, linguistic variables, membership functions and fuzzy rules that were generated by pyFUME.

2.2 Model interpretability

Model interpretability has many definitions that are used across fields. One of the definitions of model interpretability is how easy it is for humans to understand the processes the model uses to make its predictions. The demand for interpretable models is growing as more and more decisions are taken based on AI systems. The interpretability of a predictive model becomes important when there is a desire to understand the rationale of the model. Currently, many machine learning tools are seen as black box models, because most humans fail to understand the bases of the predictions of the model. Rules based models, such as fuzzy models, are already easier to interpret because of their design.

There are a few aspects of model interpretability that are receiving a lot of attention in the research field. One of the aspects is how to measure model interpretability. Measuring interpretability is not an easy task, because it depends on a lot of factors. Interpretability is subject to human understanding, which is subject to the persons background (for instance knowledge, intelligence and experience). Therefore, it is unknown what the best interpretability indices are. There have been a few developments of interpretability indices, however, there is no index that is widely accepted (Alonso et al., 2009). A few of the fuzzy model interpretability indices are the following.

The first and most simple measuring method of a fuzzy model's interpretability is the following. Three simple indices to measure the rule readability, which has an impact on model interpretability (Ishibuchi & Nojima, 2007):

- Number of rules.
- Total rule length: the number of premises in all rules are summed together.
- Average rule length: The total rule length is divided by the number of rules.

The second measuring method is an interpretability index based on the product of three terms, which is called the Nauck's index (Nauck, 2003). A model is less interpretable when the value of the index is closer to zero and more interpretable when the value of the index is closer to one. The Nauck's index is calculated as following:

- $Index_{Nauck} = Comp * Part * Cov$
- $Comp$ is the complexity of a model measured by the number of clusters identified in the data divided by the total number of premises.
- $Part$ is the average normalized partition index for all variables. The partition index is calculated as the inverse of the number of membership functions minus one for each input variable.
- Cov is the average normalized coverage degree of all variables. The degree of coverage of a variable is determined by the membership functions. It is calculated with the integrals of the membership functions over the domain of the variable divided by the integral of the domain.

Another interpretability measuring technique, inspired by Nauck's index, is to build a fuzzy system for measuring the interpretability of linguistic knowledge bases (Alonso et al., 2008).

This system is a hierarchical fuzzy system that takes six input variables and provides the interpretability index as output. The six input variables are:

- Total number of rules.
- Total number of premises.
- Number of rules which use one input.
- Number of rules which use two inputs.
- Number of rules which use three or more inputs.
- Total number of labels defined by input.

Most of these methods are very simple and do not cover every aspect of interpretability. However, measuring interpretability is an unsolved problem in the research field. For this research the length of the rules and the number of rules are considered when evaluation the model interpretability.

2.3 Feature selection

2.3.1 Purpose of feature selection

Feature selection is the process of reducing the datasets dimensionality by eliminating irrelevant, redundant, and noisy features. There are several reasons why feature selection could be desirable for building a predictive model.

The first reason is to reduce the model complexity to increase the model interpretability. Model interpretability is explained in more detail in section 2.2.

The second reason is that in general, feature selection could significantly improve the predictive model's performance. When the redundant and irrelevant features are removed from the dataset, the model only focuses on the relevant data. This reduces the risk of overfitting the model. Also, when feature selection is applied on the data, the predictive model cannot use the redundant and irrelevant features for its predictions. Meaning there is less misleading data involved in the prediction process, therefore, the model's accuracy improves.

The third reason is to reduce the computational time of the model. The computational time is, among other factors, depending on the dimensions of the input data. A high number of input features makes the model more complex, which results in longer computational times.

2.3.2 Filter, wrapper, and embedded approach

All feature selection methods use different approaches to select the best features from the dataset. The approaches can be categorized in the filter approach, the wrapper approach, and the embedded approach.

The filter approach uses independent measurement methods to score or rank the features without involving the learning algorithm. This approach has relatively fast computational time, because the performance of the prediction model is not considered. However, the filter approach does not consider relations between features, and can miss features that are relevant when they are combined with other features (Kumar & Minz, 2014).

The wrapper approach uses the performance of the learning algorithm to find the best subset of features. In most cases, this approach uses multiple iterations to evaluate the performance of the learning algorithm and to adjust the subset of features, until the optimal subset is found. Compared to the filter approach, the wrapper approach typically takes more computational time. But the advantage of the wrapper approach is that the relations between features are taken into account (Kumar & Minz, 2014).

The embedded approach does not separate the feature selection process from the learning algorithm (Lal et al., 2006). With the embedded approach, the features selection algorithm is integrated in the learning algorithm. The learning algorithm performs the feature selection method and the prediction process simultaneously.

2.3.3 Feature selection methods from literature

A systematic literature review was conducted to search for feature selection methods that are appropriate for fuzzy models. In total a number of 480 documents were retrieved from the databases IEEE Xplore, SpringerLink, and ScienceDirect. The total number of documents were filtered according to inclusion, exclusion, relevance and quality criteria, and are reduced to a number of 16 documents. The search terms and the mentioned criteria are given in Appendix B. The final selection of 16 documents are analyzed and interpreted, which resulted in a list of 11 feature selection methods for fuzzy models. These feature selection methods all have different characteristics: filter methods, wrapper methods, nature inspired methods, single objective, and multi objective methods. These feature selection methods are alle briefly described here.

Correlation

The research of Ghazavi & Liao (2008) implemented three different mutual correlation based feature selection methods. Each method calculates the mutual correlation for all possible pairs of features. If there appear high correlations for pairs of features, one of these features is removed to reduce redundancy in the dataset. This method is considered a filter method.

Soares et al. (2018) implemented a correlation based feature selection method with the Spearman coefficient (Spearman, 1904). They calculated a S score for every feature in the dataset with support of the Spearman coefficient. One feature at a time is removed based on the S score until the performance of the model reduces significantly. This method is considered a wrapper method because the feature subset is evaluated by the learning algorithm. Soares et al. (2018) also mentioned the Pearson correlation coefficient (Pearson & Henrici, 1896),

however, the authors prefer the Spearman coefficient because it does not assume linearity or normality (Bonett & Wright, 2000).

The research of Kerr-Wilson & Pedrycz (2020) implemented the Pearson correlation coefficient (Pearson & Henrici, 1896).They measured the correlation between the input features and the output feature. Only the input features with high correlation (according to a certain threshold) with the output feature were included in the model. This feature selection method is considered as a filter method.

This shows there are multiple ways to use correlation for feature selection. It can be applied to measure correlation between input features to reduce redundancy, or to measure the correlation with the output features to only include features that have impact on the output feature. One can choose between applying the Pearson's or Spearman's correlation coefficient. In addition, correlation based feature selection can be used as a filter method or as a wrapper method.

Sequential selection

Sequential feature selection is based on the idea of adding or removing features until a certain criteria is met. Forward selection is to begin with an empty set and to iteratively add features to the set until a termination criteria is met. Backward elimination is to begin with all the features in the set and the remove one feature at a time until the termination criteria is met. In the research by Lee et al. (2001), the backward elimination method was applied, and the termination criteria was a certain threshold of the classification error, which makes it a wrapper method.

Mutual information

As defined in the research of Chen et al. (2018), mutual information (MI) measures the amount of information obtained about one random variable, through another random variable. MI can be applied for feature selection purposes, by measuring the MI between input variables and the output variable. The variables are ranked based on the MI, meaning this is a filter method. MI does not assume linearity or normality and can measure any kind of relationship between random variables, which is an advantage compared to the Pearson's correlation coefficient. Chen et al. (2018) also suggest to use MI to find the optimal number of features by evaluating the performance of the model. This was done by following three steps: 1) rank the features based on the MI with the output variable, 2) build models with the first k features and measure the performance of the model (k is from 1 to the total number of features), and 3) find the optimal number of features by selecting the model with the best performance. Finally, Chen et al. (2018) concluded that this method of finding the optimal number of features should work for other filter feature selection methods as well.

Relief algorithm

In research of Ghazavi & Liao (2008) the Relief algorithm was used for feature selection. This algorithm was originally proposed by Kira & Rendell (1992) and works as following. The algorithm estimates the quality of a feature by selecting an instance and finding the two (or more when the dataset has more than two classes) nearest neighbors: one in the same class and one in another class. Based on the values of the nearest neighbors the feature receives a weight.

If the instance has a large difference with the same class neighbor, this means the feature separates two instances while they are in the same class, which is undesirable, and will result in a low weight. On the other hand, if the instance has a large difference with the other class neighbor, this is desirable, because the feature separates the instances from different classes, and will therefor result in a high weight.

The Relief algorithm has many variations, two of those variations are ReliefF and RReliefF (Robnik-Šikonja & Kononenko, 2003). The ReliefF algorithm, which was applied in the research of Tsang et al. (2007), can deal with multiclass problems. The RReliefF algorithm can be applied to continuous class or regression problems. The Relief algorithm, and all its variations, are filter feature selection methods.

Fisher interclass separability method and linear discriminant analysis

In the researches of Pulkkinen & Koivisto (2007) and Roubos et al. (2003) the Fisher interclass separability method (FISM) was used for feature selection. FISM was originally proposed by Abonyi et al. (2001), and is a filter feature selection method. The FISM is based on the concept of the Fisher score (Fisher, 1936). The working of the FISM is to find the features for which the between-cluster covariance is large, and for which the within-cluster covariance is small. If the between-cluster covariance is large, this means the clusters are far apart. If the within-cluster covariance is small, this means the cluster is compact. The features with the worst covariance values are removed from the dataset.

The Linear Discriminant Analysis (LDA) feature selection method, used in the research of Gayathri & Sumathi (2015), is very similar to the FISM. Both methods are a statistical method to find the best features to be able to separate the clusters. However, both LDA and FISM, make the assumption of linearity and LDA makes additional assumptions about the data (such as, normality and homoscedasticity), which makes them not appropriate to use for all datasets. LDA is also considered a filter method.

Genetic algorithm

In the research of Tiruneh & Robinson Fayek (2019) the genetic algorithm (GA) is used for feature selection. The GA was originally proposed by Holland et al. (1992) and was used for feature selection for the first time by Siedlecki & Sklansky (1989). As defined by Tiruneh & Robinson Fayek (2019), the GA is an optimization algorithm which does not optimize a single solution, but it modifies a population of individuals simultaneously to find the optimal solution. The GA applies the principle of survival of the fittest. Each solution is represented by a sequence of 1's and 0's, meaning, the features with a 1 are included in the set and the features with a 0 are excluded from the set. In each iteration (or generation) the best solutions of the population are selected based on the fitness value, which in this case is the classification error of the model. The selected solutions are updated with crossover and mutation and continue to the next generation. The GA is a single objective optimization algorithm, which only optimizes the performance of the model by finding the optimal set of features. However, if one wants to minimize the number of features in the model, the GA does not consider this. Therefore, in the research of Yu et al. (2002) was suggested to use a constraint for the model's performance while optimizing the number of features. The GA uses the performance of the model to find

the optimal set of features, therefore, this feature selection method is considered a wrapper method.

Particle swarm optimization

Particle swarm optimization (PSO) is a population-based optimization algorithm introduced by Kennedy & Eberhart (1995). The algorithm starts with a population of candidate solutions (or particles). These particles move around in the search space to find the global optimal solution. Each iteration the movement of the particles are updated based on their position and velocity. The algorithm takes into account the local optimum for each particle, but also considers the best known solution in the search space. The swarm of particles will gradually move to the global optimum. PSO can be used for feature selection, as was done in the research of Nasir et al. (2019). In this research each particle represents a set of n features, where n is the maximum number of available features. A particle is encoded as a vector of n real numbers. To determine which features are selected, a threshold is needed to compare the number in the vector with. Each iteration the solutions are evaluated with the classification error as fitness function, therefore, PSO is considered as a wrapper approach. The global optimal solution will be the solution with the best model performance. In the research of Nasir et al. (2019), the PSO algorithm was used for single-objective optimization. However, they suggested to build in a constraint for maximum number of features, to reduce the data dimensionality even more.

In 1995, PSO was originally designed to solve continuous optimization problems. Two years later, binary PSO (BPSO) was proposed to be able to solve discrete optimization problems (Kennedy & Eberhart, 1997), which is a great option for feature selection. With BPSO each particle represents a subset of the features, and the position of each particle is defined by a string of zeros and ones.

Ant colony optimization

Ant colony optimization (ACO) was originally introduced by Dorigo et al. (1991) and has been in development for many years. ACO is a nature inspired algorithm and is based on the behavior of certain ant species. These ants leave pheromone on a favorable path to guide the other members of the colony to follow the most optimal path to a target. ACO uses the same principle to find the optimal solution to an optimization problem (Dorigo et al., 2006).

One of ACO's applications is feature selection, this is called ant feature selection (AFS). AFS was originally proposed by Vieira et al. (2007). Another research by the same authors a few years later suggests an algorithm that uses two cooperative ant colonies to optimize two different objectives (Vieira et al., 2010). The two objectives are: minimizing the number of features and minimizing the classification error. The first ant colony finds the optimal number of features based on a ranking method. The second ant colony selects the optimal features based on the optimal number found by the first ant colony. The optimization process uses two pheromone metrices and two different heuristics. The two cooperative ACO algorithm aggregates the two objectives into one objective function, which will minimize the number of features plus the classification error. One of the objectives involves the performance of the model, therefore, AFS is considered as a wrapper method.

Multi objective evolutionary algorithms

Multi objective evolutionary algorithms (MOEAs) are used to solve multi objective optimization problems. Therefore, these algorithms could be utilized for feature selection when the objectives are to minimize the number of features and to minimize the classification error. Compared to GA's, MOEAs have the advantage of being able to optimize two objectives simultaneously, instead of having to use a constraint to take into account a second objective. However, MOEAs do not present a single optimal solution, but they identify a Pareto front of nondominated solutions. Jiménez et al. (2019) suggested two options to select the optimal solution from the Pareto front. When all the objective functions are linear, one can use a linear programming algorithm (for example the simplex method) to find the optimal solution. When the objective functions are non-linear, in principle any search algorithm to select the optimal solution. However, there is no guarantee that the search algorithm will find the optimal solution, but it will be an approximation.

In the research of Jiménez et al. (2019), two MOEAs were proposed for feature selection: NSGA-II and ENORA. NSGA-II is a non-dominated sorting genetic algorithm, which was introduced by Deb et al. (2002). ENORA is an evolutionary non dominated sorting with radial slots based algorithm, which was introduced by Jiménez et al. (2002). NSGA-II and ENORA both use binary tournament selection and rank the individuals in the population based on Pareto fronts and crowding. The difference between NSGA-II and ENORA is the approach for ranking the individuals in the population. The advantage of NSGA-II is its availability in many implementations. The advantage of ENORA is the good performance, which is in most cases higher than the performance of NSGA-II (Jiménez et al., 2019). MOEAs are categorized as wrapper feature selection methods.

Rough set theory

Rough set theory (RST), introduced by Pawlak (1982), is a good tool the cope with vagueness and uncertainty information to select the most relevant features (Reddy et al., 2020). RST for feature selection can only be utilized for discrete data. When the dataset contains continuous values, one can utilize fuzzy rough set theory for feature selection (Anaraki & Eftekhari, 2013). The difference between RST and fuzzy RST is that the fuzzy RST requires a discretization process, in which some kind of fuzzification technique is applied. The disadvantage of the discretization process is the risk of losing important information (Shen & Jensen, 2004). A popular method to apply RST is the QuickReduct algorithm, which is applied in the research of Shen & Jensen (2004). The QuickReduct algorithm is a greedy search algorithm using dependency. The algorithm starts with an empty set and each iteration adds the feature which the best increase in dependency score. The algorithm terminates when adding another feature does not improve the dependency score. The QuickReduct algorithm does not involve the learning algorithm in the process, so it is a filter method.

Stability selection

This method, stability selection, was applied in an experiment in the research of Amaral et al. (2020) and was inspired by the research of Meinshausen & Bühlmann (2010). This method must be utilized in combination with another feature selection method, and cannot be used

independently. Therefore, depending on the chosen feature selection method to combine with, the stability selection method could be a filter or a wrapper method. The idea is to apply another feature selection method (for example the GA) on distinct subsets of the data with diverse subsets of features. When this process is performed multiple times, one can evaluate the necessity of each feature. If the feature was selected from each subset, this means the feature is very relevant. When the feature was selected a couple of times, this means the feature is somewhat relevant, but is not essential in every subset. When the feature is selected zero times, this means the feature is irrelevant. The user of the stability selection method can determine a threshold for how many times a feature must be selected to be included in the final subset.

2.3.4 Feature selection in pyFUME

Currently, pyFUME provides two options for feature selection: sequential forward selection (SFS) and Integer and Categorical PSO (ICPSO). In the SFS method the features are selected sequentially and evaluated each iteration. This means pyFUME starts with an empty set and adds one feature at a time. For the first iteration, each feature is used to make a FIS separately and is evaluated on classification error. The feature with the smallest classification error is added to the set. In the next iteration, all features which are not in the selected set are used to make a FIS together with the features that are already in the selected set. The feature that caused the smallest classification error is added to the set. The iteration process is terminated when there is no feature that can be added to the set that lowers the classification error. This method is not very efficient, because pyFUME has to evaluate every feature in each iteration. If the dataset is of an substantial size, then this process will take a lot of computational time. In addition, this method is only focused on optimizing the performance of the model, and the total number of the features is not considered as an optimization objective.

The second option for feature selection within pyFUME is ICPSO. This method was proposed by Strasser et al. (2016) and is a form of PSO that can handle discrete optimization problems. In general, PSO algorithms require some hyperparameters to be set by the user. In pyFUME the user does not have to define the hyperparameters, because the standard PSO was replaced with fuzzy self-tuning PSO (FST-PSO). This algorithm determines the best hyperparameters automatically during the optimization process (Nobile et al., 2018). Within pyFUME a combination of ICPSO and FST-PSO is applied for feature selection, but is extended to also optimize the number of rules (or clusters). The dimension of the candidate solutions for the optimization process is extended with an extra variable, which represents the number of rules (or clusters). This feature selection method is a wrapper method, because the fitness evaluation of the candidate solutions is based on the models performance.

Regardless of the two feature selection methods that are already implemented in pyFUME, the developers believe there is room for improvement regarding feature selection. The main purpose of extending the feature selection options within pyFUME is minimizing the number of selected features to increase model interpretability, without neglecting the model's performance.

3 Methods

3.1 Build fuzzy model with pyFUME

The goal of this research is to find an appropriate feature selection method for fuzzy models. A fuzzy model must be implemented to be able to test the appropriateness of the feature selection methods. The fuzzy model is implemented with the support of pyFUME. To generate a fuzzy model, pyFUME requires two input arguments: the path to the dataset and the number of clusters. Besides these two arguments, pyFUME also takes other arguments, but these two arguments are mandatory. An example of another argument that can be used is k-fold cross validation. The number of clusters and k-fold cross validation are described in more detail in section 3.1.1 and section 3.1.2 respectively.

3.1.1 Number of clusters

The number of clusters has an impact on the performance of the model and on the interpretability of the model. For each cluster identified, pyFUME generates a fuzzy rule. Meaning, if one identifies three clusters in the dataset, pyFUME will generate three fuzzy rules. The number of rules affects model interpretability, since a high number of rules makes the model more complex, and can make it difficult to understand the rationale of the model.

While the model interpretability is compromised when the number of rules become too high, in some cases this could positively influence the performance of the model. The data can be described more precisely when there are more rules, which results in a higher performance. However, a high number of rules could also lead to overfitting on the training data, which could have a negative effect on the model's performance. The challenge is to find the optimal number of clusters by balancing model interpretability and performance.

Having domain knowledge about the dataset could be helpful for making the decision on the number of clusters. When it is known how many clusters exist in the dataset, this number could be used for the 'number of clusters' argument in pyFUME. When the number of clusters is a logical choice based on domain knowledge, the model becomes more interpretable, because each rule will have more meaning.

For this research the number of clusters is determined by using domain knowledge, testing the model's performance, and considering the model interpretability.

3.1.2 K-fold cross validation

k-fold cross validation is a procedure to test the model's performance on unseen data. By testing the model on a different test dataset for *k* times, the estimate of the model's performance

is less biased then only testing on one test dataset. The procedure of k-fold cross validation consists of four steps:

1) Shuffle the data randomly
2) Split the data into k groups
3) For each group:
    a. Use this group as the test dataset
    b. Use the remaining $k - 1$ groups for the training dataset
    c. Generate a model based on the training dataset and test the model on the test dataset
4) Summarize the model's performance by calculating the mean and standard deviation of the model's performance metrics

A k-fold cross validation function is available within pyFUME, with the default value $k = 10$. These default settings are used for this research.

3.2 Feature selection methods

There are multiple feature selection methods that are tested. Most of these feature selection methods were retrieved from the systematic literature review. However, not all feature selection methods that were found during the literature review are tested. Some feature selection methods use the same selection criteria. To avoid redundancy of feature selection methods, not all of them were selected to test. In addition, some feature selection methods were too complicated to implement because a lack of effective Python libraries. A few additional methods were found in literature later in the process. Each feature selection method that is tested in this research is explained below.

3.2.1 MI

The MI feature selection method is implemented with the 'scikit-learn' Python library (Pedregosa, F. et al., 2011). This library provides many machine learning applications for Python. The feature selection module from scikit learn (sklearn.feature_selection) is used to implement the MI feature selection method. The 'mutual_info_regression' function is used to perform feature selection. This function requires two arguments: a dataset consisting of all the input variables, and vector consisting of the target variable. The function calculates the MI between each input variable and the target variable. The output of the function is a list of scores, where the first score represents the MI between the first input variable and the target variable. The higher the MI value, the higher the dependency between the variables. The output of the function is used to select the best k features, where k could be any number between 1 and the total number of features in the dataset.

The MI feature selection method is tested for k from 1 to the total number of features in the dataset. The k selected features are tested with pyFUME, and the MAE is the performance indicator. The optimal value for k is the value which corresponds with the smallest MAE.

3.2.2 F-score

A feature selection method based on F-test statistics is proposed by Elssied et al. (2014). The F-test is utilized to test the linear relation between the features and the target variable. This feature selection method is implemented with the 'scikit-learn' Python library. The implementation is almost similar to the implementation of the MI feature selection method. The only difference is the function for feature selection. The feature selection method is defined as 'f_regression'. This function requires two arguments: the dataset with all the input variables, and the target variable. This function calculates the Pearson's correlation between each input variable and the target variable. Then the function converts the correlation values in to F-scores, which are then converted to p-values. The F-scores are calculated as following:

$$F_{score} = \frac{(\text{Pearson's correlation coefficient})^2}{(1 - (\text{Pearson's correlation coefficient})^2) * degrees\ of\ freedom}$$

High F-scores and low p-values indicate a strong relation between two variables. The F-scores are used to select the best k features, where k could be any integer between 1 and the total number of features in the dataset. The selection procedure of selecting the optimal value for k is similar to the procedure that is used for the MI feature selection method (section 3.2.1). The p-values are not utilized for this feature selection method since the p-values indicate the exact same features to select as the F-scores.

3.2.3 Fisher score

The Fisher score is a feature selection method that is based on similarity of values within a class and between classes. When the similarity within a class is high and the between classes similarity is low, then this is considered a good feature.

The Fisher score feature selection method is implemented with the 'fisher_score' function from the 'skfeature' Python library (Li et al., 2017). The function 'fisher_score' requires for two arguments, which are the input data, and the target variable data. The output of the function is a list containing a ranking of the features. The features are ranked based on their Fisher scores, where the highest Fisher score receives the highest ranking. The optimal number of features is not provided by this function. Therefore, to find the optimal number of features (k), the same approach is used as for the MI feature selection method (section 3.2.1).

The Rrelieff algorithm, which was discussed in section 2.3.3, is a feature selection method that is also based on difference between clusters and within clusters. The Rrelieff algorithm searches for the nearest hit (within the cluster) and nearest miss (outside the cluster), and ranks the features based on the within cluster and between cluster distances. The principles of the Rrelieff algorithm and Fisher score method are quite similar since they both analyze the within and between class distances. Therefore, only one of these feature selection methods is implemented. The Fisher score is implemented, because the availability of effective Python libraries for the Rrelieff algorithm is missing.

3.2.4 GA

A GA is a population based algorithm that is inspired by natural selection, crossover, and mutation in nature (Holland et al., 1992). The population consists of chromosomes, which are binary strings in this research. A process of evolution takes places, where the chromosomes (from now on called individuals) move in the direction of the optimal solution due to crossover, mutation, fitness evaluation, and selection.

The GA that is implemented is a single objective optimization algorithm. The objective is to minimize the MAE of the fuzzy model. The GA is implemented with the "Distributed evolutionary algorithms in python" (DEAP) library (Fortin et al., 2012). DEAP is an evolutionary computation framework that supports testing of evolutionary algorithms.

The first information that DEAP requires is the type of optimization problem. The type of problem can be defined in the creator module. In the creator module is defined that the problem is a single objective minimization problem. The creator module also allows for creating the 'individual' class.

After the type of problem is defined, DEAP requires to define the needed tools. The toolbox is a container in which all the tools are stored. The tools that are used are and stored in the toolbox:

- "Attr_bool". With this tool is defined that the individual consists of binary values.
- "Individual". With this tool the shape of the individual is defined. The individual is a vector of binary values with the length of the total number of features. In the binary string 1 represents that the feature was selected, and 0 represents that the features was not selected.
- "Evaluate". With this tool the fitness evaluation function (which calculates the fitness value of an individual) is defined. The fitness value is the MAE of the fuzzy model, obtained with pyFUME. The MAE is calculated for the features that are defined in the individual.
- "Mate". This tool allows to perform crossover after every generation. There are multiple crossover functions that can be applied with DEAP. The crossover function that is utilized for this research is the "cxTwoPoint" function. Two point crossover works as following. Two parents are selected from the population for the crossover operation. Two points in the binary string are selected randomly. The bits between the two points are swapped between the two parents. This creates two new individuals, called the offspring.
- "Mutate". This tool provides multiple functions to perform mutation. The function that is utilized for this research is the "mutFlipBit" function. With flip bit mutation, a predefined number of bits in the binary string are selected and are flipped. Meaning if the selected bit is a zero it becomes a one, and if the selected bit is a one it becomes a zero. This function requires the user to define the probability for every bit to be flipped. For this research was chosen to flip one bit on average, therefore the probability is set to $\frac{1}{length\ of\ individual}$. The length of the individual is equal to the total number of

features in the dataset. Therefore, in each mutation operation, one feature is added or one feature is removed from the individual (on average due to randomness).

- "Select". This tool allows the user to define the selection method. The selection procedure is a tournament with size three. This means, three individuals are randomly selected from the population. The best individual out this group of three is selected and continues to the next generation. This process is repeated until the desired size of population is reached.
- "Logbook". This tool was used to register the outcomes of the algorithm in each generation. This information is utilized to evaluate the convergence of the algorithm.
- "selBest". This tool was used to select the best individual in the population when the evolution process is finished.

There is a possibility that the GA creates an individual that is a string of zeros. This means the individual presents a solution with zero features. This is not a useful solution, and besides that, this individual cannot be evaluated since it is impossible to make a fuzzy model with zero features. An individual with only zeros can occur at three moments during the GA optimization.

The first moment is at the creation of the initial population. If there occurs an individual with zeros only, this individual is replaced with an individual that has a one at a random location in the binary string. This means the individual now represents one feature instead of zero.

The second and third moments are during crossover and mutation. At these moments, new individuals are created, which are exposed to the risk of becoming a string of zeros. To avoid this from happening, the decorator function from DEAP was utilized. The decorator function 'decorates' the crossover and mutation tools. After crossover and mutation, the decorator evaluates whether there exist individuals with only zeros in the offspring. If these individuals occur, they are replaced with an individual which has a one at a random location in the binary string.

Now the type of problem and the Toolbox is defined, the next step is to define the parameters. There are a few parameters required for the algorithm:

- Population size, which is the number of individuals that exist in the population (shortened as 'pop')
- The number of generations, which is the number of times the evolutionary process is performed (shortened as 'gen')
- The probability that an offspring is produced by crossover (shortened as 'XCPB')
- The probability that an offspring is produced by mutation (shortened as 'MUTPB')
- Mutation rate, which is the probability for every bit to be flipped

Ideally, one should perform a sensitivity analysis to find the optimal parameters for optimal solutions and convergence. However, finding the optimal parameters of evolutionary algorithms a research topic on its own. Therefore, in this research there is no sensitivity analysis performed to find the optimal parameters. However, there is experimented with a few different combinations of parameters to create the best model possible without performing a sensitivity analysis. The parameters of the initial GA are set to:

$$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$$

as these were commonly observed in the literature (for example in the research of Tiruneh & Robinson Fayek (2019)). A further description of the parameter tuning process is presented in section 4.3.2.

Once the parameters are defined, the next step is to program the evolutionary process. A simplified version of the pseudo code of the program is presented in Figure 3.1.

```
def main():
      pop = initial population
      for p in pop
            if sum(p) ==0
            replace p with an individual which has at least one 1
            end if
      end for loop
      evaluate all individuals in pop
      while gen < generation_size
            gen = gen + 1
            offspring = selection of pop with 'select' tool
            perform crossover with 'mate' tool
            perform mutation with 'mutate' tool
            evaluate new individuals in offspring
            pop = offspring
      end while loop
      return pop
```

Figure 3.1: Pseudo code of evolution process

3.2.5 NSGA-II

The NSGA-II is a variant of the GA that can optimize multiple objectives simultaneously. The selection procedure of the NSGA-II is very different from the selection procedure of the GA. The NSGA-II uses the following steps to perform the selection procedure:

1. The parent and offspring populations are combined to one population. If the size of both populations is n, then the total population has size $2n$.
2. The fast non-dominated sorting is applied. This technique finds the Pareto front of the population and calls this front 1. Then the second pareto front is identified by ignoring the solutions in front 1 and finding the Pareto front, called front 2. This process continuous until every solutions is part of a front.
3. The crowding distance is calculated for every solution. This is the distance between a solution and its neighbors in the same front. A large crowding distance indicates more diversity of solutions in the front.
4. Selecting the final population of size n involves both step 2 and step 3. First, front 1 is added to the final population. Adding fronts to the population continues until the population size exceeds n. Then, the front that caused to exceed the desired population size is evaluated on the crowding distance. The solutions in the front are ranked from greatest to smallest crowding distance. The solutions that have the highest ranking are added to the population until the population consists of n solutions.

The NSGA-II is implemented with the DEAP library. The same approach was used as for the GA. However, a few changes were made to the GA algorithm to make it a NSGA-II. The first change is in defining the type of optimization problem. The NSGA-II is used to optimize multiple objectives simultaneously, therefore, the type of problem is defined as a two-objective minimization problem. The type of individual class stays the same as in the GA implementation.

The tools that are changed for the NSGA-II are the following:

- "select". The selection method that is used for this algorithm is "selNSGA2", which performs the NSGA-II selection procedure.
- "Evaluate". The evaluation function evaluated the fitness value for two objectives. The first fitness function is the MAE of the fuzzy model that was created with the features that are presented in the individual. The second fitness function is the sum of the individual, which is the number of features the individual presents.
- "selBest". This tool is not used for the NSGA-II implementation.

And additional tool that is used for the NSGA-II is the following:

- "ParetoFront". Since the NSGA-II is optimizing two objectives, there is not one single optimal solution, but there is a Pareto front of solutions. The Pareto front tool registers the dominating solutions in every generation. The Pareto front is updated every generation, and adds new dominating solutions and removes solutions that are dominated. The tool returns the final Pareto front when the evolution process is finished.

The parameters for the initial NSGA-II are similar to the parameters of the GA:

$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$.

A further description of the parameter tuning process is presented in section 4.3.2.

3.2.6 NSGA-III

Just as the NSGA-II, the NSGA-III is a variant of the GA. The NSGA-III, which was introduced by Deb & Jain (2014), can optimize multiple objectives simultaneously. The selection procedure of the NSGA-III uses a combination of the fast non-dominated sorting technique and reference points to determine the diversity. Therefore, the first two steps of the selection procedure of the NSGA-II are also used within the NSGA-III. All the steps of the selection procedure of NSGA-III are defined below.

1. The parent and offspring populations are combined to one population. If the size of both populations is n, then the total population has size $2n$.
2. The fast non-dominated sorting is applied. This technique finds the Pareto front of the population and calls this front 1. Then the second pareto front is identified by ignoring the solutions in front 1 and finding the Pareto front, called front 2. This process continuous until every solutions is part of a front.
3. Define the location and number of reference points, which will form a hyperplane in the solution space. Defining the reference points is done manually.
4. Each reference point is converted to a reference line by joining the reference point with the origin.
5. The solution space is normalized.
6. Calculate the perpendicular distance between each reference line and each solution. The reference line with the shortest distance to the solution is from now on associated with that solution.
7. Selecting the final population of size n involves both step 2 and step 6. First, front 1 is added to the final population. Adding fronts to the population continues until the population size exceeds n. Then, the front that caused to exceed the desired population size is evaluated on the following.
    a. Count the number of solutions that are already selected that are associated with each reference point.
    b. Identify the reference point with the least number of associated solutions that are already selected.
    c. If there is already one or more solutions associated with this reference point in the final population: select one of the solutions in the front that is associated with this reference point at random. If this is true, continue to step 7f. If this is not true continue to step 7d.
    d. If there is no solution is the front that is associated with this reference point, the reference point is no longer considered in the selection process of the particular generation. If this is true, continue to step 7f. If this is not true continue to step 7e.
    e. If there is no solution already selected for the final population that is associated with this reference point, then select the associated solution in the front with the smallest distance to the reference point, and add this solution to the final population.
    f. Repeat step 7b until 7e until the final population consists of n solutions.

The implementation of the NSGA-III is almost similar to the implementation of the NSGA-II. The only difference is in the "select" tool. The selection method that is used for this implementation is "selNSGA3". This selection function requires an extra argument, which is the reference points. The reference points were defined with the "uniform_reference_points" tool. This tool requires two input arguments: the number of objectives, and the number of points. The number of objectives is two, and the number of points is set to 12. The number of points is determined with trial and error.

The parameters for the initial NSGA-III are similar to the parameters of the GA:

$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$.

A further description of the parameter tuning process is presented in section 4.3.2.

3.2.7 SPEA2

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) is a multi-objective optimization algorithm (Zitzler et al., 2001). The SPEA2 is a variant of the GA, and is almost similar to the NSGA-II. The only difference between these algorithms occurs in the method for selecting the population for the next generation. The population selection procedure of SPEA2 consists of four steps:

1. The initial population and an empty archive of size n are created.
2. Two fitness values are calculated for each solution. The first fitness value is the number of solutions the solution dominates (S). The second fitness value is a summation of the S value of all the solutions that dominate the solution (R). If the value for R is zero, then the solution is non-dominated.
3. Copy all nondominated solutions to the archive. If the archive exceeds n solutions, then the solutions with the smallest distance to other solutions are removed. If the archive contains less than n solutions, then the best solutions from the previous population and previous archive are added to the archive.
4. The archive becomes the new population in the next generation.

The SPEA2 is implemented with the support of the DEAP library. The implementation of the SPEA2 is almost similar to the implementation of the NSGA-II. The only difference is in the "select" tool. The selection method that is used for this implementation is "selSPEA2".

The parameters for the initial SPEA2 are similar to the parameters of the GA:

$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$.

A further description of the parameter tuning process is presented in section 4.3.2.

3.2.8 PSO

The PSO algorithm, or a variant of this algorithm, could be utilized for feature selection. A

form of this algorithm has already been implemented within pyFUME, namely, the ICPSO algorithm. Instead of implementing another variant of the PSO algorithm, the ICPSO algorithm is tested and compared to the other feature selection methods in this research. A description of this algorithm is given in section 2.3.4.

3.3 Measuring model's performance and interpretability

3.3.1 Performance measurement

Within pyFUME there are multiple options available for measuring the performance of the fuzzy model: MAE, RMSE, and MSE. For this project, the MAE was chosen for measuring the model's performance, because this metric has the most raw representation of the prediction error. The MAE is obtained by testing the model on unseen data (the test data) and calculating the mean of all absolute prediction errors.

Measuring the model's MAE is combined with 10-fold cross validation. Therefore, the mean of the MAE's of all folds are obtained. The purpose of the 10-fold cross validation is to reduce the bias of the estimate of the performance of the model. Meaning the value of the MAE will be more reliable than without cross validation.

To compare the performance of the feature selection methods, the independent two-sample t-test is used to determine whether two populations have equal means. The populations consist of 50 values for MAE, since the feature selection methods are tested with 50 repetitions. The feature selection methods are performed 50 times because there is randomness in the train/test split and in the wrapper methods. There are multiple conditions to conduct a two-sample t-test. Two of these conditions, that must be met before conducting the two-sample t-test, are normal distributed data and homogeneity of variance (Kim & Park, 2019). Testing the normality of the data is done by plotting the data in a histogram and observe whether there appears a 'bell curve', which is an indication for normal distributed data. Testing the homogeneity of variance is done with the F-test. The F-test is used to compare the two sample variances (Snedecor & Cochran, 1989). If these two conditions are not complied, then the t-test is not used. For the t-test and the F-test the null hypothesis of equal means and equal variances is rejected when the p-value is smaller than 0.05.

3.3.2 Interpretability measurement

During the process of choosing the number of clusters, the interpretability of the model was already considered. Because the number of clusters is the number of rules, which affects the interpretability. However, determining the best number of clusters is not the focus of this research. The focus of this research is to use feature selection to increase model interpretability. Therefore, measuring the interpretability is done by measuring the number of features in the model. The more features are selected, the longer the fuzzy rules are, which will reduce model interpretability. Therefore, the goal is to keep the number of features as low as possible.

Measuring the number of features is fairly easy. In this research, all feature selection methods are implemented such that the output represents the selected subset of features with a vector, consisting of binary values. In this vector, a one means the feature is selected and a zero means the feature is not selected. Therefore, taking the sum of this vector represents the number of features that were selected.

3.3.3 Pareto fronts

The multi-objective feature selection methods return a Pareto front of solutions instead of one single optimal solution. The Pareto front presents a set of solutions, where some solutions score very well on number of features, some solutions score very well on MAE, and some solutions are in between.

Hypervolume

To be able to compare the results of the multi-objective feature selection methods with each other, the hypervolume is calculated. The hypervolume represents the size of the space enclosed by the solutions in the Pareto front and the defined reference point (Zitzler & Thiele, 1998). When two Pareto fronts are compared, and one Pareto front dominates the other, the hypervolume of the dominating Pareto front is greater than the hypervolume of the other Pareto front. An example of the hypervolume concept is visually presented in in Figure 3.2. In the example, the second feature selection method (color red) performs better than the first feature selection method (color blue), because the surface area is bigger for the red Pareto front than for the blue Pareto front.

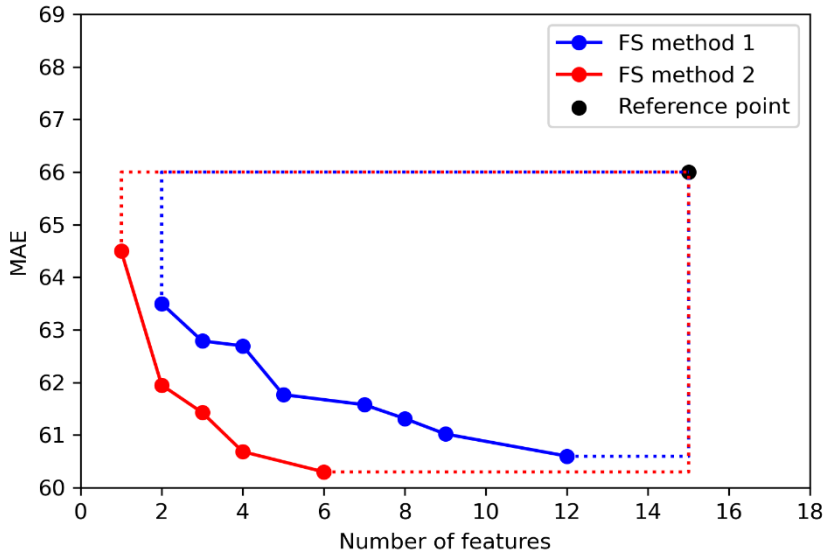


Figure 3.2: Hypervolume example

The user has to choose the reference point manually to be able to calculate the hypervolume. For a two-objective minimization problem, the x and y coordinates of the reference point must be equal or greater than the maximum values of x and y in the Pareto front. Otherwise, the

hypervolume calculation cannot take all solutions in the Pareto front into account. Usually, the reference point is either chosen as the worst values of each objective or as slightly worse values than the worst values of each objective (Zitzler et al., 2007).

For this research, the x-coordinate of the reference point is set to the total number of features in the dataset plus one. And the y-coordinate of the reference point will be set to a value which is greater than the highest value of MAE.

In this research, all multi-objective feature selection methods are implemented with Python library DEAP. DEAP also provides a function for calculating the hypervolume of a Pareto front. The function is called 'hypervolume'. This function requires two arguments: the datapoints in the Pareto front, and the reference point. Therefore, the hypervolume function of DEAP was utilized to calculate the hypervolume.

The hypervolumes of the feature selection methods could be compared with the support of the independent two-sample t-test. This statistical test is described in section 3.3.1.

Knee point

The multi-objective feature selection methods present a Pareto front of optimal solutions, while the single objective feature selection methods present a single optimal solution. To be able to compare the multi-objective feature selection methods with the single objective feature selection methods, the Pareto front has to be converted to a single solution. In this way, each feature selection method is represented by one solution.

The conversion of the Pareto front is performed by selecting a single solution from the Pareto front. Choosing a solution from the Pareto front is not an easy task, as all solutions are optimal and it is impossible to make a distinction about the optimality of the solutions. Although all solutions in the Pareto front are optimal, making a distinction about the trade-off between objectives is possible. The solution with the optimal trade-off between the two objectives is selected as the single solution that represents the Pareto front.

The method that is used to determine the solution with the optimal trade-off between the two objectives is to draw a line which passes through the two boundary points of the Pareto front and then to select the point with the greatest perpendicular distance to the line. This method is a distance-based knee detection method (Das, 1999). A visual example of this method is presented in Figure 3.3. In the visual example, the knee point (which has the greatest distance to the dashed line) is marked with a cross.

This method is implemented by drawing a line is through the boundary points of the Pareto front. The point with the greatest perpendicular distance to the line is selected as the knee point.

Please note that the knee point is an approximation of the optimal trade-off solution of the Pareto front. The solution is selected based on the maximum distance to the line, however, this does not mean this is the optimal trade-off in every situation. In this research the knee point is considered as the solution with the best trade-off between the two objectives in the Pareto front. However, this does not mean that the other solutions in the Pareto front should be ignored.

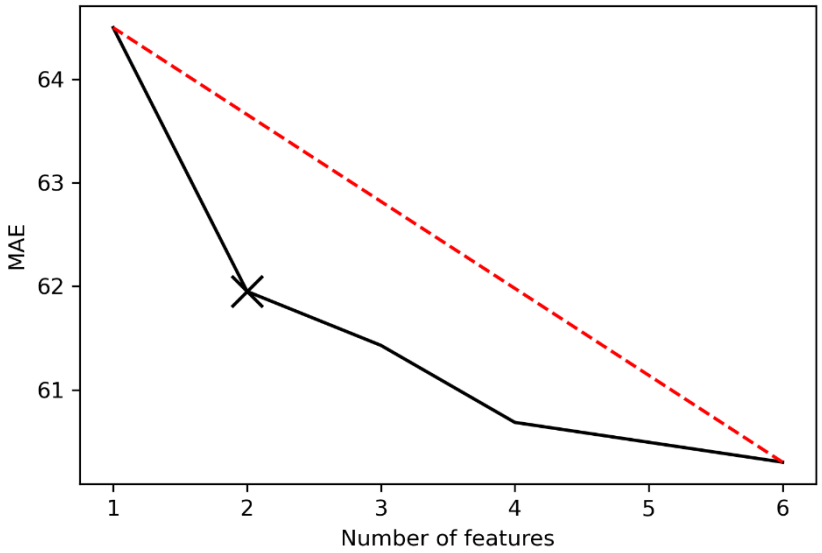


Figure 3.3: Knee point example

3.4 High performance computing

3.4.1 Computational time

The computational time of performing feature selection can vary from short to very long. The computational time is highly depending on the chosen feature selection method, the size of the input dataset, and the complexity of predictive model.

The dimensions of the input data have an impact on the computational time in the sense that the feature selection method has to evaluate all features, and consider a lot of feature combinations. Generally, a filter feature selection approach has a shorter computational time than a wrapper feature selection approach. A wrapper approach evaluates the model's performance in each iteration. To obtain the total computational time of a wrapper feature selection process, the computational time of evaluating a model's performance is multiplied by the number of evaluations (or iterations), which can result in long computational times.

Long computational times are especially common in evolutionary algorithms. Evolutionary algorithms start with an initial population of solutions. This population is updated in each generation based on the fitness evaluations to get closer to the optimal solution. The size of the population definitely has an impact on the computational time. If the size of the population is 100, which is a very common setting, then there are 100 fitness evaluations per generation. If the number of generations is also 100, which is also a very common setting, there are 100*100=10,000 fitness evaluations performed. When the computational time is 1 second, this means the total computational time is 10,000 seconds. However, in pyFUME the computational time of a fitness evaluation for a large dataset will take much longer than 1 second. A fitness

evaluation with pyFUME with all features of the COVID-19 dataset takes approximately 40 seconds.

The sort of predictive model can also influence the computational time. For this study, the only predictive model that is considered is a fuzzy model implemented with pyFUME. However, there is a distinction between predictive models, where some have naturally longer computational times than others.

Another factor that has an impact on the computational time is k-fold cross validation. For instance, if 10-fold cross validation is applied, the computational time will be 10 times longer for a performance evaluation than without cross validation.

3.4.2 Multiprocessing

Multiprocessing means to use more than one central processing unit (CPU) simultaneously to perform a computing task. When there are two or more CPUs available within a computer system, the computational work can be split across the processors (Rajagopal, 1999). Multiprocessing can help to save on computational time, because tasks can be performed in parallel.

Python provides a multiprocessing package that allows the programmer to leverage multiple CPUs on a machine, this package is called `multiprocessing`. One of the features that the `multiprocessing` package offers is the Pool object. The Pool object allows the programmer to execute certain computing tasks in parallel (Hunt, 2019).

```
from multiprocessing import Pool
def fitness_eval(individual):
    model = pyFUME_make_model(individual)
    MAE = model.calculate_error(method="MAE")
    return MAE
if __name__ == '__main__':
    with Pool(5) as p:
        print(p.map(fitness_eval, [ind_1, ind_2, ind_3]))
```

Figure 3.4: Example code for multiprocessing with Pool object in Python

The Pool object in the `multiprocessing` package is a perfect tool to perform fitness evaluations in parallel. For instance, when an evolutionary algorithm with population size 100 is used to solve an optimization problem, and there are 10 CPUs available. The Pool object can

distribute the fitness evaluations over these 10 CPUs, which allows to process the fitness evaluations in parallel. This will shorten the computational time with factor 10.

A basic demonstration is given on how the Pool object can be utilized for parallelizing execution of certain computations in Figure 3.4.

3.4.3 Supercomputer: Cartesius

Cartesius is a Dutch supercomputer for high performance computing, offered by the company SURF. SURF is a Dutch company which offers ICT facilities and services to education and research institutions. A few examples of the services they offer are network connectivity, security, data services, and high performance computing services. One of the high performance computing services they offer is the Cartesius system (*Services Offered by SURF | SURF.Nl*, n.d.).

The Cartesius system has a large number of processors and a fast internal network which results a high speed computer system. The system is mainly used for large scale experiments, for instance simulations and AI modelling. The system offers a high supply of memory, and it provides options for communication between various processors (*Cartesius | Userinfo.Surfsara.Nl*, n.d.). The Cartesius system was used to perform all feature selection methods for this research.

Performing computations on the Cartesius system is not as simple as performing computations on for example a laptop. There has to be some level of understanding about accessing the system and communicating with the system. Access to the Cartesius system is provided through a command line interface (CLI) with a Secure Shell (SSH). Connecting to the system goes through a terminal. The terminal that was used, and works well for Windows computers, is PuTTY. The communication with the system happens through a CLI, which responds to Linux command lines. In most cases, running a computation on Cartesius requires transferring files. To make this possible, the WinSCP software (which only works for Windows systems) was used to transfer files from the computer to the Cartesius system (*Getting Started with Cartesius | Userinfo.Surfsara.Nl*, n.d.).

To be able to use the computational capacity of Cartesius, one has to submit a job to the scheduler. The scheduler used for the Cartesius system is called Slurm. Slurm is an open source job scheduling system for large and small Linux clusters. This system manages all the submitted jobs and fairly distributes the computational capacity over the submitted jobs (*Slurm Workload Manager - Overview*, n.d.). It is important to be precise with requesting resources, because large resource requests will be queued longer than small resource requests. For example, when a job requires 10 computing nodes and a computational time of 24 hours, this job will most likely be queued for a longer time than a job which only requests one computing node for a duration of 10 minutes. Therefore, to be the most efficient, being precise in resource requests will save on queueing time.

To submit a job to Slurm, one has to define a job script. The job script usually contains information about the computing tasks and the required resources. Examples of the resources that can be defined are the required running time, number of computing nodes, and type of nodes. The job script should also specify the Python files that need to be executed, and the location of these files. Basic information about the job, such as job name, and name of the output file can also be included in the job script. Figure 3.5 shows an example of a job script that was used to execute a GA.

`#!/bin/bash`	Specifies which shell should execute the job
`#SBATCH --job-name=GA_1`	Specifies job name
`#SBATCH -n 1`	Request for one computing node
`#SBATCH -t 3:00:00`	Request for 3 hours of computational time
`#SBATCH --output=%x_%j.out`	Specifies output file name (Job name and ID)
`module load 2020`	Load modules that are needed for computing task
`module load Python`	
`cd $HOME/GA_1`	Define location/directory of files
`python GA.py`	Specify Python file that is requested to be executed

Figure 3.5: Job script and description

After the job script is submitted to Slurm, the job will be scheduled. The status of the job is 'pending' when the job is queued. When the Cartesius starts executing the job, the status becomes 'running'. When Cartesius is finished with the job, the user can collect the output files. It is also possible to submit multiple jobs to Slurm simultaneously. The Cartesius system will execute the jobs in parallel.

3.5 Experimental setup

3.5.1 Test on artificial dataset

The first experiment is applied on the artificial dataset.

Artificial dataset description

The artificial dataset was created by C.E.M. Fuchs with the purpose of testing feature selection methods before moving on to a more complex dataset. The advantage of testing on an artificial

dataset is that the relevant features are known in advance. Which makes it easy to evaluate whether the feature selection method selected the most relevant features.

The dataset consists of eleven variables; ten input variables ($x_1, x_2, x_3, \dots, x_{10}$) and one target variable y. Each variable has 1000 datapoints. The relevant input variables are x_1 and x_2, which are related with the target variable y. The other input variables were not programmed to be related to the target variable and are therefore irrelevant.

The sets of datapoints of x_1 , x_2, and y are divided into six subsets: $x_{1,1}$, $x_{1,2}$, $x_{2,1}$, $x_{2,2}$, y_1, and y_2, each containing 500 datapoints. The datapoints for all input variables are created with a normal distribution, all distributions are shown in Table 3.1. To create a relation between x_1 and y and between x_2 and y, the following equations were used to create the datapoints of the target variable:

$$y_1 = 5 * x_{1,1} + 2 * x_{2,1} \tag{1}$$

$$y_2 = 2 * x_{1,2} + 5 * x_{2,2} \tag{2}$$

The final step was to merge the subsets of datapoints by adding the first and second subset together:

$$x_1 = [x_{1,1}, x_{1,2}] \tag{3}$$

$$x_2 = [x_{2,1}, x_{2,2}] \tag{4}$$

$$y = [y_1, y_2] \tag{5}$$

Table 3.1: Normal distributions for artificial dataset

Variable	Mean (μ)	Standard deviation (σ)
$x_{1,1}$	5	2
$x_{1,2}$	15	2
$x_{2,1}$	5	2
$x_{2,2}$	15	2
x_3	7	3
x_4	8	2
x_5	9	1
x_6	6	2
x_7	10	4
x_8	11	1
x_9	5	4
x_{10}	3	4

Process

First the seven feature selection methods are tested on the artificial dataset. The number of clusters has to be determined to evaluate the performance of the fuzzy model. Two clusters can be identified in the artificial dataset, since these are defined by equations 1 and 2. The number of cluster can also be determined by evaluating the scatter plots in Appendix C. These scatter plots show a clear distinction between the two clusters.

Then the feature selection methods are performed. The FS methods are evaluated based on the number of features that were selected and on the MAE of the fuzzy model. The FS methods are also evaluated on which features are selected. In the artificial dataset it is known that x_1 and x_2 are the only relevant features. The feature selection methods that succeeded in selecting the relevant features are considered in the experiment on the next dataset.

3.5.2 Test on 'real world' dataset (COVID-19 dataset)

This experiment is performed on a 'real world' dataset, namely the COVID-19 dataset. This dataset is more complex than the artificial dataset. A complex dataset makes it more difficult to find the relevant features. Therefore, this dataset will provide a better test environment to observe the difference of performance between the feature selection methods.

COVID-19 dataset description

The 'real world' dataset that is used for this research is the COVID-19 dataset. This dataset is more complex than the artificial dataset, which is better for evaluating the feature selection methods. The COVID-19 dataset consists of data that was retrieved from the "European registry of patients with COVID-19, cardio-vascular risk and complications". This registry was created by the Italian Cardiology Network as a response to the pandemic due to the new coronavirus. The new coronavirus will from now on be referred to as COVID-19. The registry was created to investigate which variables influence the severity of a COVID-19 infection. When the COVID-19 patients arrives at the hospital, a lot of information is collected and registered about this patient. Also, during the hospitalization period, information is registered about the state of the patient every couple of days. COVID-19 can cause Acute Respiratory Distress Syndrome (ARDS). ARDS is a life-threatening syndrome that allows fluid to leak into the lungs. This makes breathing very difficult and there will arise a lack of oxygen in the patient's body (*Acute Respiratory Distress Syndrome (ARDS)*, n.d.). The severity of the ARDS is defined by the ratio of the partial pressure of oxygen in the patient's arterial blood (PaO2) to the fraction of oxygen in the inspired air (FiO2) on 5 cm of continuous positive airway pressure. The severity of the ARDS is defined by three categories (The ARDS Definition Task Force, 2012), these categories are presented in Table 3.2. The PaO2/FiO2 ratio is measured throughout the patient's hospitalization. In the end of the hospitalization, the worst PaO2/FiO2 ratio is registered. The "worst PaO2/FiO2 ratio" is the target variable in the dataset. To goal is to predict the worst PaO2/FiO2 ratio at the time of the patient's arrival at the hospital. This prediction is beneficial, in the sense that the medical staff could make decisions about the treatment and resources while having knowledge about the future state of the patient.

Table 3.2: ARDS severity categories (The ARDS Definition Task Force, 2012)

ARDS severity	PaO2/FiO2 value
Mild	200-300
Moderate	100-199
Severe	≤ 100

The original dataset contains the information of 291 records of hospitalized patients, and each record consists of 1212 variables. The records that miss a value for the target variable were removed from the dataset, which reduced that dataset to 228 patients. The following six preprocessing steps were performed on the dataset to obtain the final dataset. These steps were performed by Fuchs et al. (2021, under review).

1. All the technical columns related to information about whether a certain variable was collected or not were removed from the dataset. After this step, the dataset consists of 964 variables.
2. A new variable "age" was created from birth and admission dates, and a new variable "BMI" was calculated according to the patient's weight and height. This increases the number of variables to 966.
3. The model is trained according to the values measured at admission, therefore all columns from the dataset containing the minimum or maximum values measured during the hospitalization were removed from the dataset, with the exception of the PaO2/ FiO2 ratio. This step reduced the number of variables to 930.
4. pyFUME currently does not support categorical or qualitative variables. Hence, all non-numeric variables from the dataset were filtered out. This step reduced the number of variables to 665.
5. All variables measured after the admission were filtered out, because they cannot be considered for the model. The number of variables was reduced to 657.
6. Many fields are empty or filled with 'Not a Number' values. It was decided to keep only the variables whose values are ≥ 75% non-empty, which corresponds to 39 independent variables + 1 target variable. The empty fields in the remaining data were filled with an imputation technique. The final list of 39 features is presented in Appendix D.

Process

The experiment on the COVID-19 dataset consists of three stages. In the first stage the number of clusters is determined. The process of determining the number of clusters for the COVID-19 dataset is not as easy as for the artificial dataset. Scatter plots of all the input features and the target variable are presented in Appendix E. These scatter plots do not show a distinction between the clusters. Meaning that these scatter plots cannot support the decision on how many clusters should be identified in the data. Therefore, to determine the number of clusters, the performance of the feature selection methods with different number of clusters are evaluated.

The decision on the number of clusters is supported by the consideration of model performance and model interpretability.

The second stage of the experiment is parameter tuning. All feature selection methods require the user to define one or more parameters. In this stage the parameters with the best performance are selected. Depending on the sort of feature selection method, the performance of the model is evaluated based on the MAE or on the convergence.

The third and final stage of the experiment consists of the final evaluation of the feature selection methods. The feature selection methods are compared with each other based on model performance and model interpretability. The results of the feature selection methods are not constant due to randomness. Therefore, each feature selection method is performed 50 times, to be able to make a fair comparison between them.

4 Results

4.1 Results of experiment on artificial dataset

Seven feature selection methods were tested on the artificial dataset. For the filter methods (MI, F-score, and Fisher) the number of features to be selected is set to 2, since it is known that there are only two relevant features in the dataset. The results of the feature selection methods are presented in Table 4.1. The GA and all the filter methods succeeded in selecting the two relevant features from the dataset. All multi-objective feature selection methods (NSGA-II, NSGA-III, and SPEA2) achieve the same Pareto front of solutions. These Pareto fronts all include the solution that was expected, namely x_1 and x_2. It is concluded that all the seven feature selection methods performed sufficiently to be considered for the COVID-19 dataset.

Table 4.1: Results of feature selection on artificial dataset

Feature selection method	Selected features
MI	x_1, x_2
F-score	x_1, x_2
Fisher score	x_1, x_2
GA	x_1, x_2 with MAE 1.96
NSGA-II	x_1, x_2 with MAE 1.96 x_1 with MAE 7.66
NSGA-III	x_1, x_2 with MAE 1.96 x_1 with MAE 7.66
SPEA2	x_1, x_2 with MAE 1.96 x_1 with MAE 7.66

4.2 Number of clusters for COVID data

In this case, finding the best number of clusters is a subjective task. This is done by balancing model interpretability and performance. The number of clusters is determined by evaluating the results of fuzzy models and feature selection methods on multiple numbers of clusters. The number of clusters that are tested are 2, 3, 4, and 5. The tested models are presented in Table 4.2.

When there is no feature selection applied, the higher number of clusters, the worse the model performs. This is visible in Figure 4.1. The explanation for this could be that the model already struggles with the high dimensional data, and adding extra clusters only complicates the model. Also, identifying too many clusters in the data could lead to overfitting. In this case, adding complexity to the model (i.e. increasing the number of cluster) has a negative impact on the model's performance.

Table 4.2: Tested models for determination of number of clusters

FS method	Parameters	Number of repetitions	Presented in
No feature selection	-	50	Figure 4.1
NSGA-II	$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$	1	Appendix F Table 4.3
NSGA-III	$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$	1	Figure 4.4 Table 4.3
SPEA2	$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$	1	Appendix F Table 4.3
GA	$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$	1	Table 4.4
MI	$k = [1,2,3,\ldots,39]$	30	Figure 4.2
F-score	$k = [1,2,3,\ldots,39]$	30	Figure 4.3
Fisher score	$k = [1,2,3,\ldots,39]$	30	Appendix F

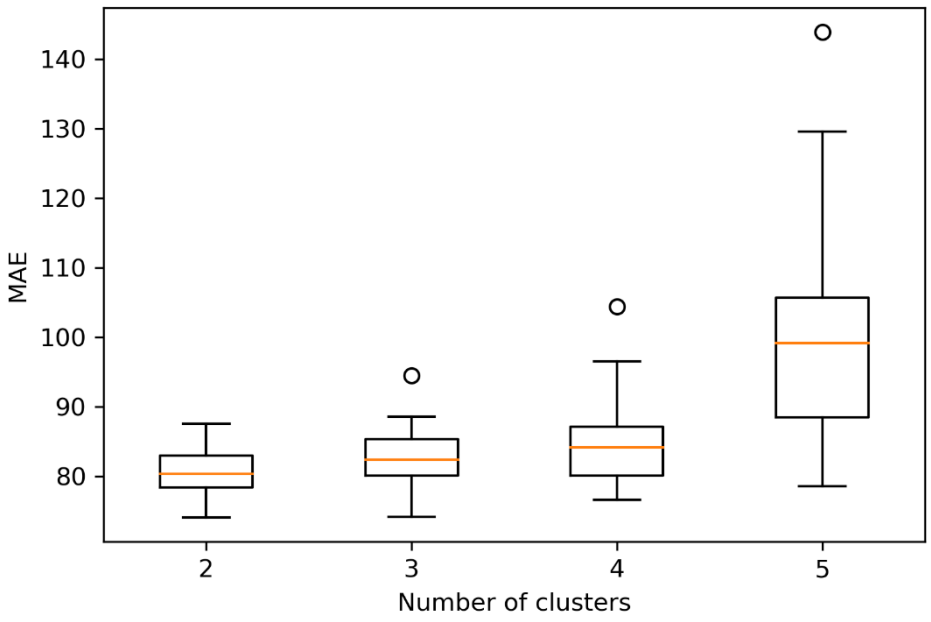


Figure 4.1: Model performance without feature selection

When the feature selection methods MI or F-score are applied, a higher number of clusters leads to better model performance when the number of features is below approximately 15. This is visible in Figure 4.2 and Figure 4.3. This means, when the data dimensionality is reduced to 15 features or less, the model responds well to a higher number of clusters in terms of performance. A similar observation is made for the other feature selection methods (NSGA-II, NSGA-III, SPEA2, and GA). For these feature selection methods is observed that the higher number of clusters leads to better performance. The results of the NSGA-III clearly show an performance gain for every cluster added to the model (Figure 4.4). The four Pareto fronts of the NSGA-III (and NSGA-II and SPEA2) are compared by evaluating the hypervolume. The greater the hypervolume the more ideal the pareto front. The hypervolumes of the Pareto fronts of all three multi-objective feature selection methods are presented in Table 4.3. The results of the GA lead to the same conclusion: a higher number of clusters results in a higher performance. The results of the GA are presented in Table 4.4. The takeaway from this is the optimal number of cluster is 5 for all seven feature selection methods, when the objective is to find the highest performance model. The graphic results of the feature selection methods that are not presented in this section (Fisher score, NSGA-II and SPEA2) are given in Appendix F.

However, the interpretability of the model should not be neglected by choosing for the best performing model. The number of clusters is equal to the number of rules in the fuzzy model. And the number of rules has an impact on model interpretability, where less rules lead to higher interpretability.

Adding one cluster to the model results in a performance gain of roughly 1% or 2%. Meaning, there is only a small performance gain for every cluster added to the model. Keeping in mind that minimizing the number of clusters will help with model interpretability, the number of clusters is for this research and this dataset is set to 3.

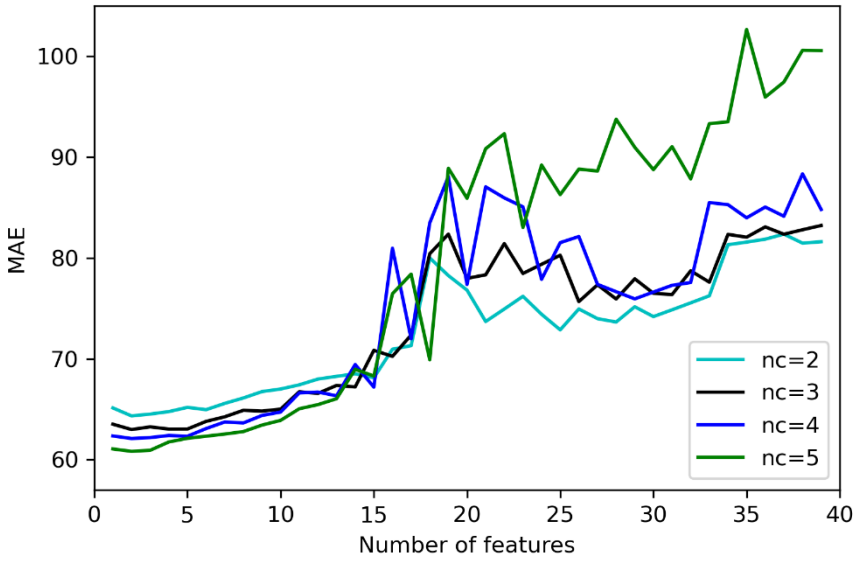


Figure 4.2: MI results for 2 to 5 clusters

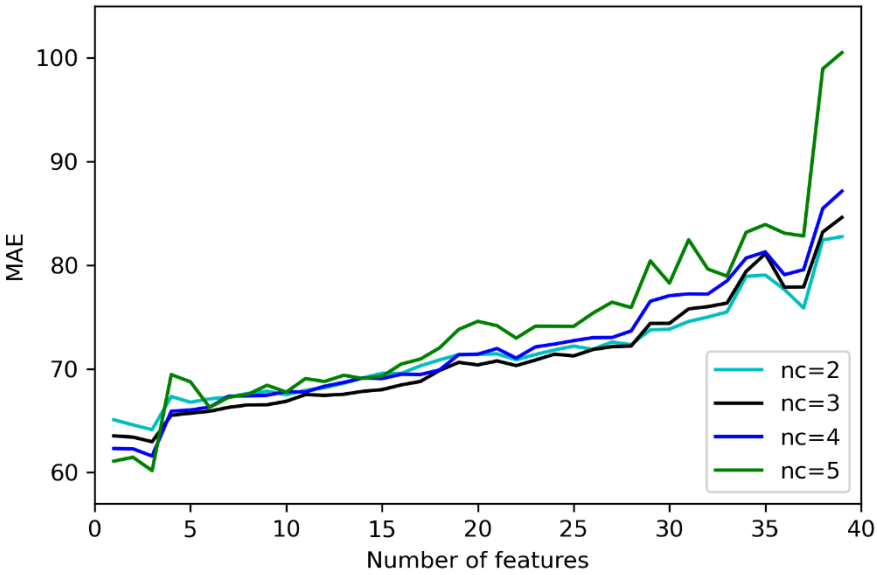


Figure 4.3: F-score results for 2 to 5 clusters

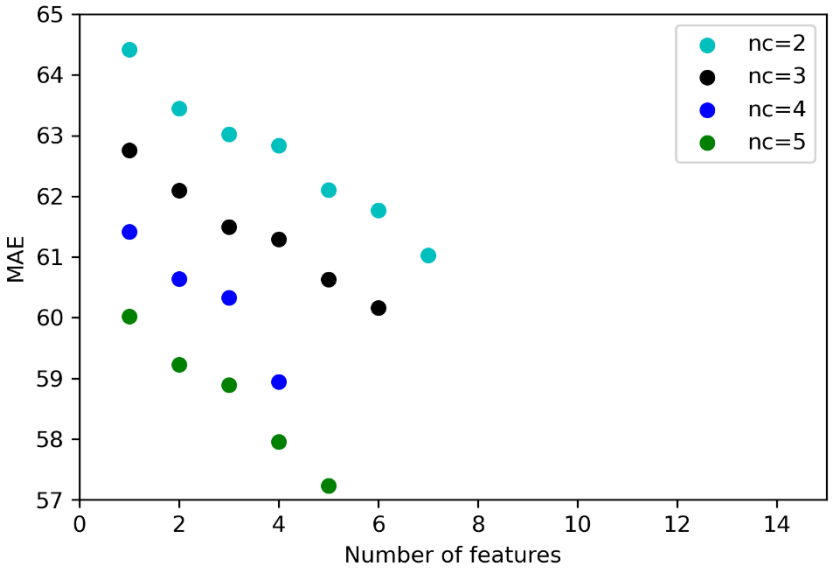


Figure 4.4: NSGA-III results for 2 to 5 clusters

Table 4.3: Hypervolumes with reference point (40, 150)

Number of clusters	NSGA-II	NSGA-III	SPEA2
2	3455.05	3458.66	3478.53
3	3527.43	3496.33	3495.44
4	3539.93	3545.53	3525.24
5	3614.81	3610.98	3577.80

Table 4.4: GA results for 2 to 5 clusters

Number of clusters	Nr of features	MAE
2	12	60.50
3	7	59.67
4	7	58.80
5	8	56.96

4.3 Parameter tuning

The parameters of the feature selection methods are important to find the optimal solution or optimal Pareto front of solutions. For this section the feature selection methods are divided into two categories, the filter methods and the wrapper methods.

4.3.1 Parameters for filter methods

The filter feature selection methods are MI, F-score, and Fisher score. The parameter for these methods is the number of features to select (k). To find the optimal value for k, each filter method is tested on $k = [1,2,3, \dots ,39]$. For each value of k, the feature selection method is performed 50 times. The performance of the model for each value of k is the average MAE of the 50 repetitions. The value of k with the smallest average MAE is considered the optimal number of features. The optimal number of features is used for the value for k. The results of this experiment are presented in Figure 4.5, and the corresponding optimal values for k are presented in Table 4.5.

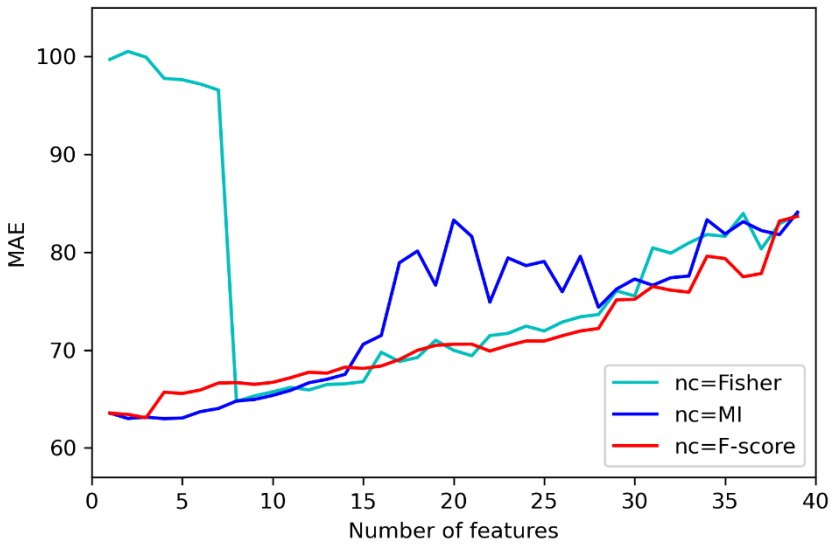


Figure 4.5: Results of filter methods to find optimal value for k with 50 repetitions

Table 4.5: Optimal number of features for filter methods

FS method	Optimal number of features (k)
Fisher	8
MI	4
F-score	3

4.3.2 Parameters for wrapper methods

The wrapper feature selection methods are GA, NSGA-II, NSGA-III, and SPEA2. The parameters for these algorithms are 'pop', 'gen', 'XCPB', 'MUTPB', and 'mutation rate'. These parameters are explained in more detail in section 3.2.4. The initial parameters for all four feature selection methods are set to $(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 100, 0.8, 0.1, \frac{1}{39})$. The population size and number of generations are both kept constant at 100. Also the mutation rate is kept constant at $\frac{1}{39}$. The only parameters that are experimented with in this research are the CXPB and MUTPB. For both values is tested is how the parameters influence the convergence of the algorithms. The parameter combinations for each wrapper feature selection method is presented in Table 4.6.

Table 4.6: Tested models for parameter tuning, remaining parameters are (pop, gen, mutation rate) = (100, 100, 1/39)

Model number	FS method	CXPB	MUTPB	Model number	FS method	CXPB	MUTPB
1	GA	0.9	0.1	9	NSGA-III	0.9	0.1
2	GA	0.9	0.2	10	NSGA-III	0.9	0.2
3	GA	0.8	0.2	11	NSGA-III	0.8	0.2
4	GA	0.8	0.1	12	NSGA-III	0.8	0.1
5	NSGA-II	0.9	0.1	13	SPEA2	0.9	0.1
6	NSGA-II	0.9	0.2	14	SPEA2	0.9	0.2
7	NSGA-II	0.8	0.2	15	SPEA2	0.8	0.2
8	NSGA-II	0.8	0.1	16	SPEA2	0.8	0.1

The first feature selection algorithm that is evaluated is the GA. The GA only optimizes a single objective: the MAE. The convergence graph of the four models with different parameter settings are presented in Figure 4.6. The MAE that is displayed in the graph is the minimum of all values for MAE in the population per generation. All four models succeed to converge very fast, therefore, all four models would be sufficient. Each model is only performed once, therefore, due to randomness it is difficult to draw strong conclusions about the 'best' parameters. However, the models with colors dark blue and red seem to have a slightly better convergence than the other models. The red model seems to converge even better than the dark

blue model, this appears especially after forty generations. Therefore, the settings of the red model are chosen for the GA, which are 0.9 for CXPB and 0.1 for MUTPB.

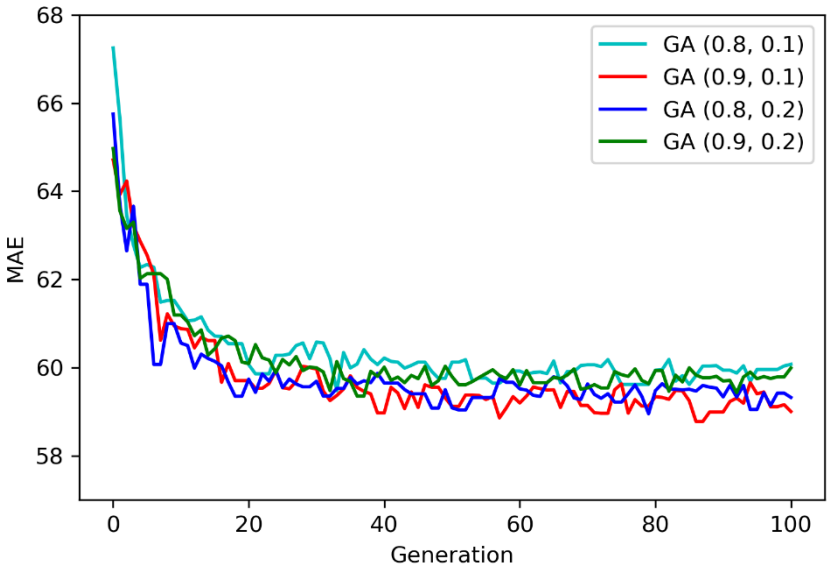


Figure 4.6: Convergence graph of GA

The next feature selection algorithms that are evaluated are the multi-objective algorithms (NSGA-II, NSGA-III, and SPEA2). The NSGA-II converges very fast with all four models on both objectives, this is visible in Figure 4.7 and Figure 4.8. The NSGA-III and SPEA2 have similar results, these results are presented in Appendix G. The convergence graphs of each model separately are presented in Appendix H. Each model is only tested once, therefore, due to randomness the results could be slightly different if the models are tested again. However, the model with parameters 0.9 for CXPB and 0.1 for MUTPB seems to have the best performance in terms of convergence. Therefore, these parameters are chosen for all multi-objective algorithms.

In addition, all wrapper methods converge within approximately 70 generations or less. Therefore, to save on computational time, the number of generations is reduced from 100 to 80. Meaning that all wrapper methods have the following parameters:

$$(pop, gen, cxpb, mutpb, mutation\ rate) = (100, 80, 0.9, 0.1, \frac{1}{39})$$

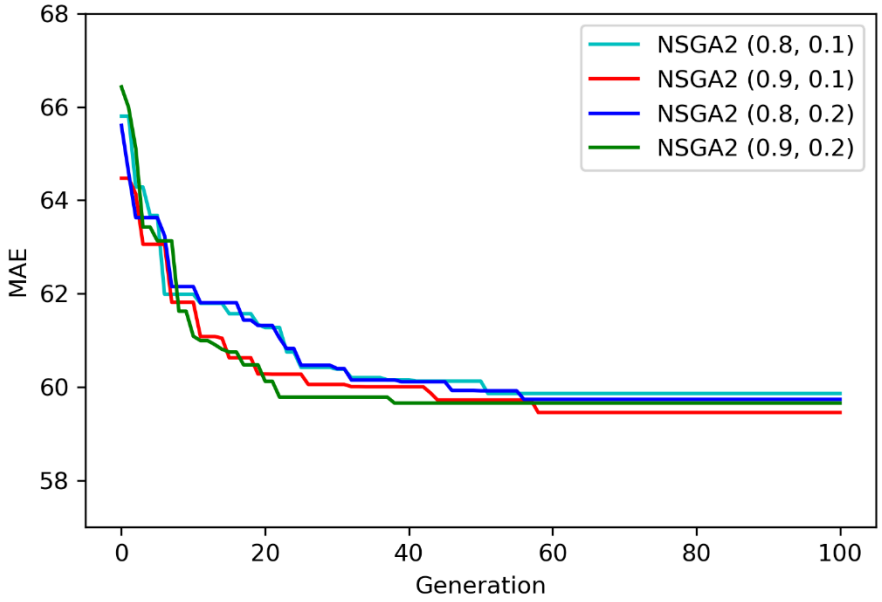


Figure 4.7: Convergence graph of NSGA-II based on MAE

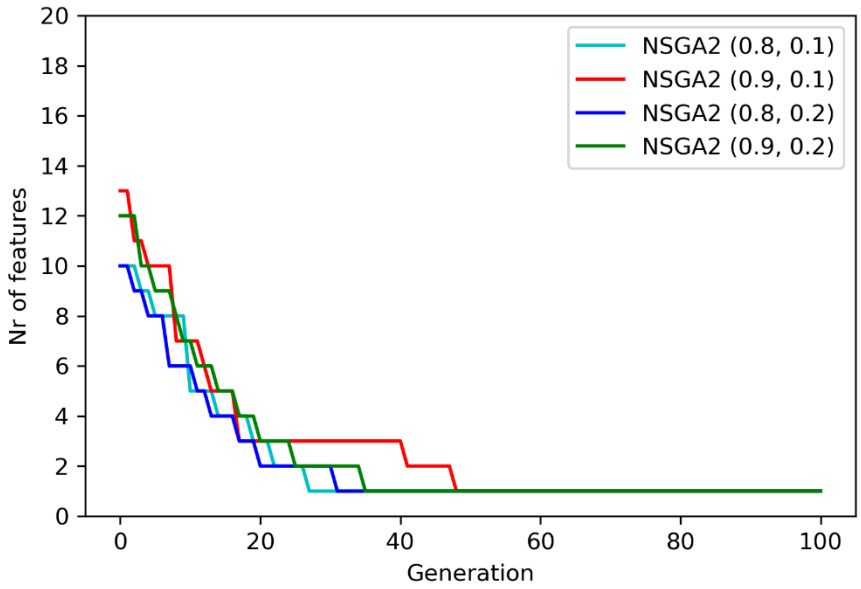


Figure 4.8: Convergence graph of NSGA-II based on number of features

4.4 Comparing feature selection methods

The seven feature selection methods are tested on the COVID-19 dataset with 3 clusters and the parameters described in section 4.3.1 and section 4.3.2.

4.4.1 Filter methods

First, the filter feature selection methods are tested (Fisher score, MI, and F-score). These feature selection methods were performed 50 times. The obtained results from the 50 repetitions consist of the average MAE, the standard deviation, and the minimum MAE. Furthermore, the selected features are constant throughout the repetitions. The results of the filter methods are presented in Table 4.7.

Table 4.7: Results of filter feature selection methods of 50 repetitions

FS method	Average MAE (st.dev.)	Min MAE	Number of features	Selected features
Fisher score	64.75 (0.53)	63.57	8	'pO2_FiO2_ratio', 'pCO2', 'Potassium_value', 'ALT_GPT__value', 'Fibrinogen_value', 'Lymphocytes_count_value', 'Platelets_value', 'Neutrophil_count_value'
MI	62.99 (0.38)	62.27	4	'pO2', 'pCO2', 'pO2_FiO2_ratio', 'Lactate_standardized_value'
F-score	63.08 (0.37)	62.03	3	'Albumin_value', 'AST_GOT_value', 'pO2_FiO2_ratio'

The distributions of the MAE of the 50 repetitions of the Fisher score method, MI method, and F-score method are presented in Figure 4.9, Figure 4.10, and Figure 4.11 respectively. In all three histograms there appears a 'bell curve'. Therefore, the assumption can be made that the data follows a normal distribution, which is one of the conditions for performing a t-test. In Figure 4.12 the boxplot of the MAE of the 50 repetitions of all three filter methods presented. In this boxplot is observed that the performance of the Fisher method is different from the performance of the MI and F-score methods. In addition, it is observed that the performance of the MI method and F-score method is almost similar. Therefore, a t-test is beneficial to determine whether there is a significant difference between the MI and F-score methods regarding their performance. First, an F-test is performed to determine whether the variances of the samples are equal, which is an assumption of the t-test. The outcomes of the F-test ($F = 0.96$, $p = 0.56$) mean that there is no significant difference between the variances of the MAE's of the MI method and the F-score method. The outcomes of the t-test ($t = 1.22$, $p = 0.23$) mean that there is no significant difference between the means of the MAE's of the MI method and the F-score method.

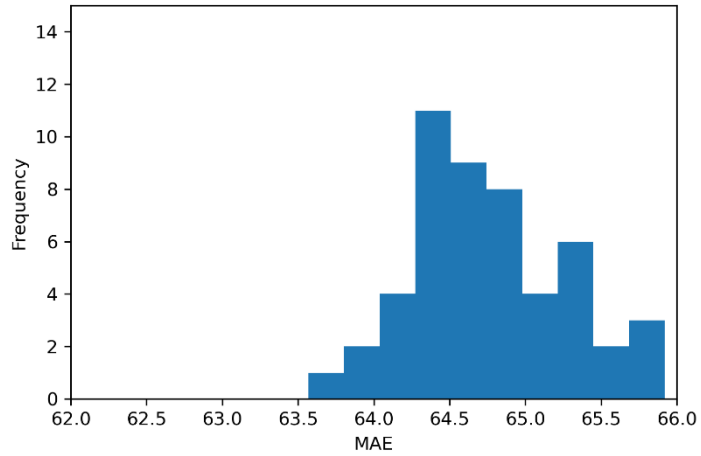


Figure 4.9: MAE of Fisher score method with 50 repetitions

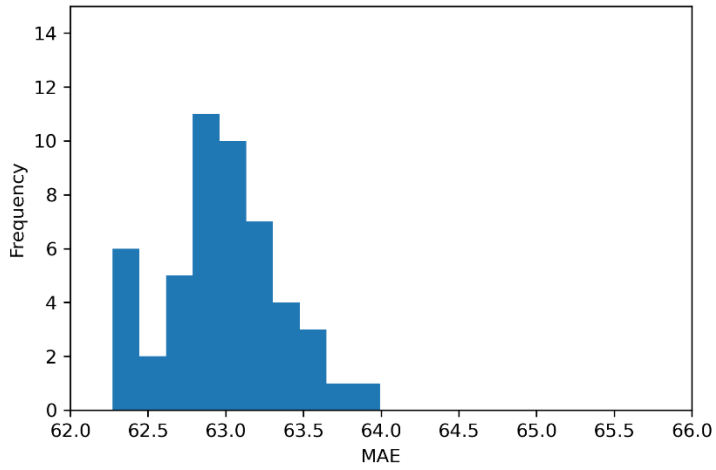


Figure 4.10: MAE of MI method with 50 repetitions

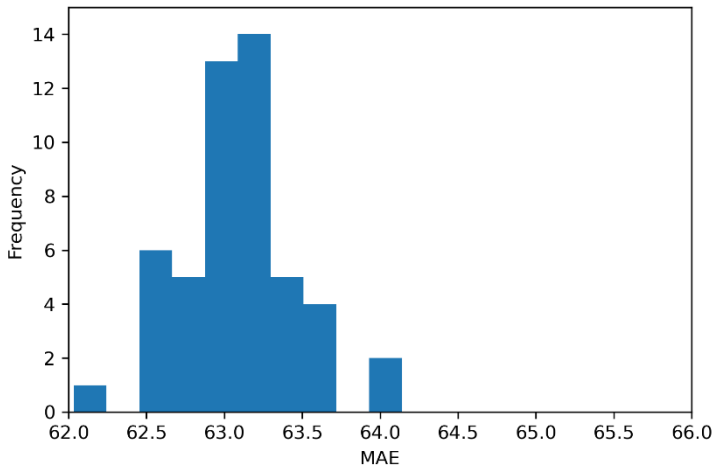


Figure 4.11: MAE of F-score method with 50 repetitions

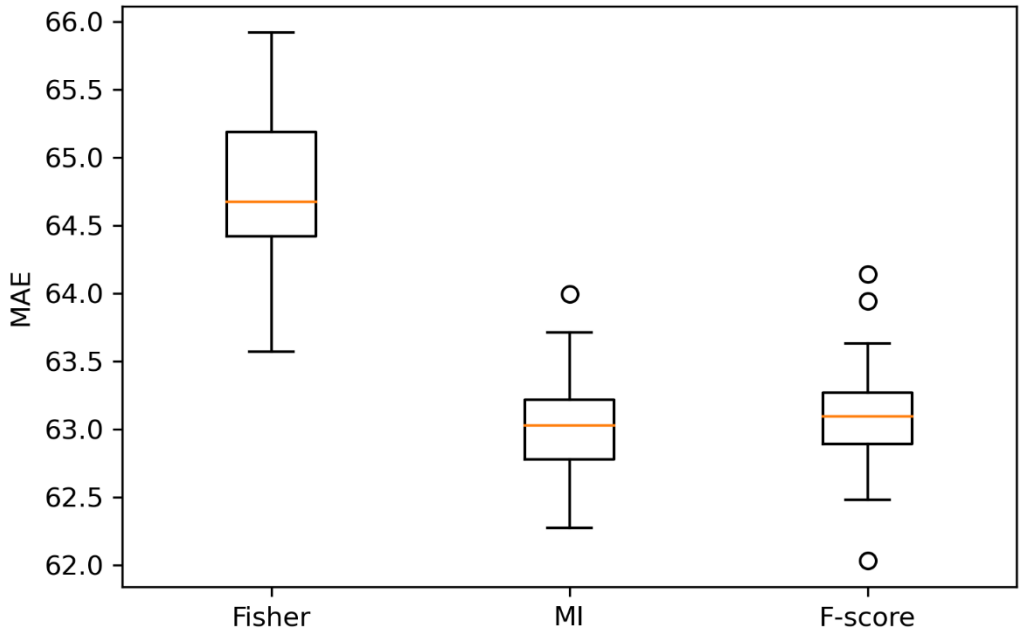


Figure 4.12: MAE boxplot of filter methods

4.4.2 Wrapper methods

GA

The first wrapper method that is tested is the GA. The GA is performed 50 times. Since the GA only minimizes the MAE, the solution with the smallest MAE is selected for evaluation. The results of the GA are presented in Figure 4.13. Form this figure is observed that the GA has high variation in the results when it comes to the number of features selected. Single objective optimization is the cause of the high variation in the number of features, because this value is not optimized. To support this observation, the coefficient of variation has been calculated, which is the ratio of the standard deviation and the mean. The coefficient of variation of the number of features and the MAE are 0.219 and 0.005 respectively.

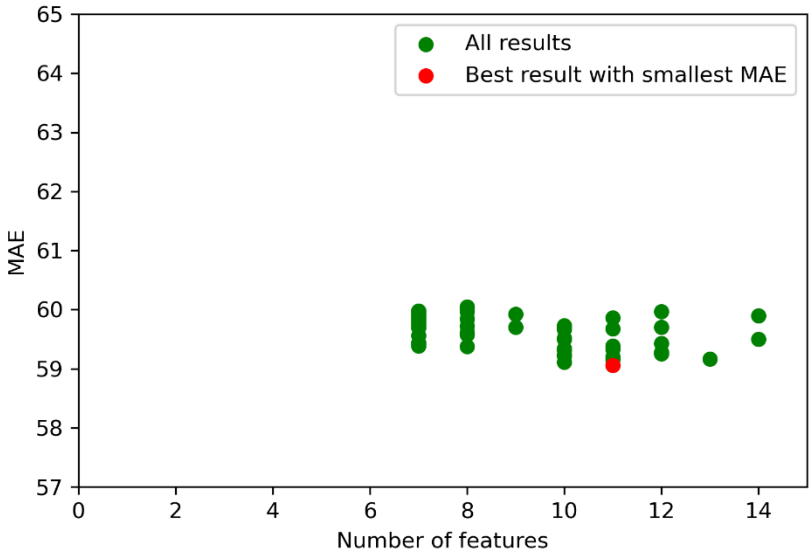


Figure 4.13: Results of GA method with 50 repetitions

ICPSO

The results of the ICPSO were obtained by Fuchs et al. (2021, under review). This feature selection method selected six features and achieved a MAE of 63.15. This feature selection method was not performed 50 times, therefore, it is not a fair comparison with the other feature selection methods tested in this research.

NSGA-II, NSGA-III, and SPEA2

Then the multi-objective feature selection methods are tested (NSGA-II, NSGA-III, and SPEA2). These methods are all performed 50 times. To compare the multi-objective feature selection methods, the hypervolume is evaluated. The average hypervolume, standard deviation, and maximal hypervolume are presented in Table 4.8. Boxplots of the hypervolumes of all three algorithms with 50 repetitions are presented in Figure 4.14. There is only a small difference between the hypervolumes of the algorithms. Therefore, based on the difference between the hypervolumes, it cannot be concluded which method performs better. Hence, the t-test is used to determine whether there is a significant difference between the performance of the multi-objective feature selection methods.

The distributions of the hypervolumes of the 50 repetitions of the NSGA-II, NSGA-III, and SPEA2 are presented in Figure 4.15, Figure 4.16, and Figure 4.17 respectively. In all three histograms there appears a ‘bell curve’. Therefore, the assumption can be made that the data follows a normal distribution. To test whether the variances of the samples are equal, the F-test is used. After the F-test, the t-test is performed. The results of the F-tests and t-tests are presented in Table 4.9. The outcomes of the F-test for NSGA-II and NSGA-III have a p-value<0.05, this means there is a significant difference between the variances of the samples. Therefore, the outcomes of this t-test are not valid, considering that equal variances is a condition for performing a t-test. The outcomes of the F-test for SPEA2 and NSGA-II show

there is no significant difference between the variances. Also, the outcomes of the F-test for SPEA2 and NSGA-III show there is no significant difference between the variances. Therefore the t-test can be performed. The outcomes of the t-test for SPEA2 and NSGA-II show that there is no significant difference between the means of the hypervolumes. Also, the outcomes of the t-test for SPEA2 and NSGA-III show that there is no significant difference between the means of the hypervolumes. This means there is not significant difference between the performance of the SPEA2 and NSGA-II, and the performance of the SPEA2 and NSGA-III.

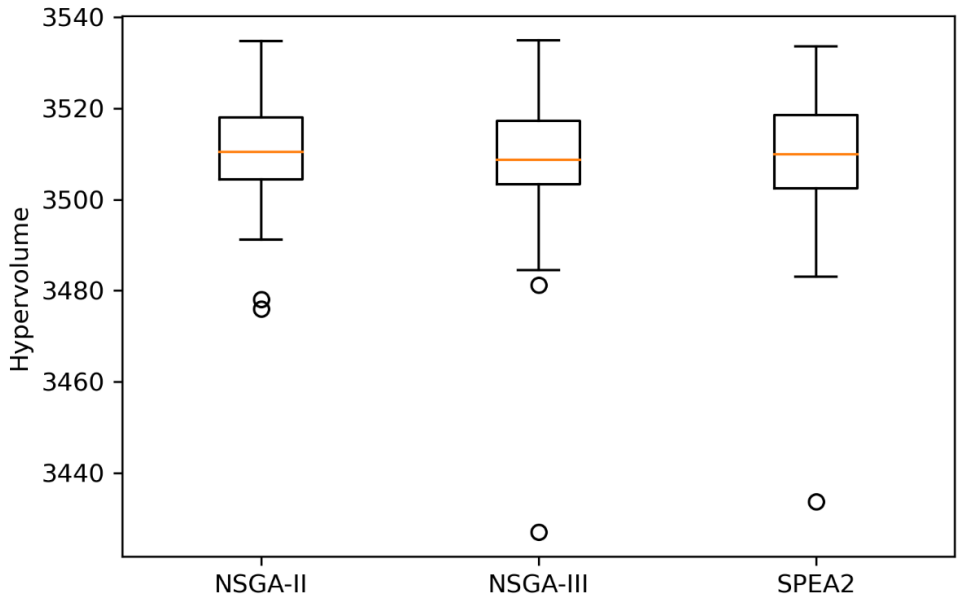


Figure 4.14: Hypervolume boxplot of multi-objective feature selection methods, reference point = (40, 150)

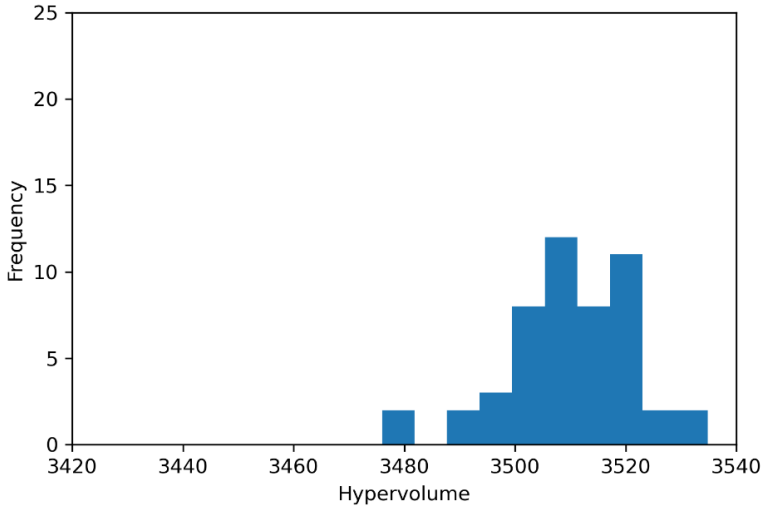


Figure 4.15: Hypervolume of NSGA-II method with 50 repetitions

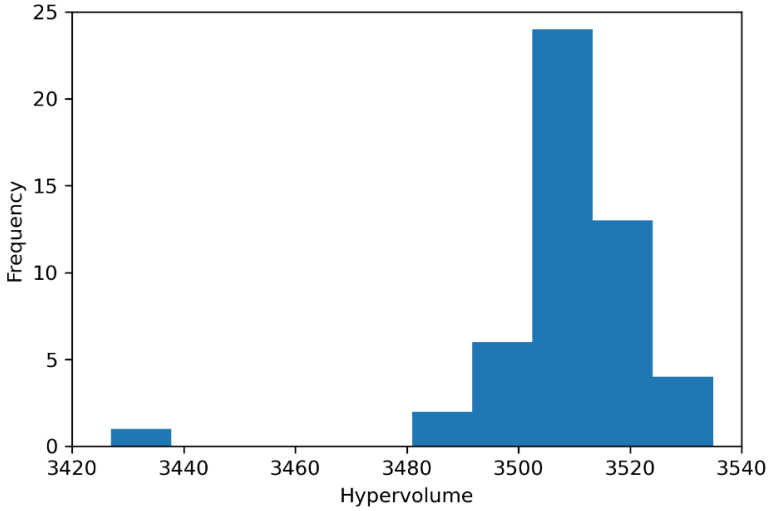


Figure 4.16: Hypervolume of NSGA-III method with 50 repetitions

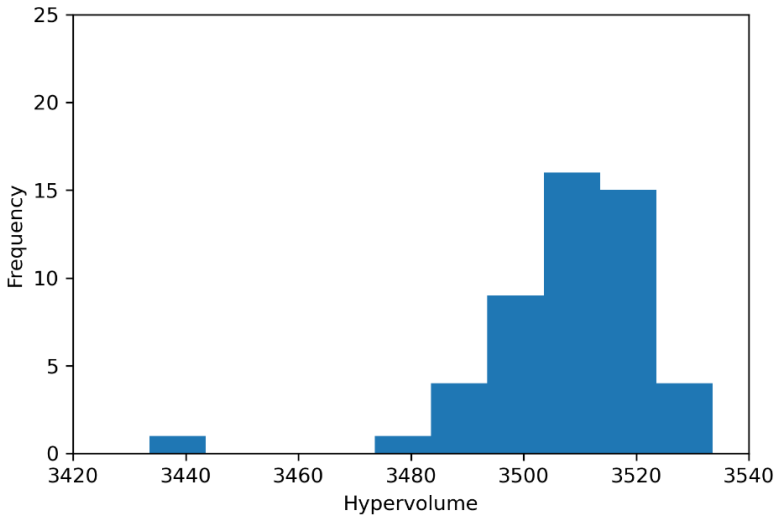


Figure 4.17: Hypervolume of SPEA2 method with 50 repetitions

Table 4.8: Hypervolume of multi-objective methods of 50 repetitions

FS method	Average hypervolume (st.dev.)	Max hypervolume
NSGA-II	3510.37 (11.49)	3534.82
NSGA-III	3508.18 (15.86)	3534.91
SPEA2	3508.25 (15.6)	3533.58

Table 4.9: Results F-tests and t-tests for hypervolume comparison

FS methods	F-test		t-test	
	F	p-value	t	p-value
NSGA-II and NSGA-III	1.90	0.01	-0.79*	0.43*
NSGA-II and SPEA2	0.54	0.98	0.77	0.44
SPEA2 and NSGA-III	1.03	0.45	-0.02	0.98

*) invalid

For each multi-objective method, the Pareto front with the greatest hypervolume is selected from the 50 repetitions. The resulting Pareto fronts from the selected algorithms are shown in Figure 4.18. Many datapoints overlap in the graph, therefore, the same results are shown in Table 4.10. The Pareto fronts of the NSGA-II and NSGA-III consist of eight solutions, and the Pareto front of the SPEA2 consists of ten solutions.

To be able to compare the Pareto fronts with the single objective solutions, one solution is selected from the Pareto front. This solution is the knee-point in the Pareto front, the knee-points are presented in Figure 4.19. The results of the knee-point solutions of the multi-objective methods and the results of the GA and ICPSO are presented in Table 4.11.

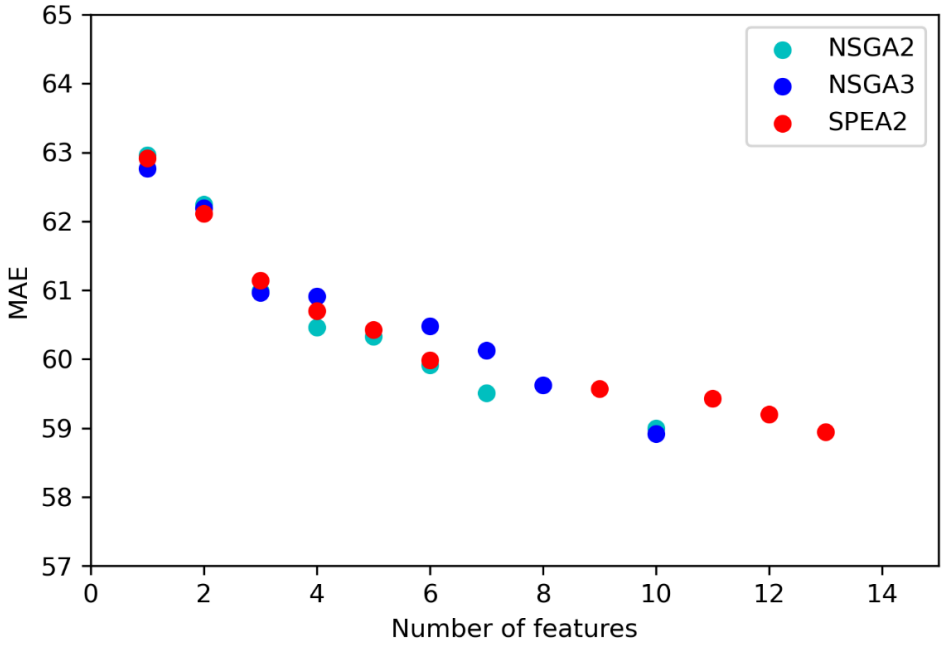


Figure 4.18: Pareto fronts of multi-objective methods

Table 4.10: Results of Pareto fronts with greatest hypervolume

FS method	Nr of features	MAE
NSGA-II	(10, 7, 6, 5, 4, 3, 2, 1)	(58.99, 59.51, 59.91, 60.33, 60.46, 60.99, 62.24, 62.96)
NSGA-III	(10, 8, 7, 6, 4, 3, 2, 1)	(58.92, 59.62, 60.13, 60.48, 60.91, 60.97, 62.19, 62.77)
SPEA2	(13, 12, 11, 9, 6, 5, 4, 3, 2, 1)	(58.94, 59.20, 59.43, 59.57, 59.98, 60.43, 60.70, 61.14, 62.11, 62.92)

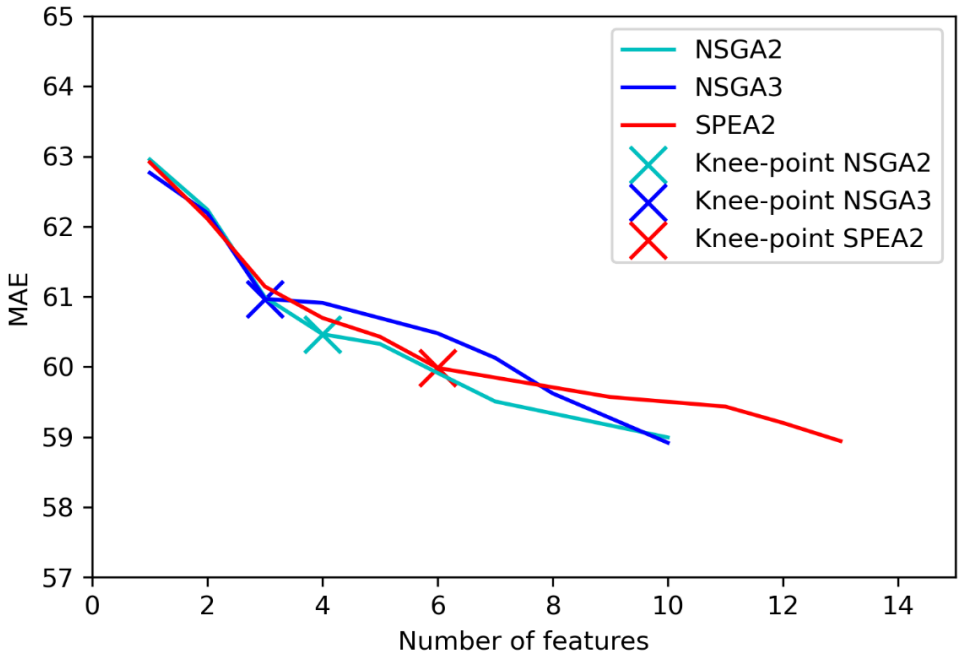


Figure 4.19: Pareto fronts of multi-objective methods with knee points

Table 4.11: Results of wrapper methods

FS method	MAE	Number of features	Selected features
GA	59.06	11	'Height', 'Diastolic_blood_pressure_on_admission', 'Urea_value', 'Serum_creatinine_value', 'Sodium_value', 'Red_blood_cell_count_value', 'Haemoglobin_value', 'Platelets_value', 'pO2', 'pO2_FiO2_ratio', 'BMI'
NSGA-II	60.46	4	'Platelets_value', 'pO2', 'pO2_FiO2_ratio', 'BMI'
NSGA-III	60.97	3	'Platelets_value', 'pO2', 'pO2_FiO2_ratio'
SPEA2	59.98	6	'Height', 'AST_GOT_value', 'Red_blood_cell_count_value', 'Platelets_value', 'pO2', 'pO2_FiO2_ratio'

4.4.3 All feature selection methods

The results of all tested feature selection methods are presented in Figure 4.20 and Table 4.12. In terms of best performing feature selection methods on two objectives (minimizing MAE and minimizing the number of features), there are four feature selection methods that dominate the other feature selection methods. The dominating feature selection methods are the NSGA-II, NSGA-III, SPEA2, and GA. These methods are considered dominant because they perform better than the other methods on both objectives, as shown in Figure 4.20.

In terms of feature selection to increase model interpretability, the smaller number of features selected the better. The NSGA-III and the F-score methods selected the smallest number of features, namely three features. The NSGA-II and MI methods selected four features. And the remaining feature selection methods selected six or more features. However, technically the three multi-objective methods (NSGA-II, NSGA-III, and SPEA2) selected the smallest number of features, since the Pareto fronts contain solutions with only one or two selected features as shown in Figure 4.18 in section 4.4.2.

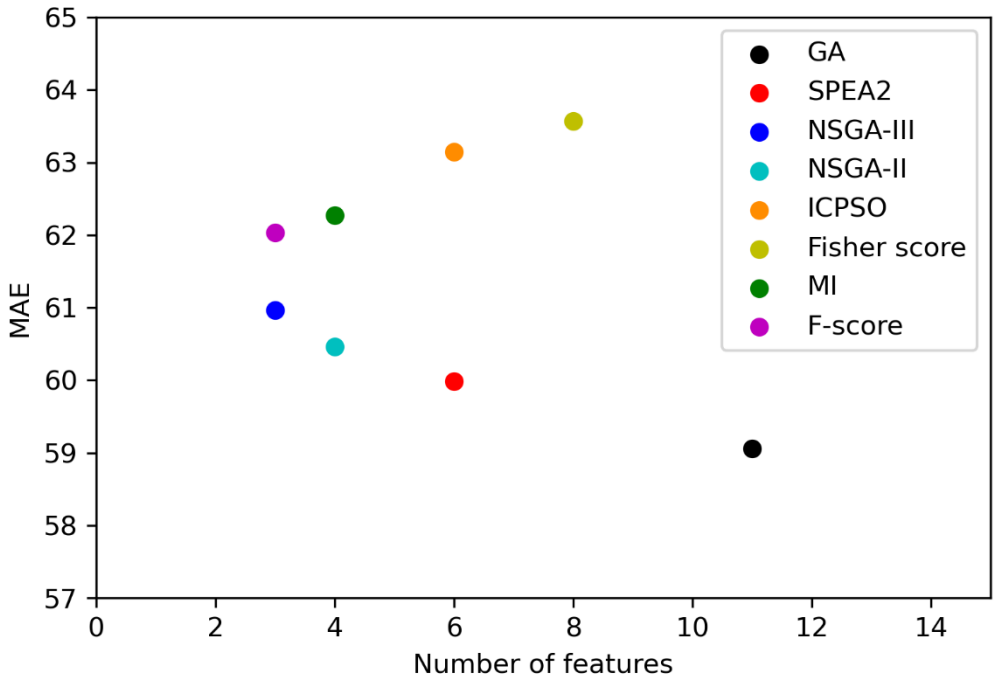


Figure 4.20: Results of all feature selection methods

The feature 'pO2_FiO2_ratio' is selected by every feature selection method. This is the PaO2/FiO2 value that is registered of the patient at the moment of hospitalization. The target variable of the dataset is the worst PaO2/FiO2 value during the hospitalization. Considering this, the PaO2/FiO2 value at admission has an influence on the worst PaO2/FiO2 value.

Other features that were selected frequently are 'Platelets_value' (selected five times), 'pO2' (selected five times), 'Red_blood_cell_count_value' (selected three times), and 'BMI' (selected three times). The other selected features were only selected by one or two feature selection methods.

Table 4.12: Selected features by all seven feature selection methods

FS method	MAE	Number of features	Selected features
Fisher score	63.57	8	'pO2_FiO2_ratio', 'pCO2', 'Potassium_value', 'ALT_GPT__value', 'Fibrinogen_value', 'Lymphocytes_count_value', 'Platelets_value', 'Neutrophil_count_value'
MI	62.27	4	'pO2', 'pCO2', 'pO2_FiO2_ratio', 'Lactate_standardized_value'
F-score	62.03	3	'Albumin_value', 'AST_GOT_value', 'pO2_FiO2_ratio'
GA	59.06	11	'Height', 'Diastolic_blood_pressure_on_admission', 'Urea_value', 'Serum_creatinine_value', 'Sodium_value', 'Red_blood_cell_count_value', 'Haemoglobin_value', 'Platelets_value', 'pO2', 'pO2_FiO2_ratio', 'BMI'
NSGA-II	60.46	4	'Platelets_value', 'pO2', 'pO2_FiO2_ratio', 'BMI'
NSGA-III	60.97	3	'Platelets_value', 'pO2', 'pO2_FiO2_ratio'
SPEA2	59.98	6	'Height', 'AST_GOT_value', 'Red_blood_cell_count_value', 'Platelets_value', 'pO2', 'pO2_FiO2_ratio'
ICPSO	63.15	6	‘Weight’, ‘Systolic_blood_pressure_on_admission’, ‘Red_blood_cell_count_value’, ‘White_blood_cell_count_value’, 'pO2_FiO2_ratio', 'BMI'

5 Discussion

5.1 Conclusion

The objective of this research was to improve interpretability of fuzzy models that use high dimensional data and automatically generated rules. This was done by implementing the most appropriate feature selection method with pyFUME such that the complexity of the model is reduced and the performance of the model is maintained in order to increase user's understandability of the model's rationale. The approach for achieving this objective was to find the most appropriate feature selection method to reduce complexity and improve interpretability of a fuzzy model. Seven feature selection methods were tested on an artificial dataset and a more complex dataset, namely the COVID-19 dataset, to find the most appropriate method. These seven methods are: MI, F-score, Fisher score, GA, NSGA-II, NSGA-III, and SPEA2. The methods are evaluated on the amount of features selected, and on how the model performs with the selected features.

For a more simple dataset (which in this research was an artificial dataset) all tested feature selection methods succeeded in selecting the relevant features in the dataset. Therefore, all feature selection methods are considered appropriate for a simple dataset. However, for a more complex dataset (which in this research was the COVID-19 dataset) the performance varied for the tested feature selection methods.

The least performing feature selection methods are the Fisher score and the GA. First, the Fisher score method ranked the features such that the minimal number of features to select from the COVID-19 dataset is eight. This means the Fisher score ranked seven other features to be more relevant than the true most relevant feature. Secondly, the GA was programmed to minimize a single objective: the MAE of the model. The result of this is that the GA achieved the lowest MAE of all seven feature selection methods. However, the number of selected features was eleven. Considering the high number of selected features, the GA obtained results that do not support the goal of this research.

The next feature selection methods that are discussed performed better than the GA and the Fisher score method. The MI and F-score feature selection methods are both filter methods which rank the features on relevance. Both methods managed to rank the features of the COVID-19 dataset such that the most relevant features were selected. The ranking process of the features is very consistent. Meaning that the feature selection methods will obtain the same ranking if the ranking process is repeated. There are two advantages of using the MI and F-score feature selection methods. The first advantage is that the computational time to return the ranking of the features is relatively short. The second advantage is that the implementation is relatively simple, since is only consists of calculating the scores and then ranking the scores and the corresponding features. The disadvantages of the MI and F-score feature selection methods are the following. The methods only return a ranking of the features and do not provide an optimal number of features to select. Also, these methods only consider the relation between the features and the target variable for each feature individually. This means that possible relations between features are not considered.

The performance of the MI method and F-score method on the COVID-19 dataset is almost similar. There was no significant difference between the achieved MAE of the methods in terms of model performance with the selected features. And in terms of number of features selected, the F-score method selected one less feature than the MI method. However, this does not mean that the F-score method is better than the MI method. Since the MI method has smaller values for MAE in most cases if there are less than 15 features selected, which is visible in Figure 4.5 in section 4.3.1. Therefore, the MI method is considered more successful in ranking the features based on relevance than the F-score method for the COVID-19 dataset.

The best performing feature selection methods in this research are the three multi-objective wrapper methods: NSGA-II, NSGA-III, and SPEA2. In contrast to the filter methods, these methods do not rely on a calculation the capture the relation between the features and the target variable. Instead, the multi-objective wrapper methods test many different combinations of features and evolve to a set of optimal solutions. These methods return a Pareto front of optimal solutions. Each solution in the Pareto front represents a set of selected features. To decide which solution should be implemented, one can use the knee method, or decide manually which solution will fit the situation best by for example assigning weights to the objectives. Since the multi-objective wrapper methods return multiple solutions, there is a great chance of returning at least one solution that supports the goal of this research. Therefore, making a comparison between the returned solutions of the filter methods and the wrapper methods is not really fair.

The advantages and disadvantages of the multi-objective wrapper methods are the following. The first advantage is that the methods return a Pareto front of solutions, which gives the user more freedom of choosing a solution that is suitable for the situation. At the same time, this could also be considered a disadvantage, because choosing one solution form a Pareto front is difficult. The second advantage is that the methods are minimizing the number of selected features, whereas the filter methods are only ranking the features and do not provide an optimal number of selected features. The third advantage is that the methods do not rely on certain relations between the features and the target variable, since the wrapper methods in this research all evaluate many different combinations of features to find the optimal set of features. Therefore, the multi-objective wrapper methods perform well on the complex dataset. Finally, another disadvantage of these methods is that they tend to use long computational times to obtain the optimal solutions.

According to the statistical tests that were performed, there is no significant difference between the hypervolumes of the Pareto fronts. Therefore, the performance of the NSGA-II, NSGA-III, and SPEA2 on the COVID-19 dataset is almost similar.

The recommended feature selection method to use in combination with pyFUME for the COVID-19 dataset is the NSGA-II, NSGA-III, or SPEA2. Since there was no significant difference between the performances of these methods, there is no recommendation for specifically one of them. It is recommended to use the knee-method to choose a single solution from the Pareto front. However, it is also possible to select a solution manually. For example, if the user for some reason has the goal to only select one feature from the dataset, then the solution which represents one feature can be chosen. The disadvantage of these methods is the long computational time, therefore, the second best method that is recommended is the MI method. This method can be used in situations where a long computational time is undesirable.

The feature selection methods aimed to select the most relevant features from the COVID-19 dataset. The three features that were selected by all multi-objective wrapper methods are 'Platelets_value', 'pO2', and 'pO2_FiO2_ratio'. Considering that multi-objective wrapper methods performed best, it is recommended to keep at least these three features in the dataset to build the fuzzy model. Building a fuzzy model with only these three features causes a large reduction in model complexity, comparing to a fuzzy model with all 39 features. This reduction in model complexity contributes to better interpretability of the model.

5.2 Suggestions for future research

In this research, the main goal was to use feature selection to reduce the complexity of the fuzzy model to increase the model interpretability. The model interpretability was measured by the number of features selected, which impacts the length of the fuzzy rules. The assumption was made that the shorter the rules, the more interpretable the model. Another assumption that was made is, the fewer rules, the more interpretable the model. The number of rules is equal to the number of clusters, therefore, this was taken into consideration for choosing the number of clusters for pyFUME. Future research could contribute to discovering new indices for measuring model interpretability.

Minimizing the number of features was used to improve model interpretability in this research. An additional objective could be to minimize the number of clusters simultaneously. This could be implemented with the multi-objective methods. These methods would then optimize three objectives: minimize number of features, minimize number of clusters, and minimize MAE. This could also be reduced to two objectives if the number of features and number of clusters are combined into one objective. The new objective could be to minimize the total rule length, which is the total number of premises in all rules. This is calculated by the number of features multiplied by the number of clusters.

Although there appeared some differences in the performance of the tested feature selection methods in this research, there was not one feature selection method that outperformed all the other methods. Testing the feature selection methods on more datasets could possibly reveal a larger gap between the performance of the methods. Which would make it easier to make recommendations for the most appropriate feature selection method.

In this research, all feature selection methods were implemented with the support of pyFUME. The next step would be to implement them within pyFUME. The recommendation is to implement one filter method and one wrapper method within pyFUME. In this way, the user can choose which feature selection method would suit the situation best. For example, if the user does not have time for long computations, then the filter method would be the best choice. Even though the filter methods in this research were not the best performing feature selection methods, it is important to offer this option to the user. Because in terms of model interpretability and performance, applying a filter feature selection method is still better than using no feature selection at all. On the other hand, if the user has plenty of time for computations and expects the best features to be selected, then the multi-objective wrapper methods would be the best choice.

References

Abonyi, J., Roubos, J. A., Oosterom, M., & Szeifert, F. (2001). Compact TS-fuzzy models through clustering and OLS plus FIS model reduction. *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*, *3*, 1420–1423 vol.2. https://doi.org/10.1109/FUZZ.2001.1008925

Acute Respiratory Distress Syndrome (ARDS). (n.d.). Retrieved July 1, 2021, from https://www.lung.org/lung-health-diseases/lung-disease-lookup/ards

Alonso, J. M., Magdalena, L., & González-Rodríguez, G. (2009). Looking for a good fuzzy system interpretability index: An experimental approach. *International Journal of Approximate Reasoning*, *51*(1), 115–134. https://doi.org/10.1016/j.ijar.2009.09.004

Alonso, J. M., Magdalena, L., & Guillaume, S. (2008). HILK: A new methodology for designing highly interpretable linguistic knowledge bases using the fuzzy logic formalism. *International Journal of Intelligent Systems*, *23*(7), 761–794. https://doi.org/10.1002/int.20288

Amaral, J. L. M., Sancho, A. G., Faria, A. C. D., Lopes, A. J., & Melo, P. L. (2020). Differential diagnosis of asthma and restrictive respiratory diseases by combining forced oscillation measurements, machine learning and neuro-fuzzy classifiers. *Medical & Biological Engineering & Computing*, *58*(10), 2455–2473. https://doi.org/10.1007/s11517-020-02240-7

Anaraki, J. R., & Eftekhari, M. (2013). Rough set based feature selection: A Review. *The 5th Conference on Information and Knowledge Technology*, 301–306. https://doi.org/10.1109/IKT.2013.6620083

Antonelli, M., Ducange, P., Marcelloni, F., & Segatori, A. (2016). On the influence of feature selection in fuzzy rule-based regression model generation. *Information Sciences*, *329*, 649–669. https://doi.org/10.1016/j.ins.2015.09.045

Bonett, D. G., & Wright, T. A. (2000). Sample size requirements for estimating pearson, kendall and spearman correlations. *Psychometrika*, *65*(1), 23–28. https://doi.org/10.1007/BF02294183

Cartesius | userinfo.surfsara.nl. (n.d.). Retrieved June 29, 2021, from https://userinfo.surfsara.nl/systems/cartesius

Chen, P., Wilbik, A., Loon, S. van, Boer, A.-K., & Kaymak, U. (2018). Finding the optimal number of features based on mutual information. *Advances in Fuzzy Logic and Technology 2017 - Proceedings of: EUSFLAT-2017 – The 10th Conference of the European Society for Fuzzy Logic and Technology, IWIFSGN'2017 – The 16th International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets*, 477–486. https://doi.org/10.1007/978-3-319-66830-7_43

Das, I. (1999). On characterizing the "knee" of the Pareto curve based on normal-boundary intersection. *Structural Optimization*, *18*(2–3), 107–115.

Deb, K., & Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box

Constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577–601. https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/MCI.2006.329691

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Ant System: An Autocatalytic Optimizing Process*. 22.

Elssied, N. O. F., Ibrahim, O., & Osman, A. H. (2014). A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology*, *7*(3), 625–638.

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, *7*(2), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A. G., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, *13*(1), 2171–2175.

Fuchs, C., Nobile, M. S., Torlasco, C., Papetti, D. M., Cascella, A., Menè, R., Besozzi, D., Parati, G., & Kaymak, U. (2021). Learning Interpretable AI Systems for Clinical Decision Support. *IEEE Computational Intelligence Magazine*.

Fuchs, C., Spolaor, S., Nobile, M. S., & Kaymak, U. (2020). pyFUME: A Python Package for Fuzzy Model Estimation. *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–8. https://doi.org/10.1109/FUZZ48607.2020.9177565

Gayathri, B. M., & Sumathi, C. P. (2015). Mamdani fuzzy inference system for breast cancer risk detection. *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 1–6. https://doi.org/10.1109/ICCIC.2015.7435670

Getting started with Cartesius | userinfo.surfsara.nl. (n.d.). Retrieved June 29, 2021, from https://userinfo.surfsara.nl/systems/cartesius/getting-started

Ghazavi, S. N., & Liao, T. W. (2008). Medical data mining by fuzzy modeling with selected features. *Artificial Intelligence in Medicine*, *43*(3), 195–206. https://doi.org/10.1016/j.artmed.2008.04.004

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Holland, J. H., Holland, P. of P. and of E. E. and C. S. J. H., & Holland, S. L. in H. R. M. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press.

Hong, T.-P., & Lee, C.-Y. (1996). Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets and Systems*, *84*(1), 33–47. https://doi.org/10.1016/0165-0114(95)00305-3

Hunt, J. (2019). Multiprocessing. In J. Hunt (Ed.), *Advanced Guide to Python 3 Programming* (pp. 363–376). Springer International Publishing. https://doi.org/10.1007/978-3-030-25943-3_31

Ishibuchi, H., & Nojima, Y. (2007). Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning*, *44*(1), 4–31. https://doi.org/10.1016/j.ijar.2006.01.004

Jiménez, F., Gomez-Skarmeta, A. F., Sanchez, G., & Deb, K. (2002). An evolutionary algorithm for constrained multi-objective optimization. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, *2*, 1133–1138 vol.2. https://doi.org/10.1109/CEC.2002.1004402

Jiménez, F., Martínez, C., Marzano, E., Palma, J. T., Sánchez, G., & Sciavicco, G. (2019). Multiobjective Evolutionary Feature Selection for Fuzzy Classification. *IEEE Transactions on Fuzzy Systems*, *27*(5), 1085–1099. https://doi.org/10.1109/TFUZZ.2019.2892363

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, *5*, 4104–4108 vol.5. https://doi.org/10.1109/ICSMC.1997.637339

Kerr-Wilson, J., & Pedrycz, W. (2020). Generating a hierarchical fuzzy rule-based model. *Fuzzy Sets and Systems*, *381*, 124–139. https://doi.org/10.1016/j.fss.2019.07.013

Kim, T. K., & Park, J. H. (2019). More about the basic assumptions of t-test: Normality and sample size. *Korean Journal of Anesthesiology*, *72*(4), 331–335. https://doi.org/10.4097/kja.d.18.00292

Kira, K., & Rendell, L. A. (1992). A Practical Approach to Feature Selection. In D. Sleeman & P. Edwards (Eds.), *Machine Learning Proceedings 1992* (pp. 249–256). Morgan Kaufmann. https://doi.org/10.1016/B978-1-55860-247-2.50037-1

Kumar, V., & Minz, S. (2014). Feature Selection: A literature Review. *The Smart Computing Review*, *4*(3). https://doi.org/10.6029/smartcr.2014.03.007

Lal, T. N., Chapelle, O., Weston, J., & Elisseeff, A. (2006). Embedded Methods. In I. Guyon, M. Nikravesh, S. Gunn, & L. A. Zadeh (Eds.), *Feature Extraction* (Vol. 207, pp. 137–165). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-35488-8_6

Lee, H.-M., Chen, C.-M., Chen, J.-M., & Jou, Y.-L. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *31*(3), 426–432. https://doi.org/10.1109/3477.931536

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature Selection: A Data Perspective. *ACM Computing Surveys*, *50*(6), 94:1-94:45. https://doi.org/10.1145/3136625

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(4), 417–473. https://doi.org/10.1111/j.1467-9868.2010.00740.x

Nasir, V., Cool, J., & Sassani, F. (2019). Acoustic emission monitoring of sawing process: Artificial intelligence approach for optimal sensory feature selection. *The International Journal of Advanced Manufacturing Technology*, *102*(9), 4179–4197. https://doi.org/10.1007/s00170-019-03526-3

Nauck, D. D. (2003). Measuring interpretability in rule-based classification systems. *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.*, *1*, 196–201 vol.1. https://doi.org/10.1109/FUZZ.2003.1209361

Nobile, M. S., Cazzaniga, P., Besozzi, D., Colombo, R., Mauri, G., & Pasi, G. (2018). Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization. *Swarm and Evolutionary Computation*, *39*, 70–85. https://doi.org/10.1016/j.swevo.2017.09.001

Pawlak, Z. (1982). Rough sets. *International Journal of Computer & Information Sciences*, *11*(5), 341–356. https://doi.org/10.1007/BF01001956

Pearson, K., & Henrici, O. M. F. E. (1896). VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, *187*, 253–318. https://doi.org/10.1098/rsta.1896.0007

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pulkkinen, P., & Koivisto, H. (2007). Identification of interpretable and accurate fuzzy classifiers and function estimators with hybrid methods. *Applied Soft Computing*, *7*(2), 520–533. https://doi.org/10.1016/j.asoc.2006.11.001

Rajagopal, R. (1999). *Introduction to Microsoft Windows NT Cluster Server: Programming and Administration*. CRC Press.

Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Rajput, D. S., Kaluri, R., & Srivastava, G. (2020). Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis. *Evolutionary Intelligence*, *13*(2), 185–196. https://doi.org/10.1007/s12065-019-00327-1

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, *53*(1), 23–69. https://doi.org/10.1023/A:1025667309714

Roubos, J. A., Setnes, M., & Abonyi, J. (2003). Learning fuzzy classification rules from labeled data. *Information Sciences*, *150*(1), 77–93. https://doi.org/10.1016/S0020-0255(02)00369-9

Services offered by SURF | SURF.nl. (n.d.). Retrieved June 29, 2021, from https://www.surf.nl/en/about-surf/services-offered-by-surf

Shen, Q., & Jensen, R. (2004). Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring. *Pattern Recognition*, *37*(7), 1351–1363. https://doi.org/10.1016/j.patcog.2003.10.016

Siedlecki, W., & Sklansky, J. (1989). *Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition*. 141–150. https://doi.org/10.1142/9789814343138_0006

Slurm Workload Manager—Overview. (n.d.). Retrieved June 29, 2021, from https://slurm.schedmd.com/overview.html

Snedecor, G. W., & Cochran, W. G. (1989). Statistical Methods. *Iowa State University Press, Ames, Iowa*, *1191*.

Soares, E., Costa, P., Costa, B., & Leite, D. (2018). Ensemble of evolving data clouds and fuzzy models for weather time series prediction. *Applied Soft Computing*, *64*, 445–453. https://doi.org/10.1016/j.asoc.2017.12.032

Spearman, F. H. (1904). *The strategy of great railroads.* https://trid.trb.org/view/590746

Spolaor, S., Fuchs, C., Cazzaniga, P., Kaymak, U., Besozzi, D., & Nobile, M. S. (2020). Simpful: A User-Friendly Python Library for Fuzzy Logic. *International Journal of Computational Intelligence Systems*, *13*(1), 1687–1698. https://doi.org/10.2991/ijcis.d.201012.002

Strasser, S., Goodman, R., Sheppard, J., & Butcher, S. (2016). A New Discrete Particle Swarm Optimization Algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 53–60. https://doi.org/10.1145/2908812.2908935

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(1), 116–132. https://doi.org/10.1109/TSMC.1985.6313399

The ARDS Definition Task Force. (2012). Acute Respiratory Distress Syndrome: The Berlin Definition. *JAMA*, *307*(23), 2526–2533. https://doi.org/10.1001/jama.2012.5669

Tiruneh, G. G., & Robinson Fayek, A. (2019). Feature Selection for Construction Organizational Competencies Impacting Performance. *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1–5. https://doi.org/10.1109/FUZZ-IEEE.2019.8858820

Tsang, C.-H., Kwong, S., & Wang, H. (2007). Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition*, *40*(9), 2373–2391. https://doi.org/10.1016/j.patcog.2006.12.009

Vieira, S. M., Sousa, J. M. C., & Runkler, T. A. (2010). Two cooperative ant colonies for feature selection using fuzzy models. *Expert Systems with Applications*, *37*(4), 2714–2723. https://doi.org/10.1016/j.eswa.2009.08.026

Vieira, S. M., Sousa, J. M. C., & Runkler, T. A. (2007). Ant Colony Optimization Applied to Feature Selection in Fuzzy Classifiers. In P. Melin, O. Castillo, L. T. Aguilar, J. Kacprzyk, & W. Pedrycz (Eds.), *Foundations of Fuzzy Logic and Soft Computing* (pp. 778–788). Springer. https://doi.org/10.1007/978-3-540-72950-1_76

Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. Springer.

Yu, S., De Backer, S., & Scheunders, P. (2002). Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. *Pattern Recognition Letters*, *23*(1), 183–190. https://doi.org/10.1016/S0167-8655(01)00118-0

Zadeh, L. A. (1965). Electrical Engineering at the Crossroads. *IEEE Transactions on Education*, *8*(2), 30–33. https://doi.org/10.1109/TE.1965.4321890

Zadeh, L. A. (1988). Fuzzy logic. *Computer*, *21*(4), 83–93. https://doi.org/10.1109/2.53

Zitzler, E., Brockhoff, D., & Thiele, L. (2007). The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, & T. Murata (Eds.), *Evolutionary Multi-Criterion Optimization* (pp. 862–876). Springer. https://doi.org/10.1007/978-3-540-70928-2_64

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm for multi-objective optimization. *Evolutionary Methods for Design Optimization and Control With Applications to Industrial Problems*, 95–100.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—A comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature—PPSN V* (pp. 292–301). Springer. https://doi.org/10.1007/BFb0056872

Appendices

A. Description of files

Datasets

File title	Description
COVID_data	This is the COVID-19 dataset with 39 features and one target variable.
FSdata	This is the artificial dataset with 10 features and one target variable.

Results data

File title	Description
Cluster_results_1	Results of feature selection methods with 2, 3, 4, and 5 clusters on COVID-19 data. Also the results of the filter methods with 50 repetitions and 3 clusters on COVID-19 data.
Final_results_1	Consists of the final results of the wrapper methods (GA, NSGA-II, NSGA-III, and SPEA2) with 50 repetitions on COVID-19 data.
Nr clusters without fs	This are the results of pyFUME without feature selection for 2, 3, 4, and 5 clusters on COVID-19 data.
Parameter_results_1	This are the results of the wrapper methods with four different parameter settings on the COVID-19 data.

Feature selection Python scripts

File title	Description
Fisher score	Implementation of the Fisher score.
GA	Implementation of the genetic algorithm.
MI and F-score	Implementation of the MI method and the F-score method.
NSGA2	Implementation of the NSGA-II.
NSGA3	Implementation of the NSGA-III.
SPEA2	Implementation of the SPEA2.

Plots Python scripts

File title	Description
Plots - clusters	This script makes plots of the results for different number of clusters. It also returns a table.

Plots - data	This script returns scatter plots of the artificial dataset and the COVID-19 dataset.
Plots - hypervolume and knee example	This script returns the hypervolume example plot and the knee-method example plot.
Plots - membership functions example	This script returns the example plots for the membership function shapes: Gaussian and trapezoidal.
Plots - parameters and convergence of wrapper methods	This script returns plots of the convergence of the wrapper methods with different four different parameter settings.
Plots - parameters for filter methods	This script returns one plot with the performance of the filter methods with different parameters. It also returns the optimal number of features for every filter method.
Plots - results of filter methods	This script returns the histograms of the filter methods and a boxplot to compare the methods. It also returns the outcomes of the statistical tests.
Plots - results of wrapper methods and final scatter plot	This script returns multiple plots: • A scatter plot with the results of the GA. • A boxplot to compare NSGA-II, NSGA-III, and SPEA2. • Histograms of the results of NSGA-II, NSGA-III, and SPEA2. • The Pareto fronts of NSGA-II, NSGA-III, and SPEA2. • The Pareto fronts of NSGA-II, NSGA-III, and SPEA2 with knee-point. • The final scatter plots with the results of all feature selection methods. It also returns the results of the statistical tests and some additional results that were used to present in tables.

B. Search terms and selection criteria for systematic literature review

Search terms and sources

Source	SpringerLink	ScienceDirect	IEEE Xplore
Search query	("Feature selection" OR "variable selection" OR "feature reduction" OR "variable reduction") NEAR ("Fuzzy regres*" OR "Fuzzy classif*" OR "fuzzy learn*" OR "Fuzzy model" OR "Fuzzy inference")	(“Feature selection” OR “variable selection” OR “feature reduction” OR “variable reduction”) AND (“Fuzzy regression” OR “Fuzzy classifier” OR “Fuzzy learning” OR “Fuzzy model” OR “Fuzzy inference”)	("Abstract":"feature selection" OR "Abstract":"variable selection" OR "Abstract":"variable reduction" OR "Abstract":"feature reduction") AND ("Abstract":"fuzzy clas*" OR "Abstract":"fuzzy regres*" OR "Abstract":"fuzzy model*" OR "Abstract":"fuzzy learn*" OR "Abstract":"fuzzy inference")
Procedure	Go to the SpringerLink website: https://link.springer.com/ . Add the search query to the search box an press ‘search’. Add the ‘date published’ filter to show documents between 2000 and 2021. Add the Language filter for ‘English’.	Go to the ScienceDirect website: https://www.sciencedirect.com/ . Click on ‘advanced search’ and click on ‘show all fields’. Add the search query in the search box for ‘Title, abstract or author-specified keywords’. Fill in the ‘Year(s)’ box with 2000-2021. Now press the ‘search’ button.	Go to the IEEE Xplore website: https://ieeexplore.ieee.org/Xplore/home.jsp . Add the search query to the search box an press ‘search’. Add the filter for the year on 2000-2021.
Results	202	142	136

Inclusion and exclusion criteria

- The article discusses a feature selection method.
    - The focus of this review is to find existing feature selection methods. Therefore, when the article is not discussing any type of feature selection, it is considered irrelevant for this review.
- There must be an indication that the feature selection method is appropriate for a fuzzy model.
    - This indication could be that the feature selection method is executed on the input data prior to the use of a fuzzy logic prediction model. Or the feature selection method is integrated in the fuzzy model. There must be some kind of indication that the feature selection method would be appropriate for a fuzzy model, otherwise the article is considered irrelevant to this literature review.
- The mechanism of the feature selection method is explained.
    - An explanation of the feature selection method is needed to understand how the method works. If there is no explanation of the mechanism of the method, the article is considered insufficient for this review.
- There is a discussion in the article about the performance, pros and/or cons of the feature selection method.
    - Information on the performance of the feature selection method is desired to be able to compare the methods. However, comparing the feature selection methods solely on the performance described in the articles is not an easy job, due to the difference in datasets, prediction models, parameters, and performance metrices. Also, the performance of the method says something about the quality of the method. If the performance is not mentioned, or there is bad performance, the feature selection method is not worthy of being included in this literature review.
- The proposed feature selection method may not make the model more complex.
    - The objective of the dissertation is to use feature selection to make the model more interpretable. When the feature selection method reduces the interpretability of the model, this method is not included in this review. An example of this would be a feature extraction method which is creating new features to use in the fuzzy model. When these new features do not have meaning to the user of the model, this will reduce the model's interpretability.
- The proposed feature selection method may not be similar to the proposed method in another document.
    - In case of similarity of prosed methods only the best document is included in the literature review to avoid redundancy. However, if the documents complement each other, they may both be included in the literature review.
- The article was written in English.
    - English is the universal language of science. Therefore, this criteria is a must to make this review understandable for as many people as possible.
- The article was published after 2000.
    - A time window makes the search for the relevant articles more efficient. Articles that were published more than twenty years ago have probably lost its

relevance, because the research field of data science and data preprocessing is developing very quickly.

Relevance criteria

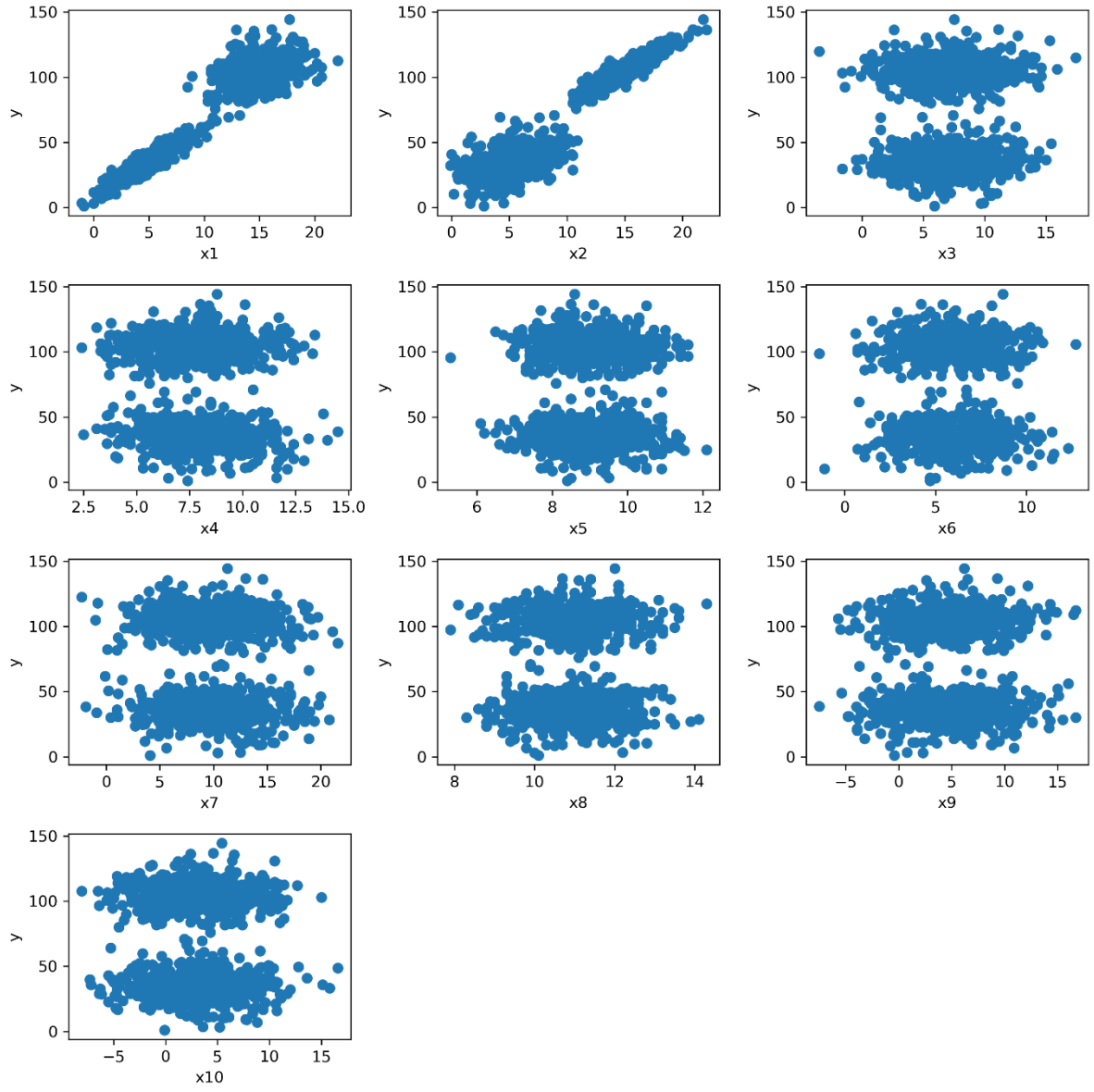
This process of evaluating the relevance is performed in two rounds. In the first round the documents are evaluated by its title. The document will pass this round of evaluation if it meets one of the following two criteria: 1) the title contains one or more search terms or highly related terms, or 2) the title indicates a link to the research topic. All documents that meet this criteria are taken to the second evaluation round. In the second round, the documents are evaluated by its abstract. The criteria for the abstract are more subjective than the criteria for the title. The abstract has to indicate relevance to the research topic. If it does not seem relevant, the document is removed from the selection. If there is doubt about the relevance of the document, in the first or second evaluation round, the document is kept in the selection. This will reduce the risk of accidently neglecting relevant documents.

Quality criteria

Three criteria to ensure quality:

- The first criteria is the relevance of the journal or book the document was published in. For example, when the journal is titled ‘Agriculture’ it is likely that the document does not contain the highest quality feature selection method. However, when the journal is titled ‘Computational Intelligence’, it is more likely the feature selection method is of quality. The documents that were not published in a relevant journal or book are removed from the selection for this literature review.
- The second criteria is the number of citations of a document. The number of citations gives an indication of the quality of a document. However, a lack of citations does not necessarily mean the document is of poor quality. It could be the result of a document being very specific or recently published. Therefore, there is no threshold for the minimum number of citation to be included in the selection for the literature review. However, the number of citations is taken into account when selecting the documents for this literature review.
- The third criteria is the scientific quality of the document. If the document is lacking validity or reliability, then the document is removed from the selection.

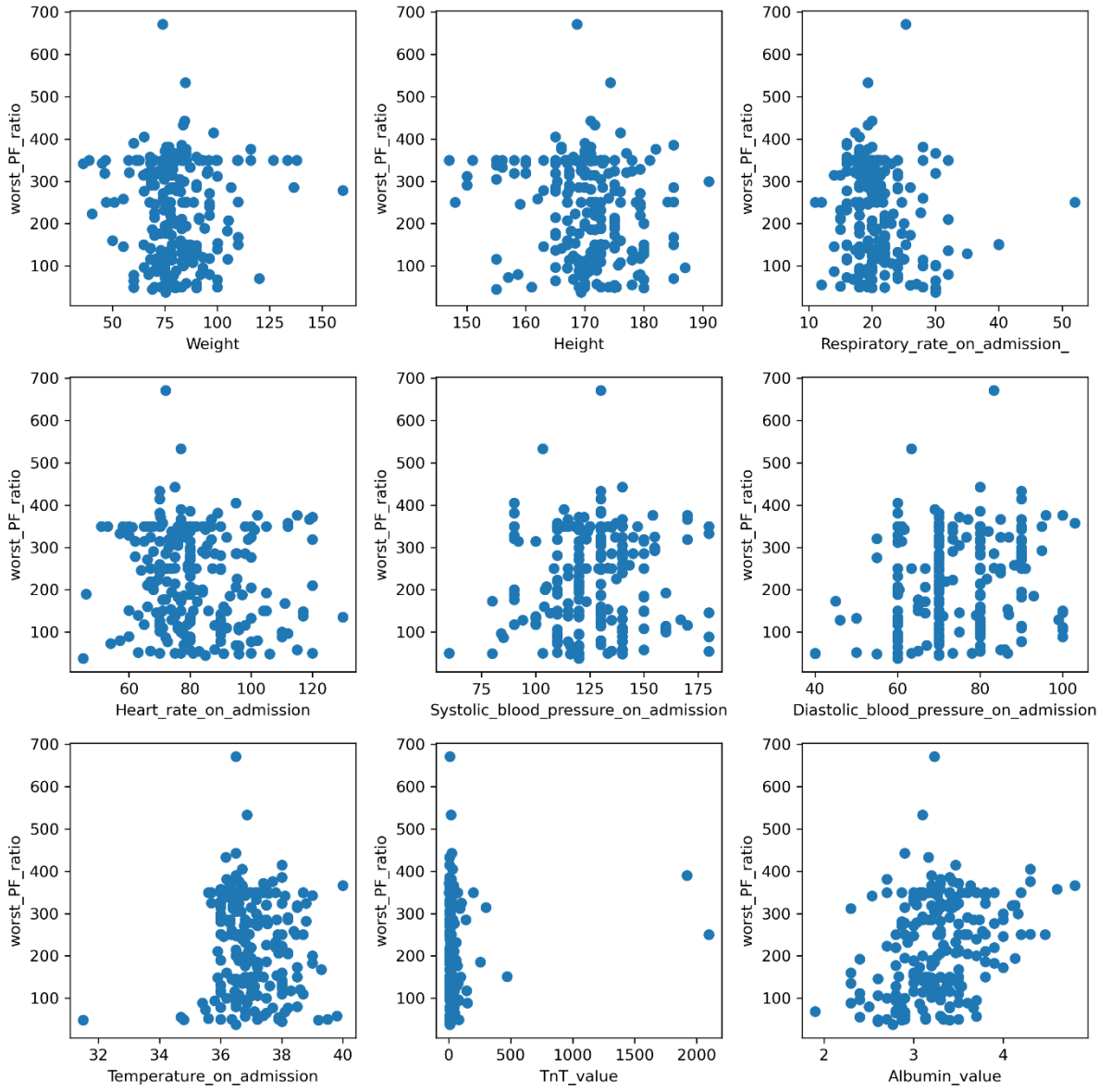
C. Scatter plots artificial dataset

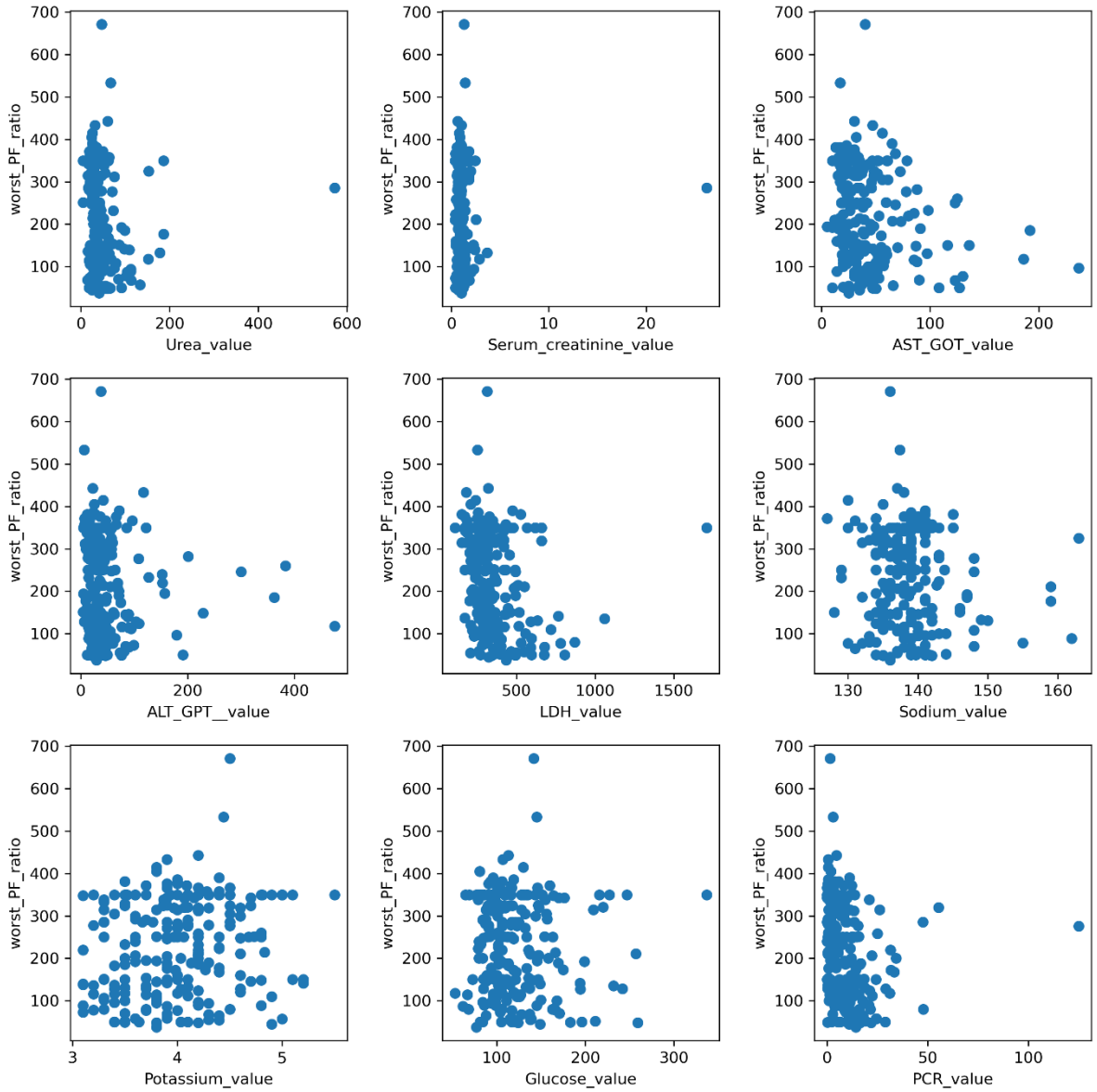


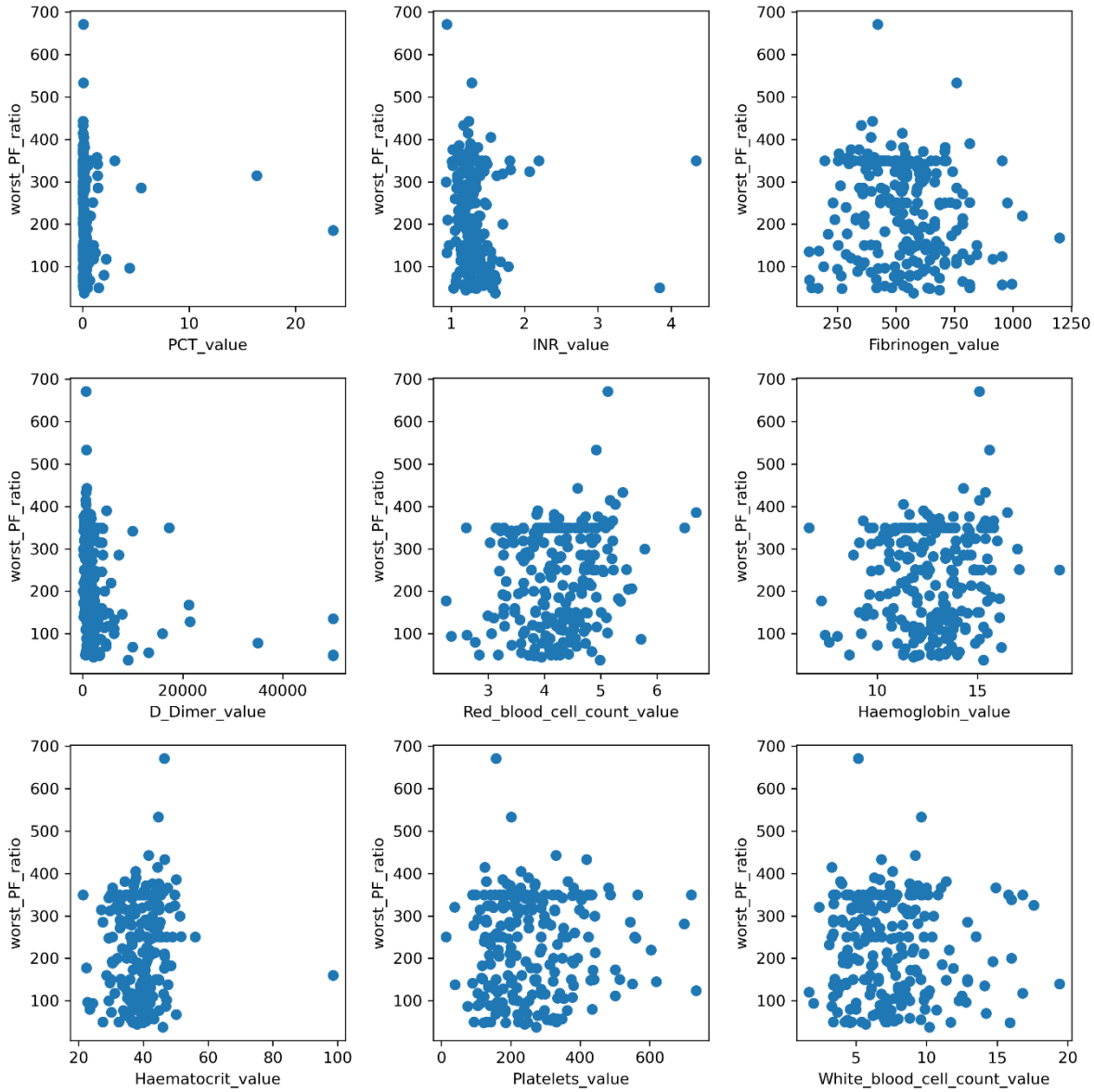
D. Features of COVID-19 dataset

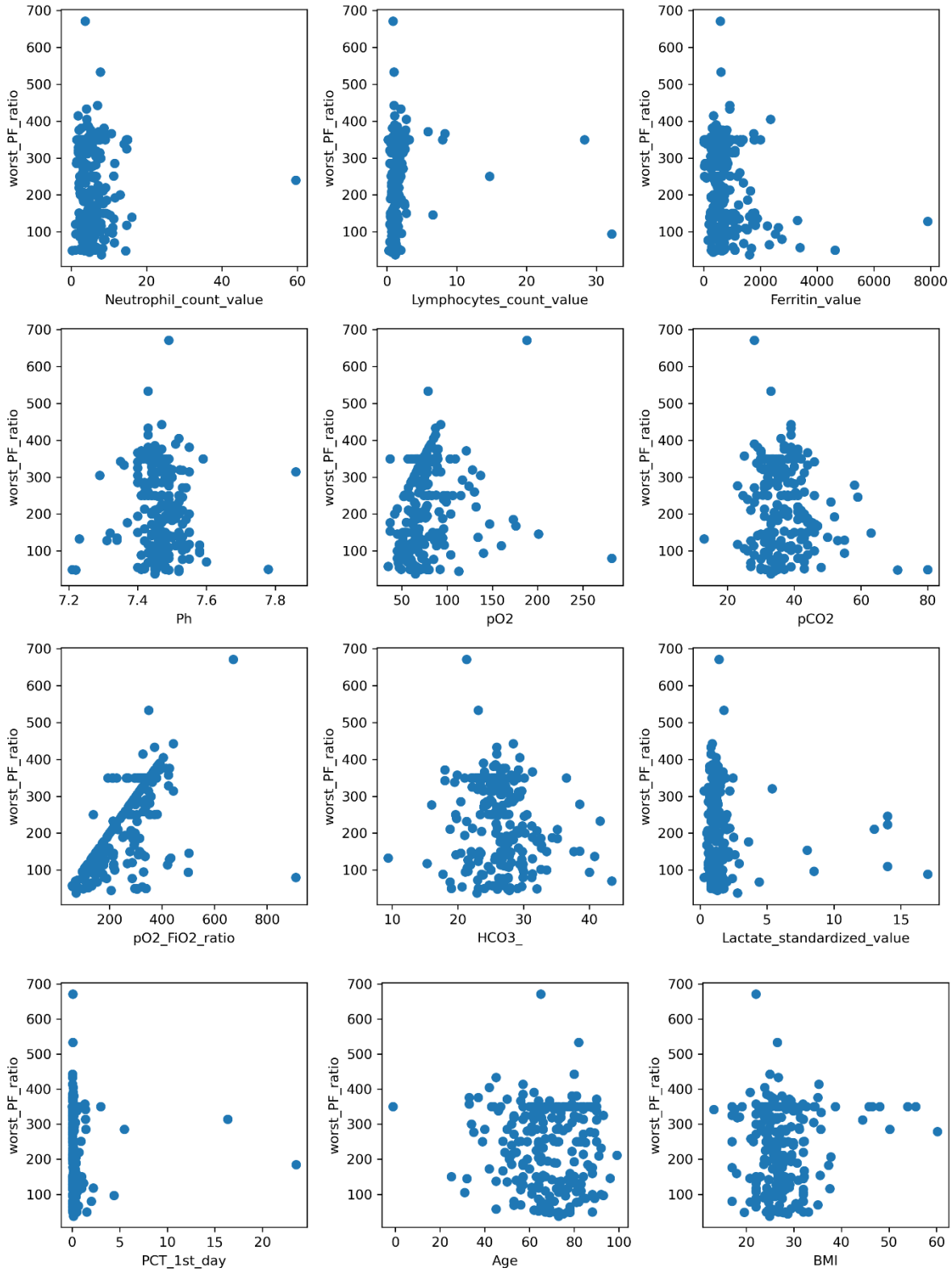
Feature	Feature name
1	Weight
2	Height
3	Respiratory_rate_on_admission_
4	Heart_rate_on_admission
5	Systolic_blood_pressure_on_admission
6	Diastolic_blood_pressure_on_admission
7	Temperature_on_admission
8	TnT_value
9	Albumin_value
10	Urea_value
11	Serum_creatinine_value
12	AST_GOT_value
13	ALT_GPT__value
14	LDH_value
15	Sodium_value
16	Potassium_value
17	Glucose_value
18	PCR_value
19	PCT_value
20	INR_value
21	Fibrinogen_value
22	D_Dimer_value
23	Red_blood_cell_count_value
24	Haemoglobin_value
25	Haematocrit_value
26	Platelets_value
27	White_blood_cell_count_value
28	Neutrophil_count_value
29	Lymphocytes_count_value
30	Ferritin_value
31	Ph
32	pO2
33	pCO2
34	pO2_FiO2_ratio
35	HCO3_
36	Lactate_standardized_value
37	PCT_1st_day
38	Age
39	BMI
Target variable	worst_PF_ratio

E. Scatter plots COVID-19 dataset

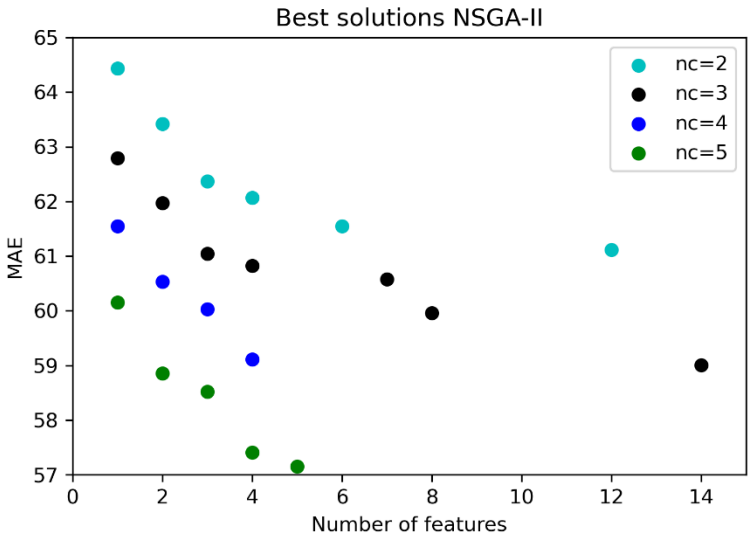


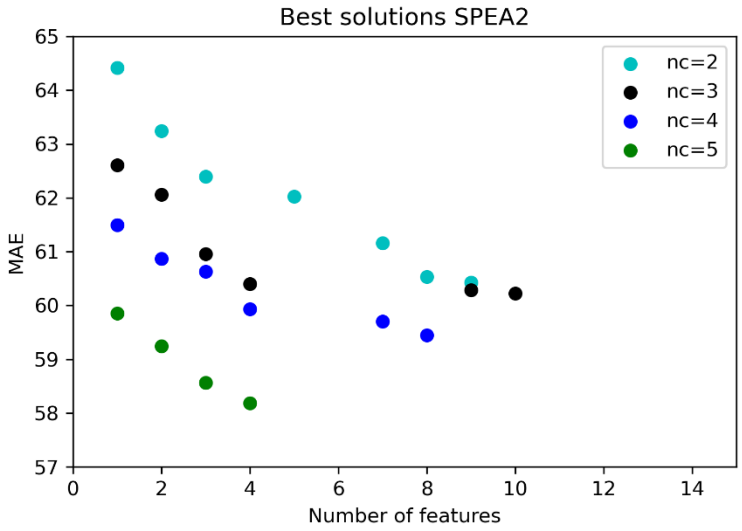

worst_PF_ratio
Urea_value
Serum_creatinine_value
AST_GOT_value
ALT_GPT__value
LDH_value
Sodium_value
Potassium_value
Glucose_value
PCR_value

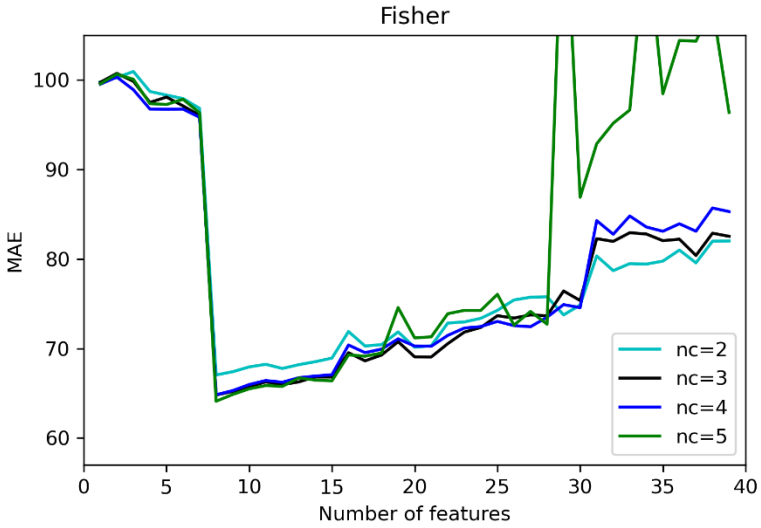

worst_PF_ratio
PCT_value
INR_value
Fibrinogen_value
D_Dimer_value
Red_blood_cell_count_value
Haemoglobin_value
Haematocrit_value
Platelets_value
White_blood_cell_count_value


worst_PF_ratio
Neutrophil_count_value
Lymphocytes_count_value
Ferritin_value
Ph
pO2
pCO2
pO2_FiO2_ratio
HCO3_
Lactate_standardized_value
PCT_1st_day
Age
BMI

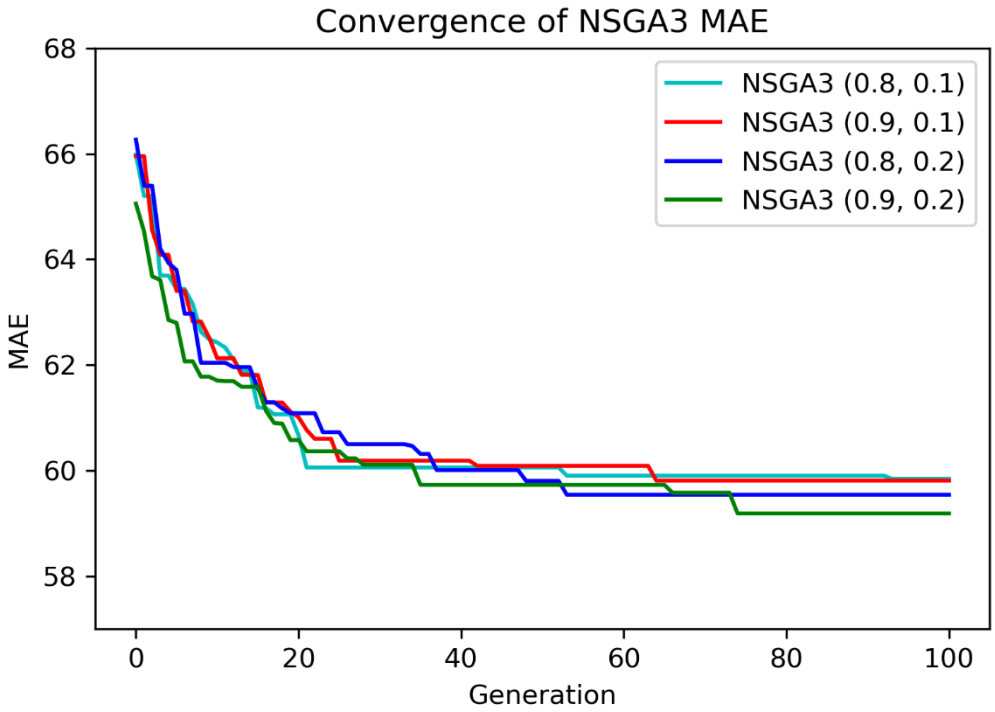
F. Results for NSGA-II, SPEA2, and Fisher score with 2 to 5 clusters

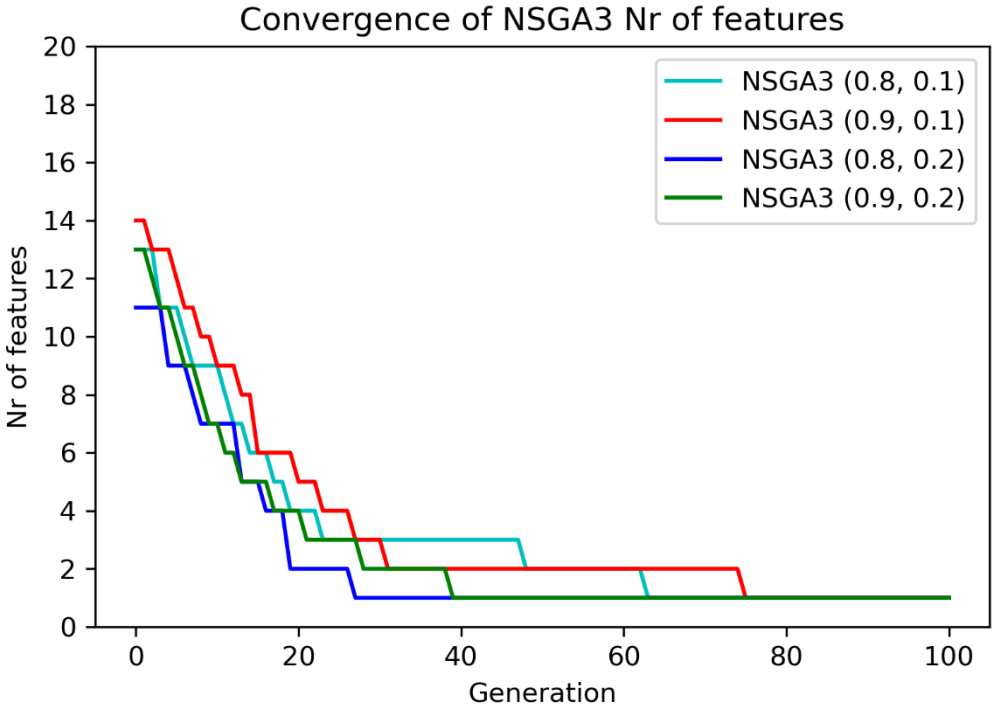


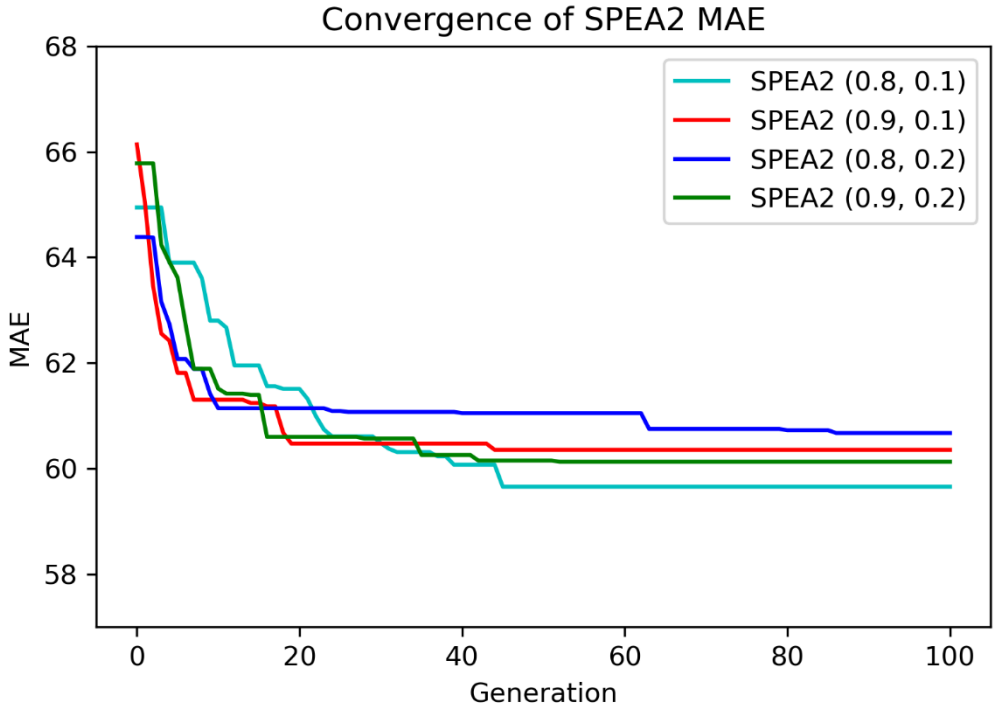


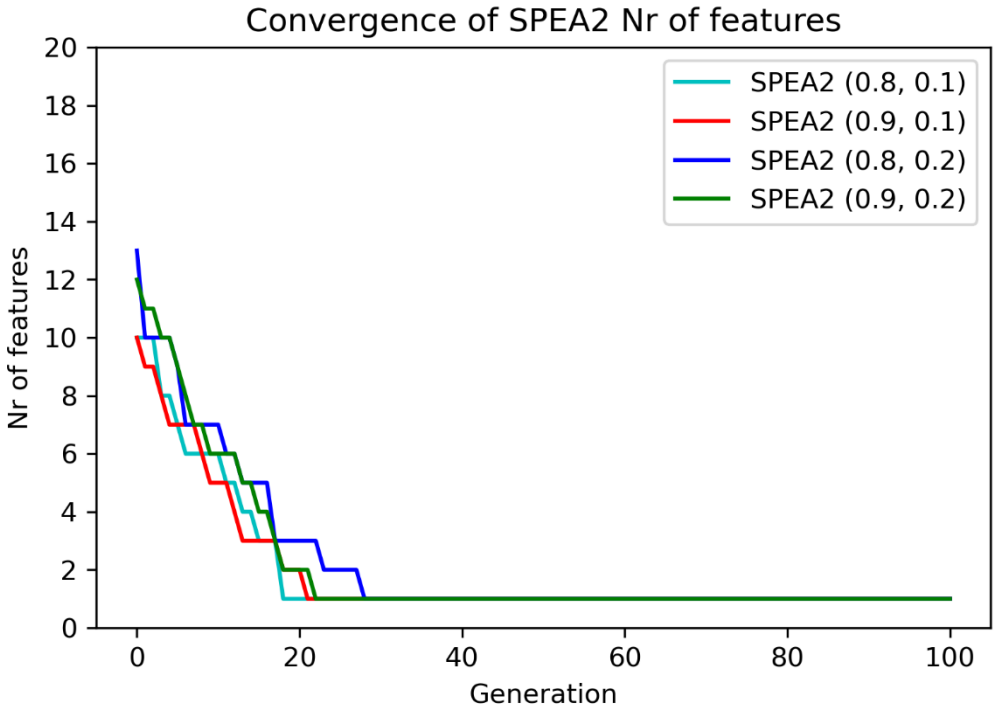


G. Convergence graphs of NSGA-III and SPEA2



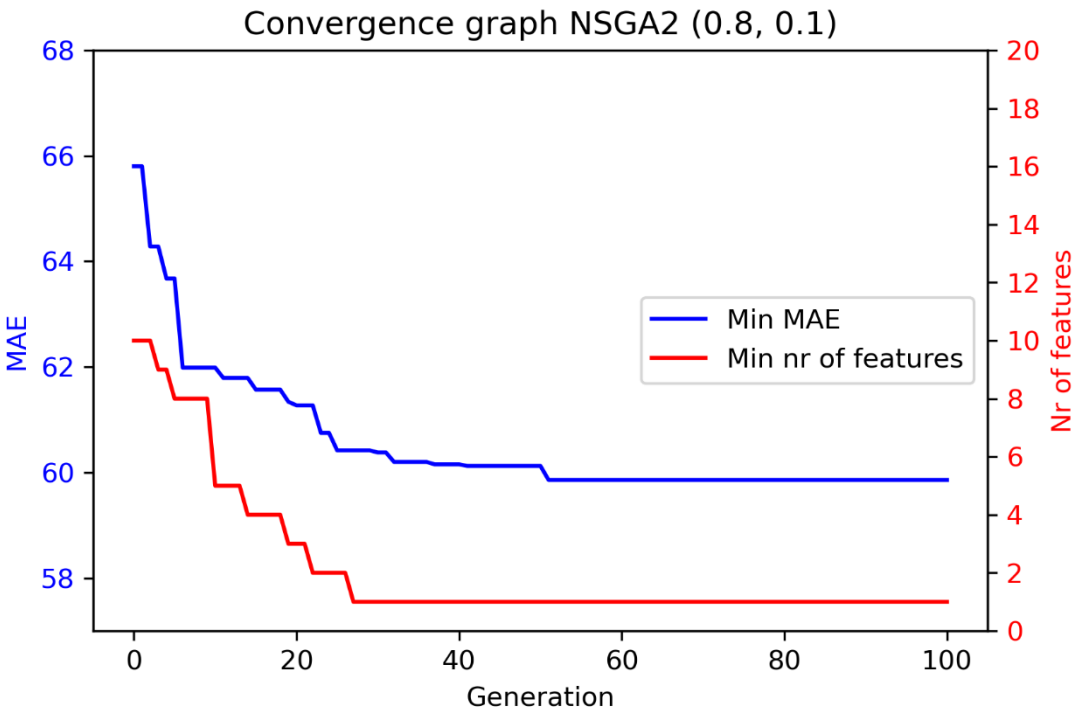


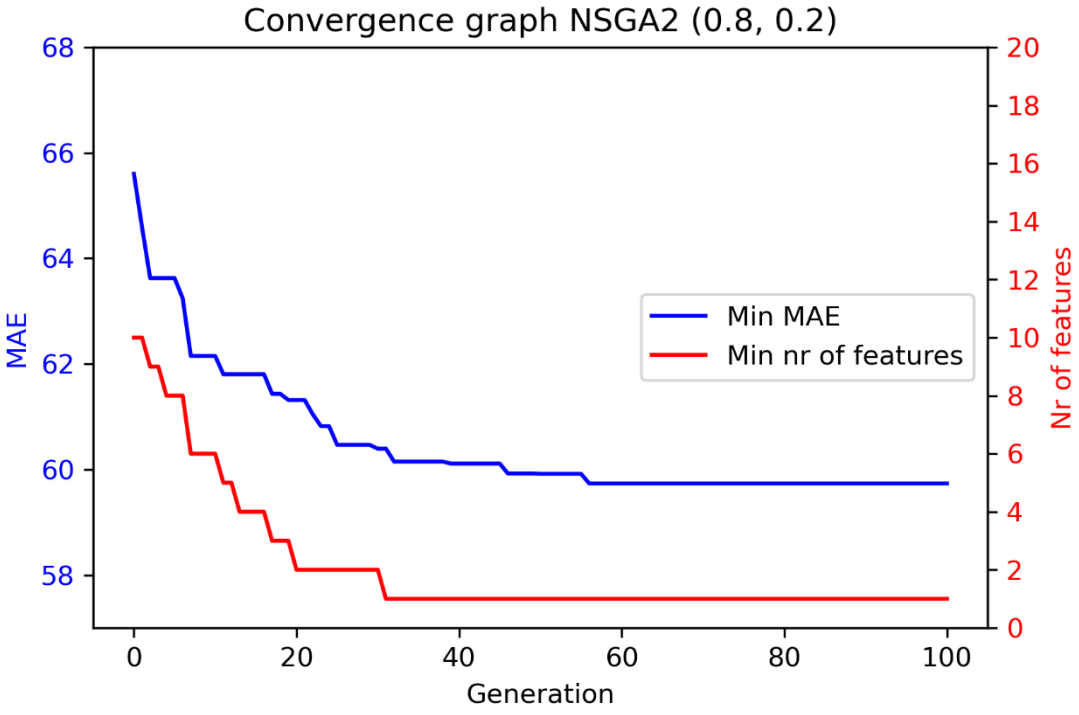

Convergence of SPEA2 MAE
SPEA2 (0.8, 0.1)
SPEA2 (0.9, 0.1)
SPEA2 (0.8, 0.2)
SPEA2 (0.9, 0.2)
MAE
Generation
68
66
64
62
60
58
0
20
40
60
80
100

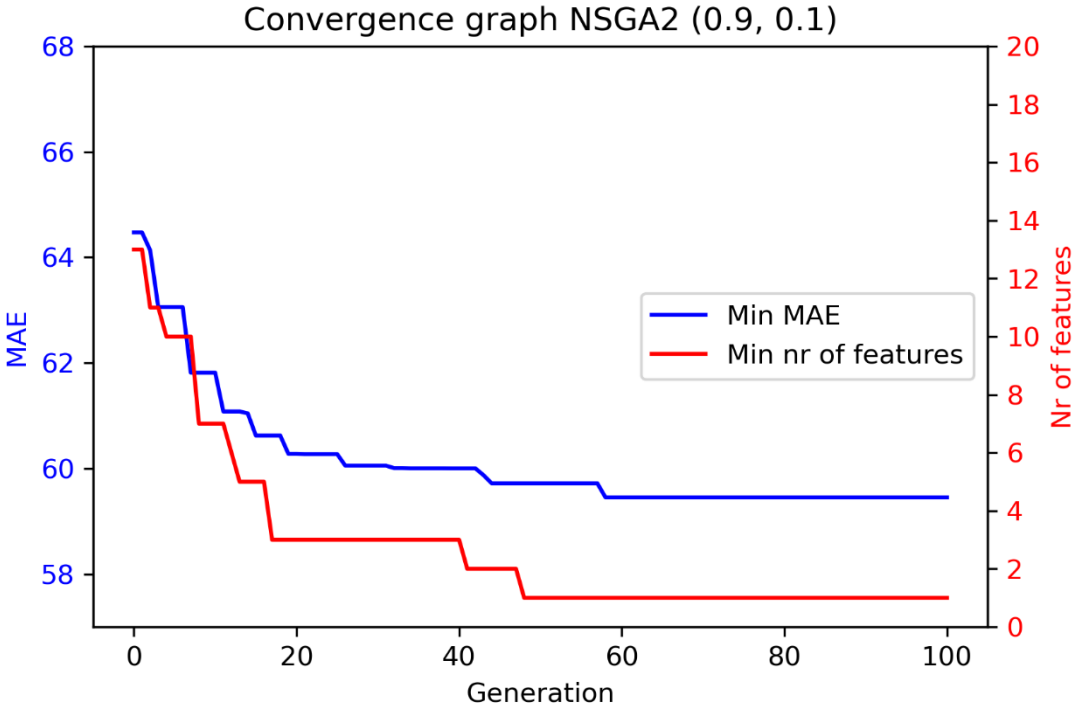

Convergence of SPEA2 Nr of features
SPEA2 (0.8, 0.1)
SPEA2 (0.9, 0.1)
SPEA2 (0.8, 0.2)
SPEA2 (0.9, 0.2)
Nr of features
Generation
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100

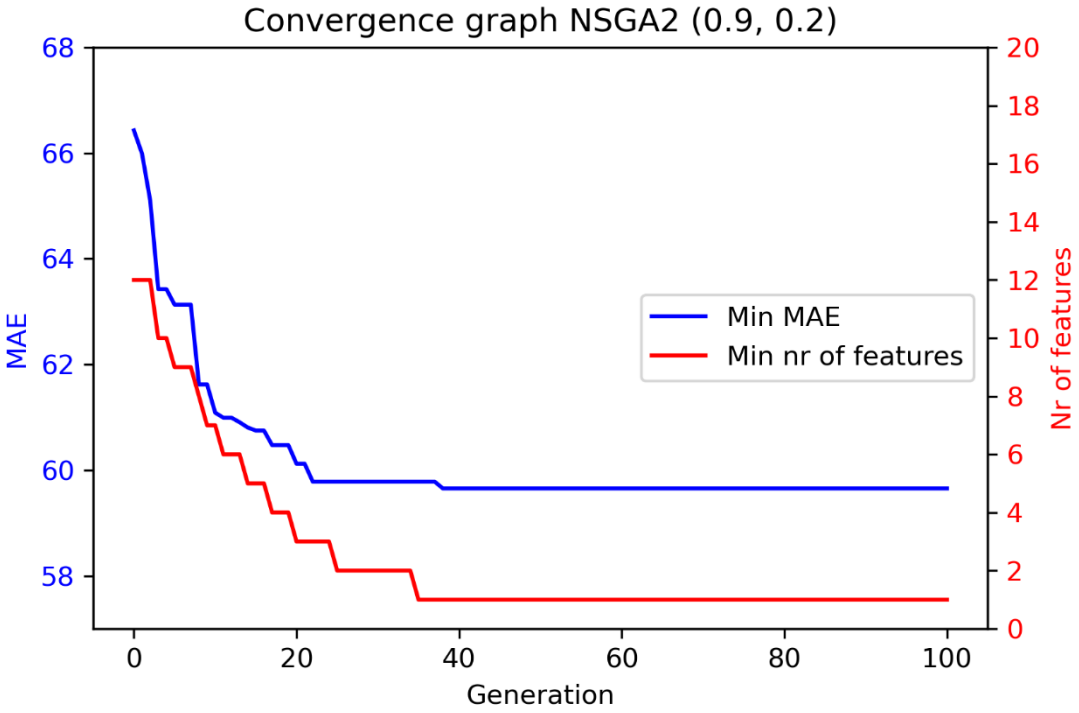
H. Convergence graphs per model

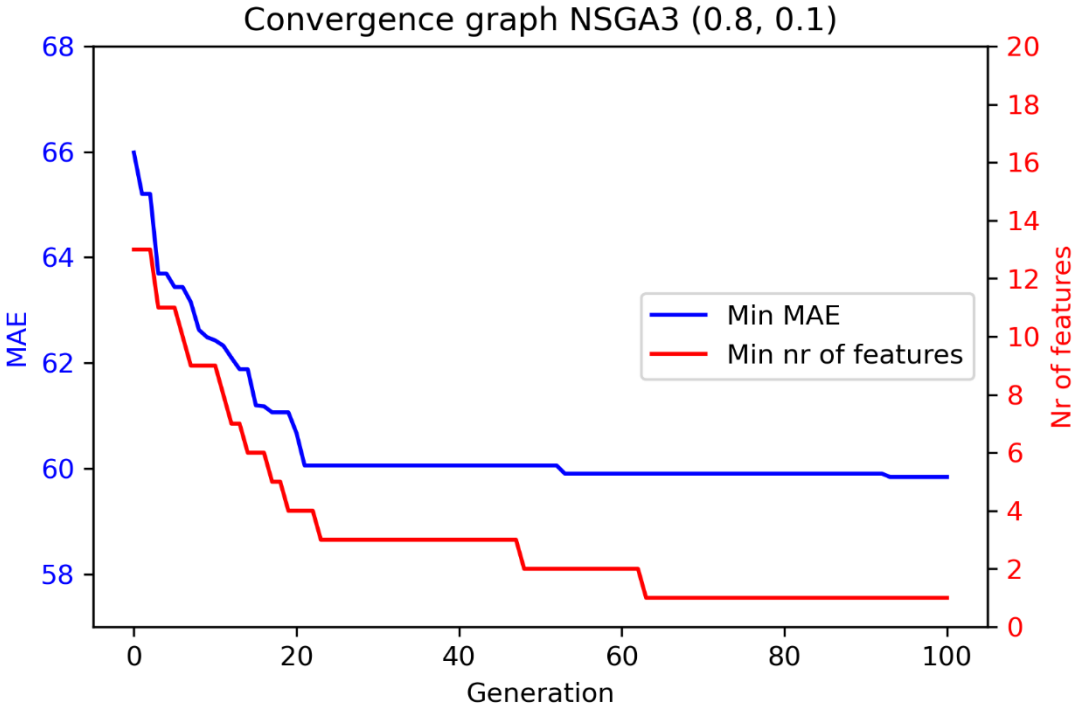
The convergence graphs of the feature selection algorithms NSGA-II, NSGA-III, and SPEA2 are presented below. The title of the graphs contain information about the parameters: CXPB and MUTPB.

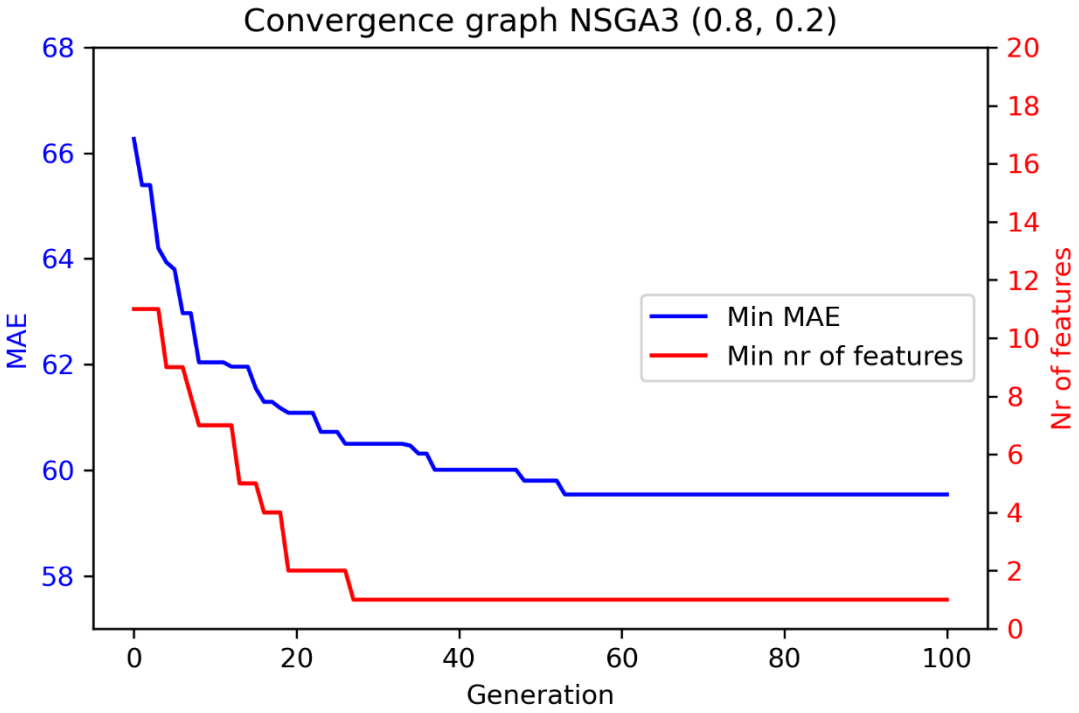


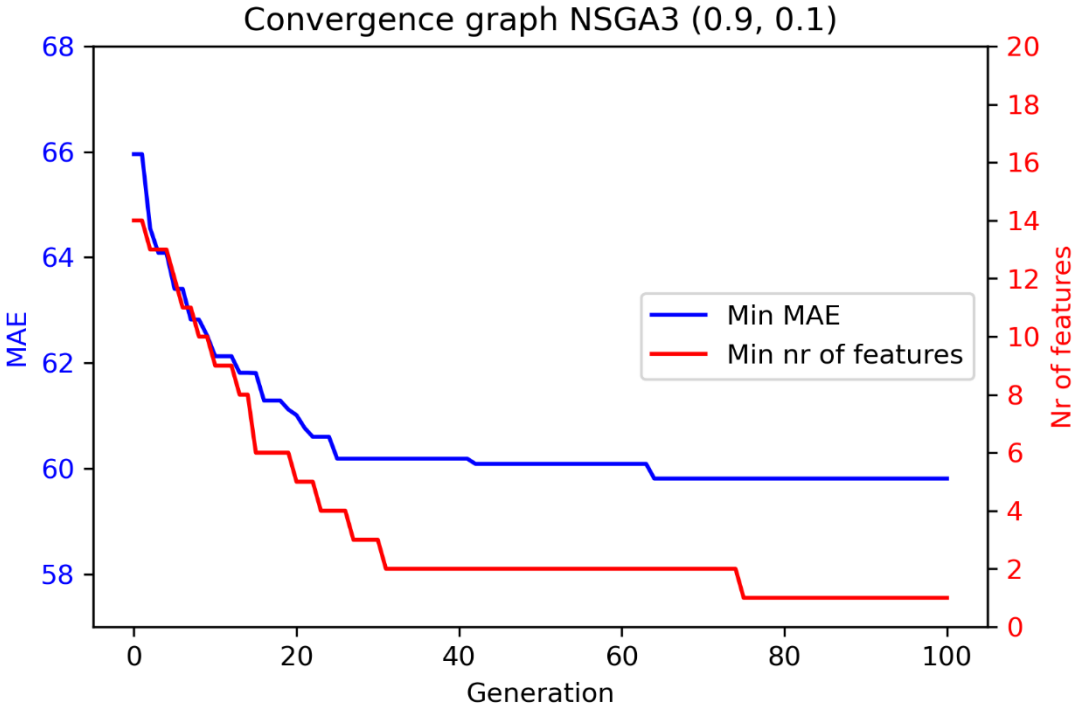


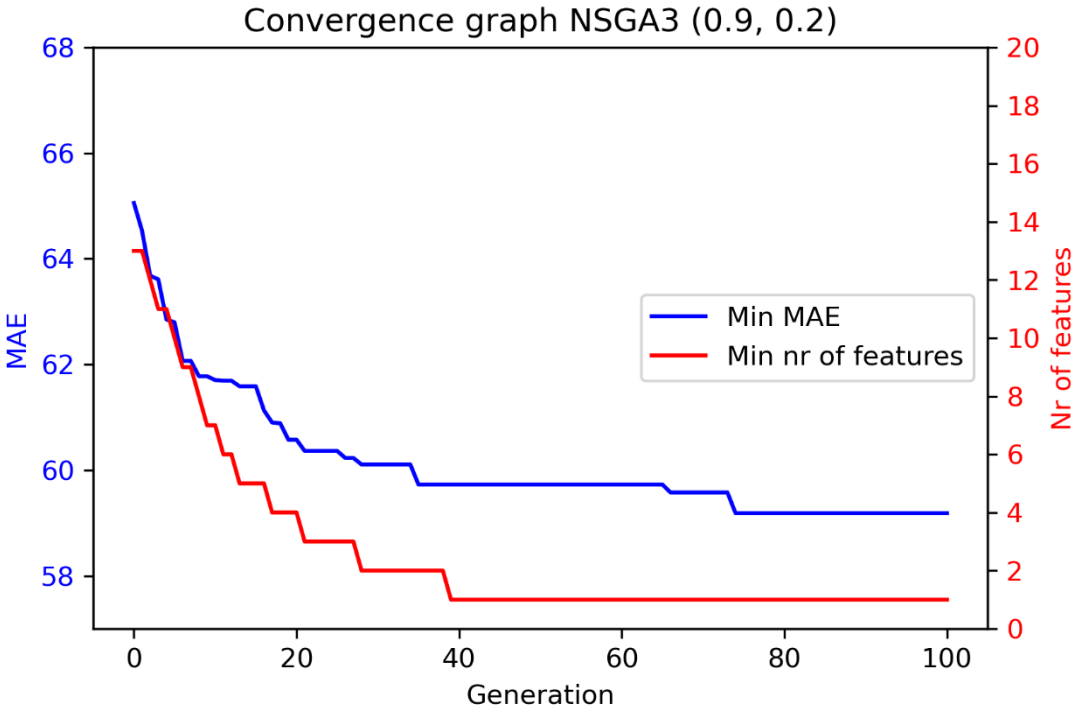

Convergence graph NSGA2 (0.9, 0.1)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100

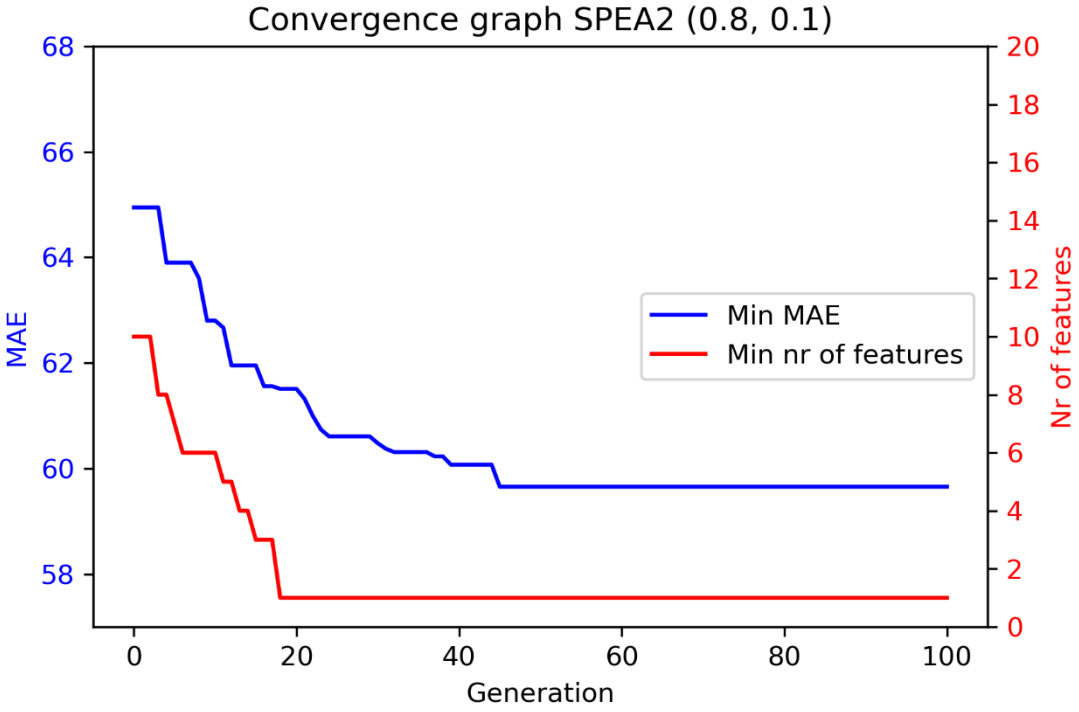

Convergence graph NSGA2 (0.9, 0.2)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100

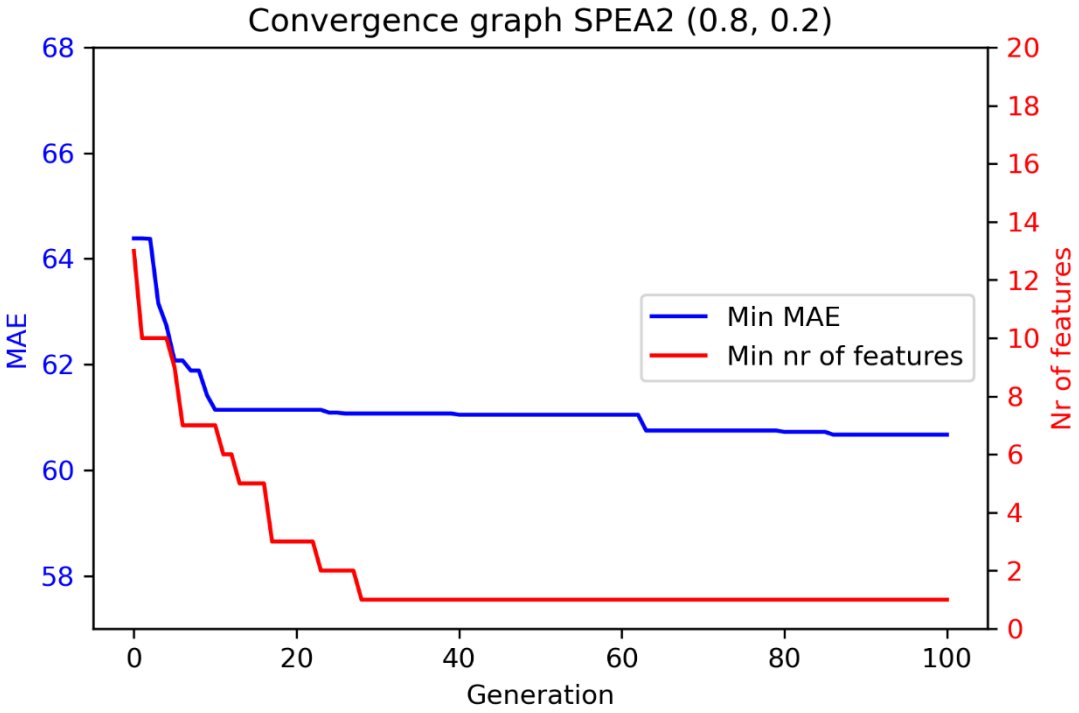

Convergence graph NSGA3 (0.8, 0.1)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100

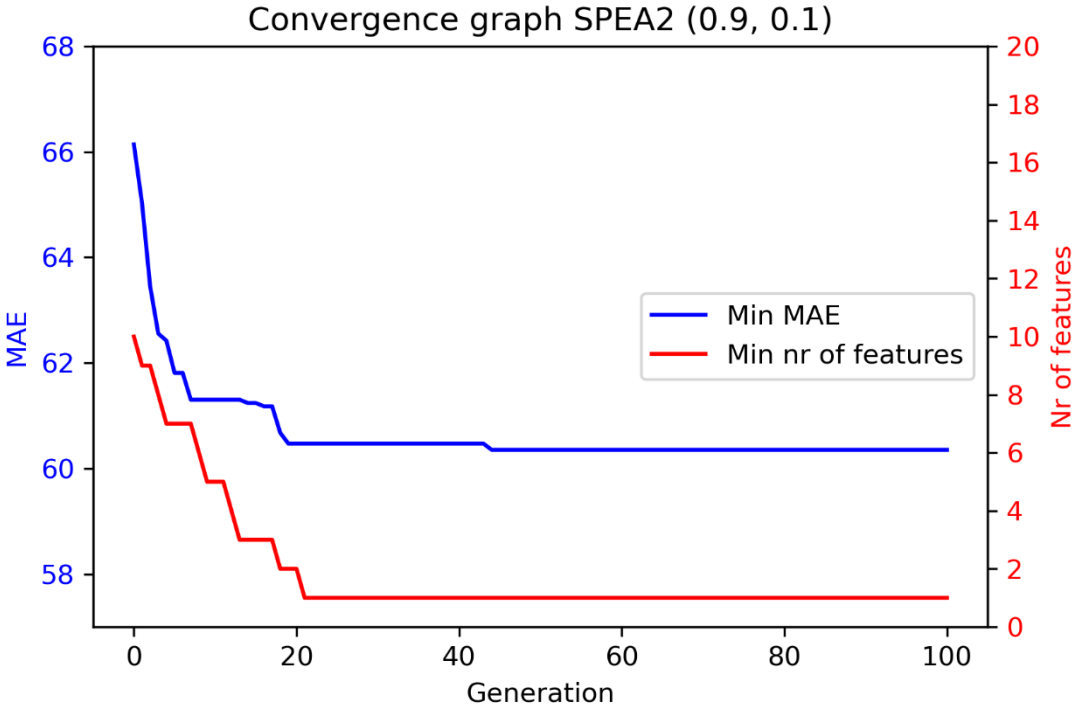

Convergence graph NSGA3 (0.8, 0.2)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100

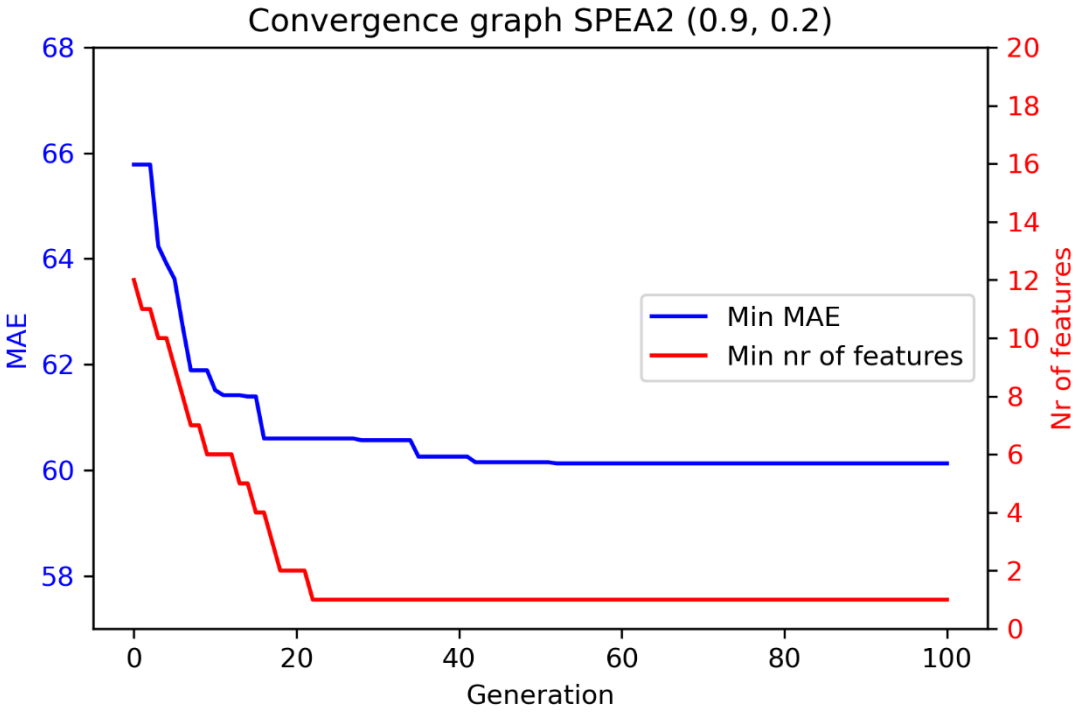

Convergence graph NSGA3 (0.9, 0.1)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100


Convergence graph NSGA3 (0.9, 0.2)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100


Convergence graph SPEA2 (0.8, 0.1)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100


Convergence graph SPEA2 (0.8, 0.2)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100


Convergence graph SPEA2 (0.9, 0.1)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100


Convergence graph SPEA2 (0.9, 0.2)
MAE
Nr of features
Generation
Min MAE
Min nr of features
68
66
64
62
60
58
20
18
16
14
12
10
8
6
4
2
0
0
20
40
60
80
100