Evolutionary Machine Learning for Classification with Incomplete Data

by

Cao Truong Tran

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2018

Abstract

Classification is a major task in machine learning and data mining. Many real-world datasets suffer from the unavoidable issue of missing values. Classification with incomplete data has to be carefully handled because inadequate treatment of missing values will cause large classification errors.

Existing most researchers working on classification with incomplete data focused on improving the effectiveness, but did not adequately address the issue of the efficiency of applying the classifiers to classify unseen instances, which is much more important than the act of creating classifiers. A common approach to classification with incomplete data is to use imputation methods to replace missing values with plausible values before building classifiers and classifying unseen instances. This approach provides complete data which can be then used by any classification algorithm, but sophisticated imputation methods are usually computationally intensive, especially for the application process of classification. Another approach to classification with incomplete data is to build a classifier that can directly work with missing values. This approach does not require time for estimating missing values, but it often generates inaccurate and complex classifiers when faced with numerous missing values. A recent approach to classification with incomplete data which also avoids estimating missing values is to build a set of classifiers which then is used to select applicable classifiers for classifying unseen instances. However, this approach is also often inaccurate and takes a long time to find applicable classifiers when faced with numerous missing values.

The overall goal of the thesis is to simultaneously improve the effectiveness and efficiency of classification with incomplete data by using evo-

lutionary machine learning techniques for feature selection, clustering, ensemble learning, feature construction and constructing classifiers.

The thesis develops approaches for improving imputation for classification with incomplete data by integrating clustering and feature selection with imputation. The approaches improve both the effectiveness and the efficiency of using imputation for classification with incomplete data.

The thesis develops wrapper-based feature selection methods to improve input space for classification algorithms that are able to work directly with incomplete data. The methods not only improve the classification accuracy, but also reduce the complexity of classifiers able to work directly with incomplete data.

The thesis develops a feature construction method to improve input space for classification algorithms with incomplete data by proposing interval genetic programming—genetic programming with a set of interval functions. The method improves the classification accuracy and reduces the complexity of classifiers.

The thesis develops an ensemble approach to classification with incomplete data by integrating imputation, feature selection and ensemble learning. The results show that the approach is more accurate, and faster than previous common methods for classification with incomplete data.

The thesis develops interval genetic programming to directly evolve classifiers for incomplete data. The results show that classifiers generated by interval genetic programming can be more effective and efficient than classifiers generated the combination of imputation and traditional genetic programming. Interval genetic programming is also more effective than common classification algorithms able to work directly with incomplete data.

In summary, the thesis develops a range of approaches for simultaneously improving the effectiveness and efficiency of classification with incomplete data by using a range of evolutionary machine learning techniques.

List of Publication

Refereed journal papers:

1. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue. "Improving performance for classification with incomplete data using wrapper-based feature selection". Evolutionary Intelligence Journal 9(3): 81-94 (2016).

2. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, Lam Thu Bui."Improving Performance of Classification on Incomplete Data Using Feature Selection and Clustering". Applied Soft Computing 2017 (Minor Revision).

3. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, Lam Thu Bui. "An Effective and Efficient Approach to Classification with Incomplete Data". Knowledge-Based Systems, Vol.154, 8/2018, pp 1–16, Elsevier .

Refereed conference papers:

1. Cao Truong Tran, Peter Andreae, Mengjie Zhang. "Impact of imputation of missing values on Genetic programming based multiple feature construction for classification". Proceedings of the IEEE Congress on Evolutionary Computation 2015 (CEC 2015): 2398-2405.

2. Cao Truong Tran, Mengjie Zhang, Peter Andreae. "Multiple Imputation for Missing Data Using Genetic Programming". Proceed-

ings of the Genetic and Evolutionary Computation Conference 2015 (GECCO 2015): 583-590.

3. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue. "Directly Constructing Multiple Features for Classification with Missing Data using Genetic Programming with Interval Functions". Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO 2016): 69-70.

4. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue. "A Wrapper Feature Selection Approach to Classification with Missing Data". Proceedings of the 19^{th} European Conference on the Applications of Evolutionary Computation (EvoApplication 2016) (1) 2016: 685-700.

5. Cao Truong Tran, Mengjie Zhang, Peter Andreae. "A Genetic Programming-Based Imputation Method for Classification with Missing Data". Proceedings of the 19^{th} European Conference on Genetic Programming (EuroGP 2016): 149-163.

6. Cao Truong Tran, Mengjie Zhang, Peter Andreae. "Directly evolving classifiers for missing data using genetic programming". Proceedings of the IEEE Congress on Evolutionary Computation 2016 (CEC 2016): 5278-5285.

7. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue and Lam Thu Bui. "Multiple Imputation and Ensemble Learning for Classification with Incomplete Data". Proceedings of the 20^{th} Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES 2016): 401-415 (**Best Paper Award**).

8. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue. "Genetic programming based feature construction for classification with in-

complete data". Proceedings of the Genetic and Evolutionary Computation Conference 2017 (GECCO 2017): 1033-1040.

9. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue. "Multiple imputation and genetic programming for classification with incomplete data". Proceedings of the Genetic and Evolutionary Computation Conference 2017 (GECCO 2017): 521-528.

10. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue. "Bagging and Feature Selection for Classification with Incomplete Data". Proceedings of the 20^{th} European Conference on the Applications of Evolutionary Computation (EvoApplication 2017): 471-486.

11. Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, Lam Thu Bui. "An Ensemble of Rule-based Classifiers for Incomplete Data". Proceedings of the 21^{st} Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES 2017): 7-12.

Acknowledgments

I would like to express my gratitude to those who gave me the assistance and support I needed to complete this thesis.

I would like to express my very great appreciation to my supervisors Prof. Mengjie Zhang and A/Prof.Peter Andreae. Prof. Mengjie has spent many dedicated hours and efforts to train my research skills and provide encouraging and challenging feedbacks to improve my research work. A/Prof.Peter has spent a lot of time to help me improve my research writing skills, and the discussions with him are a great source for novel ideas. I also would like to thank Dr. Bing Xue who has read my research papers and given me useful feedbacks.

This research was funded under Vietnamese Government Scholarship and Victoria University of Wellington Scholarship (VUW-VIED Joint Scholarship). I would like to express my thank to Vietnamese Government and Victoria University of Wellington for their invaluable funding.

I would like to offer my thanks to my friends in the Evolutionary Computation Research Group (ECRG) for creating an active and interesting research environment.

Last, but most important, I would like to dedicate this work to my family, to my parents, Tran Cao Hai and Nguyen Thi Hang for working so hard to bring up their son, to my wife, Vu Anh Nguyet for sharing a lot of happiness and difficulty in the life with me and to my children, Tran Cao Kien, and Tran Minh Ngoc for trying to grow up without their dad taking care for the past four years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces the thesis. It firstly discusses the problem statement, then presents the motivations, the search goals, the major contributions and the organisation of the thesis.

1.1 Problem Statement

Classification is a data mining task that predicts a class label for an instance based on feature values of the instance. Classification includes two main processes: a training process and an application (test) process. The goal of the training process is to use a classification algorithm on a training dataset to build a classifier. The goal of the application process is to use the built classifier to assign a class label to new instances. Classification has been widely applied to many areas such as computer science, engineering and medicine [38, 67, 177]. However, there are still issues, one of which is incomplete data [1, 13, 45, 57, 109, 139].

Missing values where the values of some features are unknown are a common example of incomplete data in many real-world datasets. For example, in the UCI machine learning repository [8], which is one of the most popular benchmarks for data mining, 45% of the datasets suffer from missing values [57]. There are various causes of missing values. For exam-

ple, in social surveys, respondents often refuse to answer some questions, so datasets collected from the surveys are incomplete [107, 138, 169]. Medical datasets usually contain a large number of missing values because not all possible tests can be done on every patient [75, 151, 178]. Industrial databases also often suffer from missing values due to mechanical failures during data collection [93].

Missing values cause serious problems for classification. One of the most serious problems is that the majority of classification algorithms do not work on datasets with missing values (incomplete datasets) [13, 57, 102]. For example, neural networks cannot directly work with incomplete data. Another problem is that missing values often lead to big classification error due to inadequate information for the training and application processes [45, 102, 109].

In statistical analysis, the problem of missing data has been tackled extensively [41, 61, 100, 141] and also, but with less effort, in the classification literature [57]. There are three common approaches to classification with incomplete data [13, 45, 57, 139]. The first approach is to delete all instances containing missing values. The main benefit of this approach is to provide complete data for classification. However, the deletion approach is only feasible for data with few missing values because it cannot provide enough information when data has numerous missing values. Moreover, this approach cannot classify new incomplete instances in the application process. Therefore, the deletion approach can be only used on a limited number of datasets [1, 13, 46]. The second approach is to use imputation methods to transform incomplete data into complete data before building a classifier in the training process or classifying a new incomplete instance in the application process. This approach can provide complete data which can then be used by any classification algorithm. It also can deal with incomplete data with a large number of missing values. Therefore, imputation is perhaps the most popular approach to classification with missing data [45, 57, 109]. However, imputation methods take time

to estimate missing values, especially with powerful imputation methods [46]. The third approach is to construct a classifier in the training process which can directly classify incomplete instances in the application process without requiring any imputation method [105, 125, 129]. This approach can save time for estimating missing values. However, existing methods are often not accurate and expensive when data contain a large number of missing values. Therefore, how to effectively and efficiently handle classification with missing data should be further investigated.

Evolutionary computation (EC) is a subfield of computational intelligence that uses computational models of evolutionary processes as the key elements in design and implementation. Two major subsets of EC are evolutionary algorithms and swarm intelligence algorithms [48]. Evolutionary algorithms are inspired by biological evolution, such as selection, recombination, mutation and reproduction. Popular evolutionary algorithms include genetic algorithms (GAs) [5] and genetic programming (GP) [87]. Swarm intelligence algorithms are inspired by the collective intelligence that emerges in the natural environment such as colonies of ants, and schools of fish, birds and flocks. Popular swarm algorithms include particle swarm optimization (PSO) [84], ant colony optimization (ACO) [37], and artificial bee colony algorithm (ABC) [82].

EC techniques have been widely applied to improve the classification with complete data. For example, GP has been used to evolve high quality classifiers and to construct new features for classification [42]. Another example is that GA and PSO have been used to select suitable feature subsets for classification [183]. However, EC techniques such as GP, GA and PSO have not been systematically explored for classification with incomplete data. Therefore, how to apply EC techniques to improve the effectiveness and efficiency of classification with incomplete data should be further investigated.

1.2 Motivations

Classification with incomplete data has to handle two different problems: handling missing values and classification [57]. Therefore, two obvious approaches to improving performance of classification with incomplete data are to improve the input by transforming the data so that it has fewer missing values and to create classifiers that are more tolerant of missing values [57].

One way of improving the input for classification with incomplete data is to improve imputation methods which are used to estimate missing values. A problem with imputation approaches is that using imputation methods take time to estimate missing values. This may not be significant in the training process, but it is not feasible in many classification tasks to spend so much time in the application process to estimate missing values for an incomplete instance, where the instances to be classified are typically presented one by one, and the application process must be applied to each instance separately. This is especially true for powerful imputation methods such as Multiple Imputation by Chained Equations (MICE) [175], which are computationally very intensive when estimating missing values for a single incomplete instance because they must rebuild the whole imputation structure from all the training instances plus the new instance [46]. Though recent research has demonstrated the increased accuracy obtained by these advanced imputation methods, the high cost of these methods in the application process has seldom been addressed. This thesis explores the question of how to reduce the computation time in the application process without sacrificing the accuracy achieved by MICE and other advanced imputation methods.

When the input space contains numerous redundant/irrelevant features, many classifiers such as decision tress cannot achieve adequate accuracy. Feature selection that chooses a tailored feature subset from original features is a well known solution to the problem. The purpose of fea-

ture selection is to eliminate redundant/irrelevant features and only keep important features, while retaining or improving accuracy of the classification tasks. In feature selection, there are two main approaches to evaluating feature subsets: the filter approach and the wrapper approach. The filter approach uses measures such as information gain to evaluate the quality of feature subsets. The wrapper method builds a classifier to evaluate the quality of a feature subset. Feature selection has been widely used for improving classification with complete datasets [23, 183]. However, there has not been much work on feature selection with incomplete data. Filter approaches based on mutual information have been expanded to evaluate feature subsets when datasets contain missing values, and the experimental results show that it can help improve regression and classification tasks when faced with missing values [9, 36, 128]. However, a wrapper-based feature selection for classification with incomplete data has not been investigated. This thesis explores the use of a wrapper approach to feature selection on incomplete data.

Another way of improving the input for classification with incomplete data is to construct new robust features that transform incomplete data to complete data. A constructed feature is a function which transforms original values to new values, and the process of making constructed features is called feature construction. Constructed features are typically mathematical expressions of the original features. A *robust* constructed feature is one where the value of the feature can be calculated even when some of original features are missing. The original purpose of GP is to evolve computer programs from the input features and set of operators. Therefore, GP is an ideal choice for making constructed features. Recently, using GP for feature construction has interested many researchers [42]. GP-based feature construction can improve the classification performance of various classifiers such as decision trees [116], neural networks, and support vector machines [63]. However, GP-based feature construction has only been applied to complete data. This thesis explores the use of GP to construct

robust features that will work with incomplete data.

The second approach is to create robust classifiers that can work directly on incomplete data. One way of doing this is with ensemble learning. An ensemble classifier is a classifier consisting of a set of classifiers, and it has been proven to improve classification accuracy [35, 120, 124]. Robust ensemble classifiers have been conducted by building multiple classifiers working on different features in the training process and then selecting applicable classifiers to classify each incomplete instance without requiring any imputation method [24, 125, 185]. However, existing ensemble methods for classification with incomplete data often do not work well on datasets with numerous missing values [24, 185]. Moreover, they usually have to build a large number of classifiers, which then require a lot of time to find applicable classifiers for each incomplete instance in the application process, especially when incomplete datasets contain a high proportion of missing values [24, 125]. Therefore, how to construct a compact set of classifiers able to work well even on datasets with numerous missing values should be investigated.

Another way to create robust classifiers is to build individual classifier able to directly work on incomplete data such as C4.5 [129]. GP has been successfully applied to induce high quality classifiers [42]. The basic idea of the application of GP for inducing classifiers is that each individual is designed to represent a classifier or a part of a classifier, a fitness function is defined to measure its quality, and GP acts as a search technique to find a high quality final classifier. GP has been used to evolve decision trees, rule-based classifiers and discriminant functions. However, GP cannot directly work with data with missing values; therefore, to use GP for evolving classifiers for incomplete data, imputation methods are required to transform incomplete data into complete data before using GP. In order to evolve good classifiers, GP has to be combined with sophisticated imputation methods such as MICE [175]. However, sophisticated imputations such as MICE are often suitable for batch imputation, but computation-

ally expensive for imputing missing values in a single instance in the unseen set for classification. This thesis explores the ability of GP to directly evolve classifiers able to deal with incomplete data without imputation.

1.3 Goals

The overall goal of the thesis is to improve effectiveness and efficiency of classification with incomplete data by using evolutionary machine learning techniques. In order to achieve this goal, a set of research objectives have been established to guide this search, which can be seen as follows:

1. *Develop new approaches to improving imputation for classification with incomplete data by using clustering and feature selection*.

   In the application process of classification with incomplete data, the computation time to estimate missing values strongly depends on how many training instances are used by the imputation method. Clustering is the process of categorising data into clusters such that the instances in a cluster are similar to one another and different from the instances in other clusters. By selecting representative instances from the clusters, it is possible to reduce the original data to a smaller but representative subset, so clustering can be used to reduce the number of training instances needed for the imputation, which in turn can reduce the computation time. In the application process of classification with incomplete data, the computation time of the imputation process also strongly depends on the number of features in the training data. Feature selection can remove redundant and irrelevant features of the training data which not only may improve accuracy, but also can reduce the computation of the imputation. The combination of *imputation*, *clustering* and *feature selection* should improve effectiveness and efficiency of imputation for classification with missing data.

2. *Develop a new wrapper-based feature selection method to improve classification with incomplete data.*

   In order to build a wrapper-based feature selection for incomplete data, this research plans to use classifiers able to classify incomplete data such as C4.5 [129] and CART [105] to evaluate the feature subsets. EC techniques such as GA [5] and PSO [84] are combined with these evaluation measures to search for sufficient feature subsets. The proposed method should enhance classification accuracy and reduce the complexity of the learnt classifier.

3. *Develop a new feature construction method to directly construct multiple complete features for classification with incomplete data.*

   GP has been successfully applied to construct new features for classification with complete data [42]. To construct features for incomplete data, existing GP-based feature construction methods need be extended to be able to work with incomplete data. GP with interval arithmetic has been shown to improve symbolic regression [83]. Interval functions can work with incomplete data because each missing value may be considered as interval between a minimum value and a maximum value. This research uses to use interval functions as function sets of GP. These function sets are combined with GP techniques for feature construction to construct new complete features for incomplete data. The proposed method should improve classification accuracy and reduce the complexity of classifiers.

4. *Develop an effective and efficient ensemble approach to classification with incomplete data.*

   This objective aims to construct a set of classifiers which can effectively and efficiently classify new incomplete instances without requiring imputation. To achieve the goal, the proposed method integrates three techniques: imputation, feature selection and ensemble learning. Imputation is used to transform incomplete training data

to complete training data which is then further enhanced by feature selection. After that, the proposed method builds a set of specialised classifiers which can classify new incomplete instances without the need of imputation. The proposed method should be more accurate and faster than existing ensemble methods for classification with incomplete data. Moreover, the proposed method is also expected to be more effective and more efficient than other common approaches using imputation for classification with incomplete data.

5. *Develop a new approach to using GP for directly evolving classifiers for incomplete data.*

   The objective aims to use GP for evolving classifiers that can classify incomplete data without using imputation methods. GP has been used to extract the high quality classifiers for complete data; for example, GP has been used to extract decision trees, rule-based classifiers and discriminant functions. To evolve classifiers for incomplete data, GP techniques for constructing classifiers for complete data need be expanded to deal with incomplete data. The proposed method uses interval functions as function sets of GP to directly evolve classifiers for incomplete data. The proposed method should be more accurate and faster than the combination of GP and imputation for evolving classifiers.

1.4 Major Contributions

This thesis makes the following major contributions.

1. The thesis presents three new approaches that improve performance of imputation methods for classification with incomplete data. The first approach uses clustering to reduce the number of instances used for imputation in the application process. The second approach uses feature selection to remove redundant and irrelevant features of the

imputed training data which is then used to build a classifier and to estimate missing values in the application process. The third approach uses both clustering and feature selection. Experimental results show that the combination of imputation and clustering can significantly reduce the computation time of imputation in the application process, and can achieve comparable accuracy to using only imputation. The combination of imputation and feature selection can also significantly reduce the computation time, and can achieve better accuracy than using only imputation. Moreover, the combination of imputation, feature selection and clustering can not only further speed up imputation, but can also improve classification accuracy, simultaneously.

Part of this contribution has been submitted to:

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, Lam Thu Bui: "Improving Performance of Classification on Incomplete Data Using Feature Selection and Clustering". Applied Soft Computing 2018 (passed the second-round review).

2. The thesis presents a wrapper-based feature selection method for classification with incomplete data. The thesis also presents the integration of the wrapper-based feature selection method into ensemble methods to improve classification accuracy and reduce the complexity of the ensemble methods. Classifiers such as C4.5 able to work with incomplete data is used to evaluate feature subsets. Evolutionary techniques such as PSO are used to search for feature subsets. The experimental results show that the proposed wrapper-based feature selection method for incomplete data is able to enhance the classification accuracy of C4.5, significantly reduce the number of original features and significantly reduce the complexity of the learned classifier. Results also show that the integration of the proposed wrapper-based feature selection method into ensemble meth-

ods help improve the classification accuracy and reduce the complexity of the ensemble methods.

Parts of this contribution have been published in:

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue: "A Wrapper Feature Selection Approach to Classification with Missing Data". Proceedings of the 19th European Conference on the Applications of Evolutionary Computation (EvoApplication 2016) (1) 2016: 685-700.

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue: "Improving performance for classification with incomplete data using wrapper-based feature selection". Evolutionary Intelligence Journal 9(3): 81-94 (2016)

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue: "Bagging and Feature Selection for Classification with Incomplete Data". Proceedings of the 20th European Conference on the Applications of Evolutionary Computation (EvoApplication 2017): 471-486.

3. The thesis presents a GP-based multiple feature construction method for classification with incomplete data. The proposed method uses interval functions as the GP function set to tackle missing values by replacing each missing feature value by an interval. Experimental results show that the proposed method can achieve better accuracy than using original features or combining feature construction with simple imputation methods. The accuracy of the proposed method is comparable with combining feature construction with expensive imputation methods. The proposed method can also reduce the complexity of learnt classifiers better than all the other methods.

   Parts of this contribution have been published in:

   Cao Truong Tran, Peter Andreae, Mengjie Zhang: "Impact of imputation of missing values on Genetic programming based multiple feature construction for classification". Proceedings of the IEEE

Congress on Evolutionary Computation 2015 (CEC 2015): 2398-2405.

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue: "Directly Constructing Multiple Features for Classification with Missing Data using Genetic Programming with Interval Functions". Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO 2016): 69-70.

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue: "Genetic programming based feature construction for classification with incomplete data". Proceedings of the Genetic and Evolutionary Computation Conference 2017 (GECCO 2017): 1033-1040.

4. The thesis presents an effective and efficient approach for classification with incomplete data by integrating imputation, feature selection and ensemble learning. The proposed method uses imputation only in the training process to transform incomplete training data into complete training data that is then further improved using feature selection to remove redundant and irrelevant features. Then the proposed method constructs an ensemble of classifiers which can classify new incomplete instances without the need for imputation. Experiments compared the classification accuracy and the computation time of the proposed method with other benchmark methods for classification with incomplete data. Results show that the proposed method achieves better classification accuracy in most cases, and is much faster than the other methods in almost all cases.

   Parts of this contribution have been published in:

   Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue: "Multiple imputation and genetic programming for classification with incomplete data". Proceedings of the Genetic and Evolutionary Computation Conference 2017 (GECCO 2017): 521-528.

   Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, Lam Thu Bui: "An Ensemble of Rule-based Classifiers for Incomplete Data".

Proceedings of the 20th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES 2016): 7-12.

Part of this contribution has been submitted to:

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, Lam Thu Bui: "An Effective and Efficient Approach to Classification with Incomplete Data". Knowledge Based Systems 2018 (passed the first-round review).

5. The thesis presents an effective and efficient GP-based method to directly construct classifiers for incomplete data. To achieve this goal, GP with interval functions is used to construct a classifier for each incomplete dataset. Furthermore, GP with interval functions is also combined with ensemble learning to construct a set of classifiers for each incomplete dataset. Experimental results show that GP with interval functions is able to construct a single classifier that is more accurate than a single classifier constructed by combining imputation and traditional GP. Moreover, the combination of GP using interval functions and ensemble learning is able to construct a set of classifiers which is more accurate than a set of classifiers which is constructed by the combination of imputation, traditional GP and ensemble learning.

   Part of this contribution has been published in:

   Cao Truong Tran, Mengjie Zhang, Peter Andreae: "Directly evolving classifiers for missing data using genetic programming". Proceedings of the IEEE Congress on Evolutionary Computation 2016 (CEC 2016): 5278-5285.

1.5 Organisation of the Thesis

The remainder of thesis is organised as follows. Chapter 2 carries out a review of related work. Chapters 3-7 present the main contributions of the

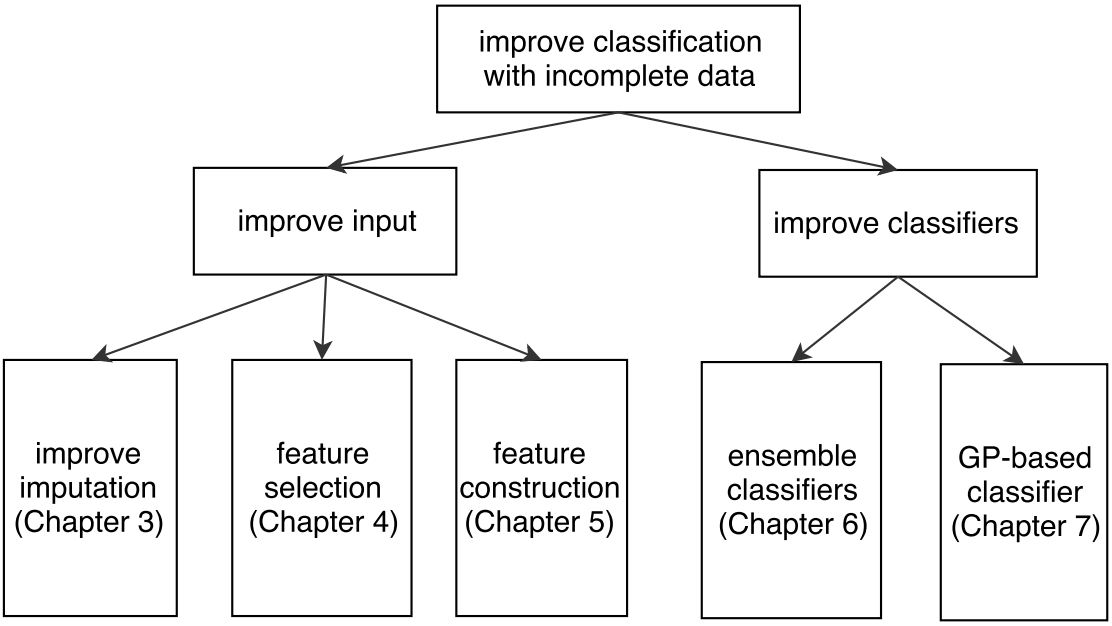


Figure 1.1: The overall structure of the contributions.

thesis as shown in Fig.1.1. Each chapter addresses one of goals described in 1.3 . Chapter 8 concludes the thesis.

Chapter 2 presents basic concepts and essential background of classification, missing data, imputation, evolutionary computation, feature selection and feature construction. It reviews typical related work in two main approaches to classification with missing data: using imputation to estimate missing values and constructing classifiers able to directly work on missing data. It also visits advances in evolutionary computation, feature selection and feature construction for classification. It then discusses open questions and current challenges that form the motivations of the thesis.

Chapter 3 proposes new approaches to improving imputation for classification with incomplete data by using clustering and feature selection. Clustering is used to reduce the number of instances used by the imputation in the application process. Feature section is used to remove redundant and irrelevant features of training data which greatly reduces the cost

of imputation in the application process. A set of experiments is conducted on classification problems with missing values of varying difficulty. The results are then presented and analysed.

Chapter 4 develops a wrapper-based feature selection method to improve the input space for classification with incomplete data. The proposed wrapper-based feature selection method is also investigated to improve ensemble learning methods. The proposed method uses evolutionary techniques such as PSO to search for feature subsets, and it uses a classifier which is able to directly work on incomplete data such as C4.5 to evaluate feature subsets. The proposed algorithm is then evaluated by measuring the classification accuracy and the complexity of learnt classifiers using selected subsets of features.

Chapter 5 proposes a GP-based multiple feature construction for classification with incomplete data. GP uses a set of interval functions as the function set to evolve a set of constructed features. After that, the constructed features are used to transform incomplete data into complete data. The performance of the proposed method is compared with a combination of feature construction and imputation and compared with classification algorithms that able to directly work on incomplete data.

Chapter 6 proposes an effective and efficient ensemble approach to classification with incomplete data by integrating imputation, feature selection and ensemble learning. Imputation is used to transform training data with missing values to complete training data which is then further enhanced by feature selection. Subsequently, the proposed method constructs a set of classifiers which is able to classify new incomplete instances without the need for imputation. The proposed method is compared with other common methods on real-world incomplete datasets using popular classification algorithms

Chapter 7 proposes an effective and efficient GP-based method to directly evolve classifiers for incomplete data. To achieve this goal, GP with interval functions is used to construct a classifier for each incomplete data.

Furthermore, GP with interval functions is also combined with ensemble learning to construct a set of classifiers for each incomplete data. The proposed method is then evaluated and compared with other common methods for classification with incomplete data.

Chapter 8 summaries the work and draws overall conclusions of the thesis. It also identifies key research points and the contributions of the thesis. It then suggests some possible future research directions.

1.6 Benchmark Datasets

Throughout the thesis, the proposed methods are evaluated and compared with other methods on a number of benchmark datasets of varying difficulty.

The datasets are chosen from the the UCI machine learning repository [8]. Table 1.1 shows the main characteristics of the chosen datasets: the number of instances, the number of features (Real/Integer/Nominal), the number of classes, the percentage of incomplete instances which contain at least one missing value, and the abbreviation of dataset.

The first fifteen datasets suffer from missing values in a "natural"way. These benchmark datasets are carefully chosen to cover a wide-ranging collection of problem domains. These tasks have various percentages of incomplete instances (incomplete instances range from 1.98% in the *Hec* dataset to 100% in the *Hed* dataset). These problems range from a small number of instances (*Hep* only has 155 instances) to a large number of instances (*Mar* has 8993 instances). These datasets also range from low to high dimensionality (*Mam* only has 5 features while *Arr* has 279 features). These problems encompass binary and multiple-class classification tasks. These datasets reflect incomplete problems of varying difficulty, size, dimensionality and type of features. All the real-world incomplete datasets are used in Chapter 3, Chapter 4, and Chapter 6.

Chapters 5 and 7 use GP to construct features and evolve classifiers, re-

spectively. A common property of datasets used in the two chapters is that they only have numerical features. This is because both constructed features and evolved classifiers are mathematical functions of the original features. Therefore, only seven real-world incomplete datasets (*Bre*, *Cle*, *Ban*, *Hep*, *Mam*, *Mar* and *Ozo*), which contain only numerical features, are used in the two chapters. To further evaluate the proposed methods in Chapters 5 and 7, "artificial"missing values are introduced into the last five complete datasets to generate incomplete datasets which are then used to test the proposed methods in the two chapters.

Table 1.1: Datasets used in the thesis

Name	**#Inst**	**#Features (R/I/N)**	**#Classes**	**Incomplete inst(%)**	**Abbrev**
Automobile	205	25(15/0/10)	6	26.83	Aut
Breast Cancer Wisconsin	699	9(0/9/0)	2	2.29	Bre
Cardiac Arrhythmia	452	279(206/0/73)	16	85.11	Arr
Chronic Kidney Disease	400	24(11/0/13)	2	60.5	Chr
Cleveland Heart Disease	303	13(13/0/0)	5	1.98	Cle
Credit Approval	690	15(3/3/9)	2	5.36	Cre
Cylinder Bands	539	19(13/6/0)	2	32.28	Ban
Hepatitis	155	19(2/17/0)	2	48.39	Hep
Horse-colic	368	23(7/1/15)	2	98.1	Hor
Housevotes	435	16(0/0/16)	2	46.67	Hou
Hungarian Heart Disease	294	13(6/0/7)	2	100	Hea
Mammographic	961	5(0/5/0)	2	13.63	Mam
Marketing	8993	13(0/13/0)	9	23.54	Mar
Ozone	2536	73(73/0/0)	2	27.12	Ozo
Primary Tumor	339	17(0/0/17)	22	61.01	Tum
Balance Scale	625	4(0/4/0)	3	0	Bal
Diabetes	768	8(8/0/0)	2	0	Dia
Iris Plants	150	4(4/0/0)	3	0	Iri
Liver Disorders	345	6(1/5/0)	2	0	Liv
Statlog Heart	270	13(13/0/0)	2	0	Sta

Chapter 2

Literature Survey

This chapter provides a review of the literature that forms the background and supports the motivations of the thesis. The chapter presents essential background on classification, ensemble learning, incomplete data, imputation, evolutionary computation (particularly GP, GP for feature construction and GP for evolving classifiers), feature selection and clustering. It reviews important related work in classification with incomplete data.

2.1 Classification and Incomplete Data

2.1.1 Classification

Classification is one of the main tasks in machine learning and data mining. It refers to the process of assigning a given instance, described by a vector of feature values, to one of the given classes. This field has been successfully applied to many scientific areas such as computer science, statistics, engineering, medicine and biology. These applications include biometrics, medical diagnosis, industrial automation and financial index prediction [38, 67, 177].

A learnt classifier is needed for classification. The classifier is learnt by a classification algorithm which is a supervised learning algorithm. The

learning algorithm uses a set of instances to learn a classifier that is expected to correctly predict the class label of unseen instances. Many different classification algorithms have been proposed in machine learning. The most popular classification algorithms are decision trees, k-nearest neighbour (kNN), Bayesian classification algorithms, support vector machines (SVMs) and artificial neural networks (ANNs) [38, 67, 177].

Decision trees have the advantage that they are easy to interpret. A major disadvantage of the standard decision trees is their weakness in separating non-rectangular areas in the input space [129]. kNN has the advantage of not requiring any assumptions on the underlying data distribution. Although kNN is a simple learning algorithm, it often works well in practice. However, for a large training set, kNN requires large memory and is very time-consuming to make a decision [179]. Bayesian classification algorithms is a probabilistic approaches to classification. Naive Bayes (NB) classification is the simplest and most common Bayesian classification algorithm. An advantage of NB is that it only requires a small amount of training data to evaluate parameters for classification. However, the assumption of features being conditionally independent from each other is not true in many real-world problems, where interdependency between input features is common [179]. SVMs generally have good performance, but are hard to interpret, particularly with numeric data. ANNs have been widely used to solve a variety of hard tasks such as computer vision and speech recognition; however, like SVMs, they are hard to interpret [67].

An important class of techniques for classification is ensemble learning which uses a set of classifiers instead of a single classifier. Ensemble techniques first build a set of classifiers, and then a new instance is classified by conducting a vote with decisions of the individual classifiers. Ensemble learning has proved capable of achieving better classification accuracy than any single classifier [35, 120, 124, 193].

An ensemble of classifiers is good if the individual classifiers in the ensemble are accurate and diverse. Bagging and Boosting are two popu-

lar approaches to building accurate ensembles. Both Bagging and Boosting use "resampling"techniques to manipulate the training data. Bagging manipulates the original training dataset of N instances by randomly drawing instances with replacement. Therefore, in the resulting training dataset, some of the original instances may appear multiple times while others might disappear. Bagging is often effective on "unstable" learning algorithms such as neural networks and decision trees where small changes in the training dataset can lead to major changes in predictions. Experimental results show that a Bagging-based ensemble almost always performs better than a single classifier [35, 120].

Boosting manipulates the training dataset for each individual classifier by using the performance of the previous classifier(s). In Boosting, instances which are incorrectly classified by previous classifiers are selected more often than instances which are correctly classified. Therefore, Boosting tries to build new classifiers that are better at classifying instances for which the current ensemble's performance is poor. Empirical results show that with little or no classification noise, a Boosting ensemble also almost always performs better than a single classifier, and it is often more accurate than a Bagging ensemble. However, in situations with substantial classification noise, a Boosting ensembles is often less accurate than a single classifier because Boosting often overfits noisy datasets [35, 120].

2.1.2 Incomplete Data

An incomplete dataset is a dataset which does not have values in some fields; in other words, it contains missing values. For example, the *Mammographic Mass* dataset [40], which is used to predict the severity (benign or malignant) of a mammographic mass lesion, is an incomplete dataset. Table 2.1 presents some instances of the *Mammographic* dataset. In Table 2.1, "?" refers to missing values. Table 2.1 shows that all five features of the *Mammographic* dataset contains missing values.

Table 2.1: An example of incomplete data.

BI-RADS	Age	Shape	Margin	Density	Class
5	67	3	5	3	malignant
4	70	?	?	3	benign
?	52	4	4	3	benign
4	?	4	5	3	malignant
4	31	1	1	?	benign

Many real world datasets suffer from the unavoidable problem of missing values [4, 49]. For example, 45% of datasets in the UCI machine learning repository—one of the most popular benchmark datasets for machine learning and data mining—contain missing values [57]. Behind this serious problem, there are various reasons [4, 100]. For example, respondents in a social survey may refuse to answer some questions [107, 138, 169]; in an industrial experiment, some results may be missing due to mechanical failures while collecting data [93, 176]; medical databases often suffer from missing data where almost every patient's record lacks some values because not all possible tests can be run on every patient [56, 75, 127, 151, 178]. Gene expression datasets usually contain a large number of missing values because of hybridization failures, inadequate resolution and image corruption and noise [31, 101, 165].

Missing values lead to a number of serious problems for data analysts. Firstly, although some methods of data analysis can work with incomplete data, the majority of existing methods of data analysis require complete data. As a result, these data analysis methods cannot work directly with original data containing missing values, unless they are combined with other techniques, which cost time, lead to complications in handing and analysing the data, and may cause biased results [100, 140].

In most cases, in order to handle missing data appropriately, information of how data became missing is essential. In [100], Little and Rubin

define three types of missing data mechanisms:

- *Missing completely at random (MCAR)*
  Missing data is missing completely at random (MCAR) if the probability that a feature value is missing is independent of the value that is missing, the value of that feature in every other instance, and the value of every other feature in every instance.

- *Missing at random (MAR)*
  Missing data is missing at random (MAR) if the probability that a feature value is missing is independent of the value that is missing, and of every other value that is missing (in other features or other instances), but may depend on values of features that are complete.

- *Not missing at random (NMAR)*
  Missing data is not missing at random (NMAR) if the probability that a feature value is missing is dependent on the missing feature values. Therefore, the missing value in the NMAR case cannot be predicted only from the available values in the database.

2.1.3 Approaches to Classification with Incomplete Data

Missing values cause serious problems for classification. One of the most serious problems as for other data analysis techniques is that the majority of classification algorithms do not work on incomplete datasets [25, 45, 80]. For example, neural networks cannot directly work with incomplete data. Even if the missing data can be handled, another problem is that missing values often result in big classification error [57, 46, 45].

There are three major approaches to classification with incomplete data: the deletion approach, the imputation approach, and directly classification with incomplete data [57].

- Deletion approach: this approach simply deletes all instances containing missing values. The benefit of this approach is that it pro-

vides complete data for classification. However, the deletion approach is only feasible for datasets with few missing values because this approach cannot provide enough information for training a classifier when a dataset has numerous missing values. Another drawback of this approach is that incomplete instances with missing values cannot be classified in the application process. Therefore, the deletion approach can be only used when a dataset contains a small number of missing values [1, 99], and few unseen instances are incomplete. Moreover, this method should only be used when missing data is introduced by the MCAR mode, otherwise the results will be biased [45, 100].

- Imputation approach: this approach uses imputation methods to transform incomplete data into complete data before building a classifier in the training process or classifying a new incomplete instance in the application process. This approach can provide complete data which then can be used by any classification algorithm. It also can deal with incomplete datasets with a large number of missing values. Therefore, imputation is the most popular approach to classification with incomplete data [45, 109, 132]. MAR is usually assumed by most of the existing imputation methods [45, 100].

- Directly classification with incomplete data: this approach builds a model in the training process which can directly classify incomplete instances in the application process without requiring any imputation method. For example, C4.5 can directly classify incomplete datasets by using a probabilistic approach [129]. Another example is to build a set of classifiers, and then choose only applicable classifiers to classify an incomplete instance [24, 125]. This approach can save time for estimating missing values, and it does not require any assumption about missing data [24]. However, existing methods are often inaccurate when faced with a large number of missing values.

2.2 Imputation

The goal of imputation is to replace missing values with plausible values. One of the most common ways to categorise imputation methods is to divide imputation methods into single imputation methods and multiple imputation methods. Single imputation methods estimate a single value for each missing value while multiple imputation methods estimate a set of plausible values for each missing value. Multiple imputation is generally more accurate, but more expensive than single imputation [7, 46, 100].

2.2.1 Single Imputation

Single imputation methods can be further divided into data-driven (or statistical), model-based, machine learning (ML)-based and evolutionary computation (EC)-based methods [46, 57, 59]. Data-driven imputation methods use only available data to estimate missing data. Model-based imputation methods make an assumption that data follows a specific model which is then used to estimate missing values. ML-based imputation methods use the available data and ML algorithms such as decision trees, kNN, Bayesian-based algorithms, and ANNs to estimate missing values. EC-based imputation methods use the available data and EC algorithms such as genetic algorithms (GAs) and genetic programming (GP) to estimate missing values .

2.2.1.1 Data-driven Imputation

Data-driven imputation methods which use available data to impute missing values include simple imputation methods such as mean/mode imputation and hot-deck imputation [6, 57].

Mean/mode imputation replaces missing values in each feature with the average of the complete values in the same feature. If a feature is categorical, the mode, which is the most frequent value of the feature, is used

instead of mean. This method maintains the mean of each feature, but it under-represents the variability in the data because all missing values in same feature have the same value [57].

In hot-deck imputation, for each instance containing missing values, the most similar instance is found and then missing values are filled with complete values from the most similar instance. An advantage of this method is that it replaces missing values by real values from the data. However, this method ignores all global properties of the data, because it uses information of only the most similar instance [6].

2.2.1.2 Model-based Imputation

In model-based imputation methods, we need to make assumptions about the joint distribution of all the features in the model. These methods include regression-based imputation [57, 133, 140] and likelihood-based imputation [44, 61, 100].

Regression-based imputation methods use regression models to estimate missing values of a given instance based on complete values of the instance. Linear regression or non-linear regression methods are suitable for continuous features while log-linear regression methods are suitable for discrete features. This approach can preserve the interaction between features, but when there exist numerous missing values, this approach requires building a large number of regression equations, each for a set of complete features, which then causes high computational cost [57, 140].

One of the most popular likelihood-based imputation methods is expectation maximization (EM)-based imputation which applies the EM algorithm to estimate a maximum likelihood variance-covariance matrix and vector of means which are then used to impute missing values. This method is an iterative procedure that includes an E-step and a M-step at each iteration. In the E-step, the means, variances and covariances are estimated from complete values and the current best guess of missing values. In the M-step of the same iteration, maximum likelihood procedures are

used to estimate new regression equations for each attribute predicted by all others which are then used to update the best guess for missing values during the E-step of the new iteration [44, 61, 100].

2.2.1.3 Machine Learning-based Imputation

Machine learning-based imputation methods use machine learning algorithms to build data models that are then used to estimate missing values. This approach includes decision tree-based imputation, kNN-based imputation, Bayesian-based imputation, and ANNs-based imputation.

Decision trees are one of the most common approaches for classification/regression. Decision trees are also used for estimating missing values. [174] uses all observations to build a binary classification tree, which is then used to estimate missing values. [29] proposes an incremental imputation algorithm which uses binary decision trees to build incremental procedures to estimate missing values. In [130], a decision tree is combined with the EM algorithm to build an efficient imputation technique. Experimental results show that the proposed method performs significantly better than EM-based imputation. In [131], decision trees and decision forests are used to identify horizontal segments of a dataset, and then the information of the similarity and correlation of the segments is used to estimate missing values. In [32], decision trees are used to exploit the within-instance and between-instance correlations to estimate missing values for traffic accident data.

kNN-based imputation method is a modified version of hot deck imputation. For each instance containing missing values, it finds the k most similar instances, and then fills missing values of the instance with the average of the values in the k most similar instances [12]. There are some improved versions of kNN-based imputation such as weighted kNN-based imputation in [58, 165, 170]. kNN-based imputation is typically better than mean imputation and hot deck imputation [12]. However, this method often suffers from a high computational cost because of having to search

through all instances to find the k most similar instances of each instance containing missing values [57].

There are several Bayesian networks-based imputation methods. In [34], Bayesian networks are used to improve the the consistency of imputed values by preserving statistical consistency and logical consistency. [73] proposes two imputation methods-based on Bayesian networks for classification. The proposed methods are compared with classical imputation methods such as EM-based imputation, and results show that the Bayesian-based imputations can achieve comparable accuracy to classical imputation methods. In [134], Bayesian networks are learnt from incomplete data using a structural EM procedure, and the learnt networks are then used to estimate missing values. Experimental results show that the proposed method usually performs better than several existing imputation methods on both real and simulated datasets.

Different kinds of artificial neural network are used for imputation. Multi layer perceptrons (MLP) are used to estimate missing values in [10, 147, 148] by building one MLP model per missing features combination. These MLP-based imputations are good at estimating missing values, but they require a large number of MLP models when data containing numerous missing values. The self-organizing map (SOM), which is a type of artificial neural network, is also used for imputing missing values [47, 50, 103]. Empirical results show that SOM-based imputation methods can work better than standard MLP-based imputation [47]. In [112], an auto-associative neural network is also used to estimate missing values.

2.2.1.4 Evolutionary Computation-based Imputation

Evolutionary computation is a subfield of artificial intelligence which is inspired by biological evolution [48]. Recently, evolutionary computation algorithms such as genetic algorithms(GAs) and genetic programming (GP) have been used for estimating missing values.

A genetic algorithm is an evolutionary algorithm which is inspired by

the process of natural selection [5]. There are several GA-based imputation methods [55, 104, 126]. In [55], a genetic algorithm is used to estimate missing values in multivariate data. In [126], a genetic algorithm is used to impute missing values in discrete features by searching for the most suitable value for each missing value. In [104], a multi-objective genetic algorithm is used to build an imputation method which can deal with both numeric and nominal features. Experimental results show that the proposed method can perform better than common imputation methods and can be flexible with different application domains.

Genetic programming (GP) is good at doing regression, so it has been used to predict missing values [11, 157]. In [11], a strong type GP is used for imputing missing values. [157] proposes a GP-based imputation (GPI) method for classification with incomplete data that uses GP as a regression method to impute missing values. The experiments compare GPI with five other popular and advanced regression-based imputation methods on two measures: classification accuracy and computation time. The results showed that, in most cases, GPI achieves classification accuracy at least as good as the other imputation methods, and sometimes significantly better. Moreover, using GPI to impute missing values for every single incomplete instance is dramatically faster than the other imputation methods.

2.2.2 Multiple Imputation

Multiple imputation estimates a set of values for each missing value [61, 100]. Multiple imputation often assumes MAR, and it can produce asymptotically unbiased estimates [175]. Moreover, multiple imputation is usually better than single imputation because it can better reflect the uncertainty of missing data than single imputation [46]. However, multiple imputation is generally more computationally expensive than single imputation since it takes time to estimate a set of values for each missing value [92, 100].

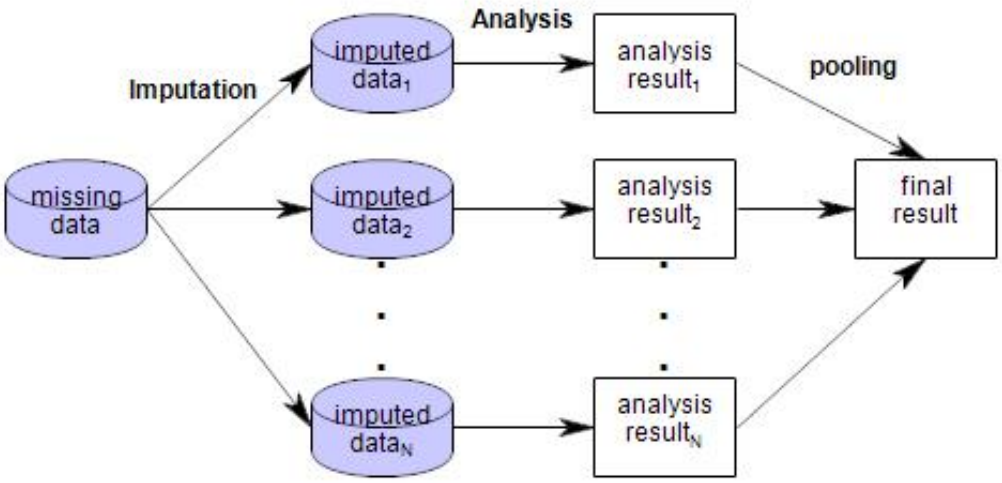


Figure 2.1: Main steps of using multiple imputation for data analysis.

Figure 2.1 illustrates three main steps of using multiple imputation for data analysis. Firstly, missing data is imputed N times (N$\geq$ 1) to generate N different imputed data sets (complete data sets) using an imputation model that incorporates random variation. Secondly, the imputed data sets are analysed separately by using standard procedures for handling complete data. The second step provides N analysis results. Finally, the N analysis results are combined into a final result [61, 100, 140].

Multiple imputation has become increasingly popular for several reasons. Firstly, although multiple imputation methods are computationally more expensive than single imputation methods, multiple imputation methods are usually better than single imputation methods [46, 150, 100]. Furthermore, many recent software developments have based on the multiple imputation framework [69].

Multiple imputation using chained equations (MICE) is one of the most flexible and powerful multiple imputation methods [22, 172, 175]. MICE uses regression methods to estimate missing values. Algorithm 1 shows

the main steps of MICE. Initially, each missing value in each feature is replaced by a random complete value in the same feature. Subsequently, each incomplete feature is regressed on the other features to compute a better estimate for the feature. The process is performed several times (n) for all incomplete features to provide a single imputed dataset. The whole procedure is repeated m times to provide m imputed datasets. Finally, the final imputed dataset is calculated by the average of the m imputed datasets [22, 175].

Algorithm 1: $MICE(D)$

Input:
D, an incomplete dataset
m, the number of imputed datasets
n, the number of cycles
Output:
An imputed dataset

```
for i ← 1 to m do
    D_i ← RandomImputation(D)
    for j ← 1 to n do
        foreach incomplete feature f in D do
            Regress f in D_i on other features in D_i
        end
    end
end
return Average(D_1,...,D_m);
```

2.2.3 Imputation for Classification with Incomplete Data

Figure 2.2 shows the main steps of using imputation for classification with incomplete data. In the training process, imputation is used to estimate missing values for the training data. Subsequently, the imputed training data is put into a classification algorithm to build a classifier. In the application process, when a new instance needs to be classified, it is directly classified by the classifier if it is complete. Otherwise, its missing values

are replaced by suitable values which are estimated by using the imputation method and the imputed training data. Finally, the imputed instance is classified by the built classifier.

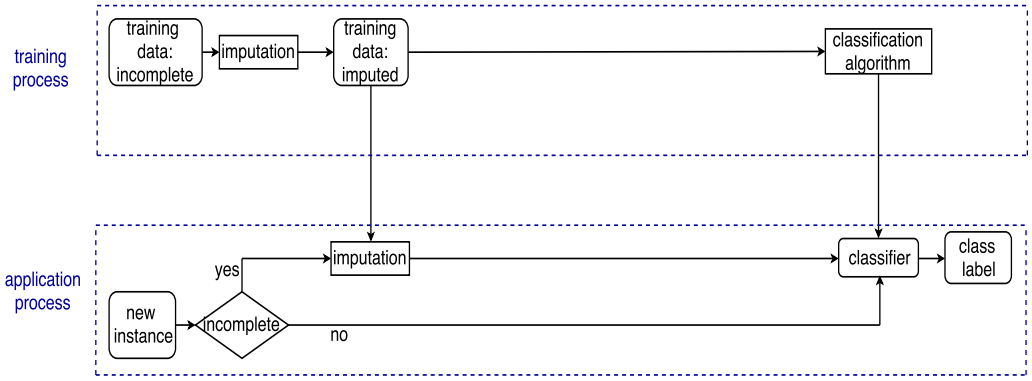


Figure 2.2: A common approach to using imputation for classification with incomplete data.

Single imputation methods have been widely used for classification with incomplete data. In [12, 13], kNN-based imputation is used to estimate missing values. Results show that using kNN-based imputation performs better than using C4.5 to directly classify incomplete data, and kNN-based imputation also performs better than the widely used mean imputation. [1] compares four common methods for tackling missing values including deletion case, mean imputation, median imputation and kNN-based imputation on twelve datasets. Results show that kNN-based imputation can outperform the other methods, particularly with datasets containing a high percentage of missing values. In [45], six common imputation methods are combined with six popular classifiers on fifteen discrete datasets. The study shows that, on average, imputation enhances classification accuracy compared to not using imputation. In [58], a mutual information-based feature-weighted distance is used to improve kNN-based imputation. Results show that mutual information-based imputation is more accurate than common kNN-based imputation. In [109], four-

teen single imputation methods are tested on three classification groups. Experimental results show that the imputation methods outperform those without imputation. Moreover, there is no imputation method that is the best for all classifiers. Furthermore, the use of a particular imputation method depends on the required classifier. [80] proposes a locally linear reconstruction (LLR)-based single imputation. The LLR-based imputation is compared with six common single imputation methods on 13 classification problems. Results show that all imputation methods improves classification accuracy, and the proposed method is more accurate than the other single imputation methods, especially when faced with a large number of missing values. In [39], a multi-layered artificial immune system is combined with a genetic algorithm to impute missing values for classification with incomplete insurance datasets. The hybrid imputation method is better than mean/mode imputation, and comparable or marginally better than hot deck imputation. [147] proposes a multilayer perceptron-based single imputation method for monotone patterns of missing values. Results show that the proposed method outperforms mean/mode imputation, regression and hot-deck imputations.

Multiple imputation has also been used to estimate missing values for classification with incomplete data. In [46], multiple imputation methods are compared with some other single imputation methods on sixteen discrete datasets. Results show that multiple imputation generally outperforms single imputation, but it is much more expensive than single imputation. In [102], two multiple imputation methods are compared with three single imputation methods in terms of the classification accuracy. The study shows that multiple imputation can perform better than single imputation, especially with datasets containing numerous missing values. In [147], multilayer perceptron is combined with k-nearest neighbours to build a multiple imputation for monotone patterns with missing values. Results show that the multiple imputation outperforms single imputation such as mean/mode, regression and hot deck imputation methods. [159]

proposes GPMI, a multiple imputation method that uses genetic programming as a regression method to estimate missing values. Experiments on eight datasets with six levels of missing values compare GPMI with seven other popular and advanced imputation methods on two measures: the prediction accuracy and the classification accuracy. The results show that, in most cases, GPMI not only achieves better prediction accuracy, but also better classification accuracy than the other imputation methods. [164] proposes a new method to use multiple imputation for classification. Each imputed dataset from multiple imputation is used to build a classifier. The set of classifiers is used to classify a new instance by majority voting in the application process. Empirical results show that the proposed method can obtain better classification accuracy compared to using multiple imputation in the traditional way (averaging the imputed datasets into a single dataset for training a single classifier). [157, 158, 161, 164, 163] further confirm that multiple imputation is more accurate, but much more expensive than single imputation for classification with incomplete data, especially for the application process.

In summary, using imputation, especially multiple imputation, on incomplete datasets can obtain better classification accuracy than not using imputation. However, imputation takes time to estimate missing values, especially in the application process and with multiple imputation. Moreover, existing imputation research for classification has focused mainly on improving classification accuracy, with little concern for the computation time. Therefore, research on how to improve both the effectiveness and efficiency of using imputation for classification in incomplete data should be further investigated.

2.3 Direcly Classifying with Missing Data

There are two main approaches to directly classifying incomplete data without requiring imputation. The first approach is to build a classifier

such as C4.5 which can directly work with incomplete data [129]. Another approach is to build a set of classifiers (one classifier for each missing pattern), and then select applicable classifiers from the set of classifiers to classify new instances.

2.3.1 Decision Trees

Some decision trees such as C4.5/CART can directly work with incomplete data. C4.5 computes the information gain of an incomplete feature by computing the gain on the complete values and discounting it by the ratio of complete instances to all instances. C4.5 utilises a probabilistic approach to addressing missing values in both the training set and test set. C4.5 makes assumption that instances with the missing values are distributed probabilistically according to the relative frequency of known values. In the training process, each feature value is assigned a weight: the weight is assigned one if a feature value is known; otherwise, the weight of any other values for that feature is the frequency of those values. In the testing process, when a test feature is chosen, the cases with known values are divided into branches corresponding to these values. The cases with missing values are passed down all available branches, but with weight that corresponds to the relative frequency of the value assigned to a branch and it decides the class label by using the most probable value [129].

The main benefit of using classification algorithms able to directly classify incomplete data is that the classification algorithms do not require any time for estimating missing values. However, when the classification algorithms work with incomplete data, they often generate more complex models and lead to large classification errors [139], especially when data contains a large number of missing values. Therefore, further approaches to improving the classification algorithms should be investigated.

2.3.2 Ensemble Methods

Ensemble learning is a learning method which constructs a set of base classifiers for a classification task in the training process. Subsequently, the predictions of base classifiers are combined to classify new instances in the application process. An ensemble of classifiers has been proven to be more accurate than any of base classifiers making up the ensemble [124].

Ensemble learning also has been applied to classification with incomplete data. One of the first work using ensembles for classification with incomplete data appears in [142], where four neural networks are built to address classification with a thyroid disease database consisting of two incomplete features (one classifier that ignores both the missing features, two classifiers that can be used if one of the features is present, and one classifier that requires both features to be present). Experimental results show that the ensemble is superior to an imputation method using neural networks to estimate missing values. The ensemble is also more accurate than an induction algorithm which builds a decision tree able to directly work with incomplete data. [79] builds an ensemble of one-class classifiers which are trained on each feature. After that, the ensemble can classify incomplete instances even when the instances have only one complete value. The systems described in [24] and [76] tackle incomplete data by learning a set of neural networks, where each neural network is trained on one complete sub dataset extracted from incomplete data. Given a new incomplete instance to classify, available learnt classifiers are combined to classify the instance without requiring imputation. Empirical results show that the ensemble of neural networks can achieve better accuracy than other two ensemble methods combined with mean and mode imputations. A similar approach is proposed in [184], where conditional entropy is used as the weighting parameter to reflect the quality of feature subsets which are used to build base classifiers. The previous proposal is extended in [185] by using the mutual information criterion to eliminate redundant feature

subsets. As a result, the extended system can not only outperform other methods, but can also reduce the computation time to classify incomplete instances. In [90] and [125], an ensemble of numerous classifiers is constructed, where each base classifier is trained on a sub dataset by randomly selecting a feature subset from the original features. Thanks to constructing a larger number of base classifiers, the system can cope with incomplete data containing a high proportion of missing values. [51] constructs a meta-ensemble of classifiers to handle incomplete and unbalanced data. The meta-ensemble is used to deal with missing values in cyber security domain. Results show that the proposed method is comparable to other ensemble-based methods with smaller number of classifiers.

Existing ensemble methods for classification with incomplete data can deal with missing values to some extent. However, the ensemble methods usually cannot obtain good accuracy when datasets contain a large number of missing values [24, 185, 142]. The underlying reason is that the complete sub datasets often only have a small number of instances when the original incomplete data includes a huge number of missing values. Therefore, base classifiers trained on the complete sub datasets are weak classifiers. To overcome the problem, numerous base classifiers have to be built [125], which requires a long time for exploring available classifiers to classify new incomplete instances. Therefore, how to build an effective and efficient ensemble for classification with datasets containing numerous missing values should be further investigated.

2.4 Genetic Programming

2.4.1 Genetic Programming Algorithm

Genetic Programming (GP) is one of the most popular evolutionary techniques inspired by biological evolution to search for a solution for a problem in the form of a computer program [87, 88]. In the 1990s, GP was

mainly applied to relatively simple problems because GP was quite computationally intensive. Nevertheless, thanks to the exponential growth in CPU power and the improvements in GP, GP has been used to solve many real-world problems. The applications of GP include electronic design, game playing, quantum computing and sorting [87, 88].

Algorithm 2: Genetic Programming

Randomly create an initial population of programs from the available primitives.

repeat

- Execute each program and estimate its fitness.
- Select one or two program(s) from the population with a probability based on fitness to attend genetic operations.
- Apply genetic operations to create new individual program(s) with specified probabilities.

until *stopping condition is met (e.g., an acceptable solution is found, or a maximum number of generations is reached)*

return the best-so-far individual;

Algorithm 2 shows pseudo-code of GP. The following steps are needed to apply GP to deal with a problem [123].

2.4.1.1 Representation of candidate solutions

Although, in a GP system, there are a number of ways proposed to represent candidate solutions , the most popular form is a tree-based representation. To generate a population of individuals, firstly, a GP practitioner has to choose a function set and a terminal set. The function set is often chosen from arithmetic operations (such as +, - , *, /), mathematical functions (such as sin, cos, exp, log), boolean operations (such as AND, OR, XOR, NOT) and conditional operations (such as IF-THEN-ELSE). The terminal set often includes a number of features and several constants. After

that, a GP individual is built up recursively from the selected functions and terminal sets [87, 123].

The selected function and terminal sets have to satisfy the *closure* and *sufficiency* properties. The *closure* property requires that each function must be able to tackle every possible input it receives. The *sufficiency* property requires that the selected functions and terminal sets must have adequate ability to represent the solution for the problem [87, 123].

2.4.1.2 Initialising a Population

The purpose of initialisation step is to generate a population of initial individuals that will be evolved in the later steps. In a tree-based GP system, the *grow* method, the *full* method and the *ramped half-and-half* method are the three main approaches to create an initial population.

To generate an initial individual in the *grow* initialisation, firstly, a function in the function set is randomly selected and is considered as the root node of the tree. Assuming that the arity of the selected function is n, then n nodes are randomly selected from the function and terminal sets as the children of the root node. If a function is chosen, the recursive process is applied to that function. If a terminal is selected, this branch of the tree is terminated. The maximal depth of tree is usually utilised to control the limitation of the size of the initial individual, .

In *full* initialisation, when building a tree, instead of choosing nodes from the function and terminal sets, only functions from the function set are selected until it reaches the maximal depth where only components from the terminal set are selected.

In the ramped half-and-half initialisation, a half of the population is generated by using the grow method and the remaining half is generated by the full method. This method often results in greater diversity of the GP population; therefore, it is the most popular initialisation technique [123].

2.4.1.3 Genetic Operators

Crossover: is the primary operator in GP. Crossover generates new children that are formed from parts taken from each parent; hence, crossover creates variation in the population. In standard crossover, firstly, two parents are selected by using a selection method, and then one subtree is randomly chosen in each parent. If these two subtrees are legal for crossover (depth of resulting children, syntactic closure properties, etc), the crossover is done by swapping the two chosen subtrees, and then the new offspring are added to the next generation [87].

Mutation: is the secondary operator in GP. Mutation is an asexual operator meaning that it operates on only one parent. In standard mutation, a mutation point is randomly chosen, and then the subtree rooted at the mutation point is removed and a newly generated subtree is added at that point [87].

2.4.1.4 Fitness Evaluation

Each individual in the population is assigned a numerical value called fitness that represents its ability to solve the problem. Fitness has to be accurate to reflect the problem, but also be efficient to compute. Therefore, it is important to make a good trade off between these two properties [88, 123].

Two popular fitness measures in GP are *raw* and *standardised* fitness. *Raw* fitness is the most simple form of fitness that reflects the ability of an individual to solve the problem. *Standardised* fitness is estimated using raw fitness so that the smaller (for minimizing problems) is the better [88, 123].

2.4.1.5 Selection Mechanism

Based on its fitness value, each individual in the population has a chance to be chosen to participate in the breeding of the new population. The three most popular form of selection operations are *tournament*, *fitness proportionate* and *ranked selection*.

In tournament selection, a randomly selected subset of individuals from the population are compared with each other in terms of fitness and the most fit is chosen to go to the mating pool. The positive aspect of tournament selection is that the selection pressure is able to adjust by changing the size of subsets. Furthermore, tournament selection does not require a comparison of the fitness between all individuals. Therefore, it may assist to reduce a large amount of processing time and make it easy to parallelise the algorithms [88, 123].

In fitness proportionate selection, based on fitness, each individual is assigned a probability to be chosen for the mating pool. Although proportionate selection has been often used in GP and GA, its behaviour strongly depends on the difference between fitnesses of the individuals [88, 123].

In ranking selection, based on their fitness, all individuals are sorted. After that, based on its order, the selection probability is assigned to each individual. Linear and exponential ranking are often used to index individuals. Although ranking selection technique assists to lessen the weakness of proportionate selection, in some cases, especially in exponential ranking, this method increases the difference between close fitness individuals; therefore, the better one can be chosen more frequently [88, 123].

2.4.2 GP for Feature Construction

Feature construction is the process of transforming the values of one or more features to a new set of values. The purpose of feature construction is to transform the original input space to a new input space, where classification algorithms can achieve better classification accuracy than using

the original input space. Feature construction has been proven to improve performance of classifiers, particularly those based on symbolic learning (e.g. decision trees and decision rules) that cannot achieve adequate predictive accuracy when faced with difficult real-world problems [116, 160].

A constructed feature is a function which transforms values of the original features to new values. The purpose of feature construction is to create a set of new features which can provide a new input space for classification that is better than the original input space. Constructed features are typically mathematical expressions of the original features. The original purpose of GP is to evolve computer programs that perform a user-defined task. Therefore, GP is an ideal choice for making constructed features. Using GP for feature contruction has been a new research trend in recent years [42, 116, 160].

In a GP-based feature construction approach, new constructed features are often represented by tree-like individuals, where internal nodes contain mathematical operators and functions, and leaf nodes contain original features or constants. GP is used as a search technique that depends on other classifiers (wrapper approaches) or some measures (filter approaches) to estimate fitness functions to guide the search for new constructed features. GP has been applied to both filter and wrapper approaches to feature construction [42, 116, 160].

2.4.2.1 Filter Approach

In [114], four fitness functions namely information gain, the gini index, a combination of information gain and the gini index and chi-square are used to choose the best constructed features. After that, four classification algorithms namely C5.0, CART, CHAID and a multilayer perceptron are used to build classifiers. Each individual in the population represents a new constructed feature and when the evolutionary process ends, the best individual is added to the original dataset to make an augmented dataset that is used to induce the classifiers. The results show that GP is effective at

building highly predictive non-linear features. With datasets having few classes, all of the classifiers generally achieve a better performance when the augmented dataset is used, irrespective of the fitness functions used by the GP to evaluate the feature. Furthermore, the size of the classification trees can be reduced significantly.

In [63], a GP-based feature construction is proposed to extract new features from the raw vibration data for the bearing fault classification problem. The Fisher criterion is used to build the fitness function. The empirical results show that GP is an efficient and powerful method for the automatic feature generation directly from the raw data. By using GP-based constructed features, the ANN and SVM achieve better classification results, compared to those using features extracted by classical methods. Moreover, the classification performance achieved from GP-based extracted features are very robust. Furthermore, GP is capable of decreasing the dimensionality to describe the problem.

In [64], a fitness measure based on modified Fisher linear discriminant analysis (MFLDA) is developed to help GP construct a useful feature which is optimized to minimize the within-class scatter and maximize the between-class scatter of pattern vectors. The experimental results on Wisconsin diagnostic breast cancer show that MFLDA performs better than other linear classical feature extraction methods. Moreover, the combination of GP-based feature construction using MFLDA and minimum distance classifier outperforms the other two GP-based feature construction methods using the same classifier and also outperforms using original features and MLP and SVM.

In [116], GP-based multiple feature construction (GPMFC) is proposed. GPMFC uses GP for evolving new features and utilizes purity of class intervals as a measure of fitness. The key difference between GPMFC and other existing filter approaches to GP-based feature construction is that it uses a decomposable fitness measure. Other approaches typically use a fitness function having a fixed formulation to estimate the usefulness of

a feature and consequently can build only a single constructed feature. In contrast, GPMFC uses a fitness function with a parameter for the class label; therefore, it can construct one feature per class label. The empirical results show that GPMFC, in most cases, enhances the classification of decision trees and rule-based classifiers. Moreover, it can also reduce the complexity of the learnt classifiers

2.4.2.2 Wrapper Approach

In [136], a GP feature constructor is wrapped around kNN algorithm. Each individual in the GP population is a forest composed by n trees (n is the number of original features), each tree t_i ($1 \leq i \leq n$) represents the i^{th} constructed feature. The results show that the GP approach provides an overall improvement in prediction accuracy opposed to the GA runs.

In [143], wrapped classification algorithm is a Generalised Linear Machine (GLIM). Each individual is a tree that uses arithmetic operations to combine the original features. Only the constructed features are used in classification and the original features are not used by the classifier. The empirical results show that GP can construct complex features that help to reduce classification error of difficult problems. In [144], the same authors extend their research by using three classifiers namely GLIM, kNN and the maximum likelihood classifier, and each individual is associated with one type of classifier. Therefore, GP can select the best constructed features, and also selects the most suitable type of classifier. In experiments performed on nine real-world data sets, the GP-based feature construction algorithm is capable of pre-processing the data to reduce the test set misclassification rate and reduce the dimensionality of the data.

In [91], the wrapped classification algorithm is C4.5, and the fitness measure is the accuracy of the decision tree. Each individual contains a certain number of trees, each tree represents a new constructed feature. The best individuals are kept in an elitist repository. Individuals preserved in the elitist subforest are selected by using a utility measure given by the

number of times that the constructed feature is used by C4.5. After the genetic run, the best feature set is used to induce the resulting classifier. The outcomes show that classifiers induced using the GP-constructed features are capable of outperforming the standard approach on some benchmark problems with a statistically significant level.

In [155], a GP feature constructor is wrapped around a Bayesian classifier to construct features for fingerprint classification. The experimental results show that the GP has ability to find good composite operators to effectively construct useful features for fingerprint classification problem.

In [149], where GP is utilised to construct features and then a GA is utilised to choose features from the constructed features. Both GP-based feature construction and GA-based feature selection are wrapped around C4.5, kNN and a Bayesian classifier. Each individual is a forest containing n trees, where n is the number of original features, and each tree is capable of representing a new constructed feature. The results show that the hybrid GP/GA system can achieve significant improvements to the classification accuracy of C4.5.

GP cannot directly work with incomplete data; therefore, to use GP for feature construction with incomplete data, imputation methods are required to transform incomplete data into complete data before using GP for feature construction. In order to obtain good performance, GP has to be combined with sophisticated imputation methods such as MICE [175]. Unfortunately, sophisticated imputation methods such as MICE are often suitable for batch imputation, but computationally intensive for imputing missing values in a single instance in the unseen set for classification [157]. Therefore, how to effectively and efficiently use GP for feature construction with incomplete data should be investigated.

2.4.3 GP for Extracting discriminant functions

Discriminant functions are a common way to represent classifiers. A function is a mathematical expression where different types of operators are applied to the features of an instance that need be classified. The value returned by the function determines the class predicted by using a threshold (binary classification) or set of thresholds (multiple classification) [42].

GP has been widely applied to evolve classifiers .The basic idea of the application of GP for inducing classifiers is that each individual is made to represent a classifier or a part of a classifier, a fitness function is designed to score its quality and GP acts as a search technique to discover a high quality final classifier[42].

The obvious approach to evolving discriminant functions with GP is to have a population where each individual encodes one discriminant function. The function set in the GP is any type of operations and functions that can perform on the data. GP has been widely applied to evolve discriminant functions including binary discriminant functions and multiple discriminant functions [42].

For binary classification problems, a single program is adequate; if the output value of the function is greater than a given threshold, the instance belongs to a certain class, otherwise it belongs to the other class. The threshold is often zero; therefore, a positive output associates with a particular class, while a non-positive value associates with to the other class. In [153], binary classification is tackled by the evolution of a single zero-threshold discriminant function. Two fitness measures are proposed, including the false alarm rate and the application of a multi-objective approach to simultaneously optimise two goals: classification accuracy and the a posteriori entropy of class distributions. In [2, 188], binary classification is addressed by evolving a single threshold function with a fitness function based on classification accuracy and size penalty. A single threshold function is evolved in [71], where a fitness function includes two components: a classification accuracy and a measure of certainty. In

[14, 15, 16], a single threshold discriminant function is evolved, but two special fitness functions are proposed to cope with class imbalance. In [95], the fitness function is based on the distance between the real and the predicted class.

For multiclass problems, two major approaches can be followed. One approach is to consider a *n*-class classification problem as *n-1* binary problems; hence *n-1* threshold binary functions can be utilised to discriminate the n classes [86, 146]. The other approach is to utilise only one program to distinguish all the classes, where *n-1* threshold values are required and these thresholds determine *n* intervals, each class is associated with one interval [189, 190, 191, 192]. In [86], an n-class problem is transferred into *n-1* binary problems and the fitness measure is classification accuracy. The system is run *n-1* times; in each running, a single-threshold discriminant function is generated for a particular class. A similar approach is proposed in [146], but a fitness function estimates the overlapping between the class outputs given by the classifier. In [189, 190, 191, 192], multiclass classification is tackled by generating a multiple threshold discriminant function. This function recognize the differences of n classes by means of n-1 threshold values. These thresholds make n intervals, and each interval is associated with a particular class.

GP has been mainly applied to evolve classifiers, but mainly for complete data. Therefore, to use GP for evolving classifiers for incomplete data, imputation methods are required to impute missing values before using GP. In order to evolve good classifiers, GP has to be combined with sophisticated imputation methods such as (MICE) [22] or multiple imputation for missing data using genetic programming (GPMI) [159]. However, sophisticated imputations such as MICE and GPMI are computationally expensive. Therefore, how to use GP to directly evolve classifiers able to deal with missing without imputation should be investigated.

2.5 Other Relevant Techniques

This section presents several techniques used in the thesis. It firstly presents feature selection which is used in Chapters 3, 4, and 6. It then presents clustering which used in Chapter 3.

2.5.1 Feature Selection

The purpose of feature selection is to select a subset of relevant features from the original features because many datasets often contain irrelevant and/or redundant features which can be removed without losing much information. By removing irrelevant and redundant features, feature selection can reduce the training time, simplify the classifier and improve the classification accuracy [183]. However, feature selection is a hard problem because there are 2^n possible feature subsets where n is the number of original features [23].

A feature selection method consists of two main components: an evaluation measure and a search technique [183]. The evaluation measure is used to evaluate the goodness of selected features while the search technique is used to explore new feature subsets. The quality of the feature selection method strongly depends on both the evaluation measure and the search technique [23, 183].

2.5.1.1 Evalutation Measures

Evaluation measures for feature selection can be divided into wrapper methods and filter methods [183]. A wrapper method employs a classification algorithm to score feature subsets while a filter method employs a proxy measure such as mutual information to score feature subsets. Filter methods are often more efficient and general than wrapper methods. However, wrapper methods are often more accurate than filter methods because wrapper methods directly evaluate feature subsets using classifi-

cation algorithms while filter methods are independent of any classification algorithm [183].

The Correlation Feature Selection (*CFS*) [66] is one of the most powerful filter methods, which scores feature subsets based on a correlation measure. *CFS* tends to give a good score for a feature subset which contains features highly related to the class, but uncorrelated to each other. *CFS* gives the merit for a feature subset *S* with *k* features as the following equation:

$$Merit_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{2.1}$$

where $\overline{r_{cf}}$ is the average of all feature-class correlations and $\overline{r_{ff}}$ is the average of all feature-feature correlations.

Experiments show that *CFS* can effectively identify relevant features and redundant/irrelevant features. Therefore, classification accuracy using features selected from *CFS* is often better than using all features. Moreover, in many cases, CFS is as accurate as wrapper methods and it executes much faster than the wrapper methods [66].

2.5.1.2 Search Techniques

Search techniques for feature selection can be categorised into deterministic search techniques and evolutionary search techniques [183]. Sequential forward selection (SFS) and sequential backward selection (SBS) are typical examples of deterministic search techniques [66]. In recent years, Evolutionary Computation (EC) techniques such as Genetic Algorithms (GAs), Genetic Programming(GP) and Particle Swarm Optimisation (PSO) have been successfully applied to feature selection [183]. The underlying reason is that EC techniques are good at searching for global best solutions. EC techniques also do not require domain knowledge and do not require any assumption related to the search space [183].

Particle swarm optimisation (PSO) proposed by Kennedy and Eberhart in 1995 [84, 85] is a swarm intelligence algorithm. PSO is inspired by the movement of organisms such as a bird flocking. In order to optimise a problem, PSO makes a population of particles in the search space, and moves these particles around in the search space using the information of the particles' position and velocity. The movement of each particle uses both the personal best known position and the global best known position in the search space. When enhanced positions are found, this information will be utilised to guide the movements of the swarm toward the best solution.

Algorithm 3: Particle Swarm Optimisation

Input:
$|Swarm|$: the population size
D: dimension of search space;
T : the maximum number of iterations
Output:
gbest: best fitness value

```
1  Randomly initialise the position and velocity of each particle;
2  while T or other the stopping criterion is not met do
3  |  Evaluate fitness of each particle
4  |  for i=1 to |Swarm| do
5  |  |  if fitness of x_i is better than that of pbest_i then
6  |  |  |  pbest_i = x_i ;
7  |  |  end
8  |  |  if fitness of pbest_i is better than that of gbest then
9  |  |  |  gbest = pbest_i ;
10 |  |  end
11 |  end
12 |  for i=1 to |Swarm| do
13 |  |  for d=1 to D do
14 |  |  |  update the position of particle i according to
   |  |  |    formula 2.2;
15 |  |  |  update the velocity of particle i according to
   |  |  |    formula 2.3;
16 |  |  end
17 |  end
18 end
19 return gbest and its fitness value;
```

Algorithm 3 shows the pseudo-code of PSO. Firstly, each particle is initialised with a random position and a random velocity. After that, the fitness of each particle is evaluated by a predefined fitness function, and then the velocity and the position of each particle are updated by using current *pbest* and *gbest*. Assuming that the current position and velocity of i^{th} particle are represented by a vector x_i and v_i simultaneously, then the following formulas are used to update the position and the velocity of each particle:

$$x_i \longleftarrow x_i + v_i \tag{2.2}$$

$$v_i \longleftarrow w * v_i + c_1 * r_{1i} * (p_i - x_i) + c_2 * r_{2i} * (p_g - x_i) \tag{2.3}$$

where w is inertia weight used to control the effect of the previous velocities on the current velocities; c_1 and c_2 are acceleration constants; r_{1i} and r_{2i} are random values uniformly distributed in [0, 1]; p_i and p_g indicate the local best position of i^{th} particle and the global best position, respectively.

During the search process, if the fitness of the particle is better than that of *pbest*, then its position will be saved to update the *pbest*. If the fitness of any *pbest* in the population is better than *gbest*, the *gbest* will be updated by this *pbest*. The algorithm repeatedly updates the position and velocity values of each particle to search for the best solution of the problem until a predefined stopping criterion is met. The stopping criterion can be a maximum number of iterations or a satisfactory fitness value.

One advantage of PSO is that it does not require making assumptions about the problem being optimized. Furthermore, PSO is able to search very large spaces of candidate solutions. Consequently, PSO can be used to optimise problems which are partially noisy, irregular and change over time, etc. However, the same as other evolutionary algorithms, PSO cannot ensure to find an optimal solution.

PSO has been recently used as a search technique to find feature subsets from original features in feature selection problems [94, 180, 182, 183]. If the number of original features is n, then the search space dimensionality is n. Each particle in the swarm is usually presented by a vector of n real numbers. The value of the i^{th} particle in the d^{th} dimension, x_{id}, is often in an interval [0, 1]. In order to identify whether or not a feature will be chosen, the real value in the position vector is compared with a threshold $0 < \theta < 1$. If $x_{id} < \theta$, then the d^{th} feature will be not chosen; otherwise, the d^{th} feature will be chosen.

PSO has been used for both wrapper-based and filter-based feature selection. PSO has been proved capable of having the ability to deal well with feature selection problems [26, 74, 98, 181]. However, PSO-based feature selection for incomplete datasets has not been systematically investigated.

2.5.1.3 Feature Selection for Incomplete Data

Feature selection has been mainly used on complete data, but it also has been investigated for incomplete data. In [113], kNN-based feature selection is combined with kNN-based imputation to estimate missing values for microarray data. Results show that the proposed method can achieve smaller error than the compared methods. In [9], the minimum subset of features is selected to render the rest of features independent of the class variable. In [108], the uncertainty of each instance due to the missing values is used to design a feature subset evaluation which can directly work with missing data. In [36, 128], mutual information is modified to be able to work with incomplete data. In [106], a general resampling approach is proposed to perform feature selection on incomplete data. Results show that the proposed method is suitable for both low-dimensional and high-dimensional problems. In [162], a classifier which can directly work with incomplete data is used to score feature subsets. These results show that feature selection can be useful for analysing incomplete data.

Although existing feature selection can improve classification with incomplete data, they still provide incomplete data which cannot be used by the majority of classifiers. Therefore, feature selection needs to be combined with imputation, and research is required to discover how to do this efficiently and effectively.

2.5.2 Clustering

The purpose of clustering is to partition data into different groups in such a manner that elements in each group are more similar to each other than to those in other groups. Clustering has been used in various fields such as data compression, information retrieval, pattern recognition, and bioinformatics [43].

One of the most common clustering methods is k-means clustering which groups instances into k clusters such that each instance belongs to the cluster with the nearest mean. k-means uses Euclidean distance as a metric and variance as a measure of cluster scatter. This clustering method is easy to implement, and can be used for large datasets. Therefore, k-means has been used in many fields, such as computer vision, and geostatistics [81].

Although clustering has been mainly used for complete data, it has been also used to improve imputation for incomplete data. In [96], a fuzzy clustering algorithm is used to divide instances into clusters. Subsequently, incomplete instances are imputed by using the values of cluster centroids and the information about membership degrees. In [187], complete instances are firstly grouped into different clusters. Each incomplete instance is then assigned to the most similar cluster. After that, missing values in the incomplete instance are substituted with plausible values which are generated by a kernel function on instances in the cluster. In [121], both complete and incomplete instances are divided into clusters. Subsequently, to impute missing values for an incomplete instance,

firstly, the nearest complete neighbor to the incomplete instance is identified. After that, missing values in the incomplete instance are filled by the average of centroid values and the centroidal distance of the neighbor. In [54], a clustering algorithm is used to divide complete instances into clusters, and kNN-based imputation is then used to impute missing values for each incomplete instance. A key difference between the method and the other methods is that it uses previous imputed instances to impute new incomplete instances. In [156], complete instances are firstly grouped into clusters. After that, for each incomplete instance, the closest cluster is identified by using the grey relational analysis-based distance metric. Finally, the closes instance is used to estimate missing values for the incomplete instance. In [171], missing values in medical datasets are imputed through a class-based clustering approach. Experimental results show that the proposed method can effectively impute both categorical and numeric medical data. In [166], instance selection and imputation are combined to impute missing values by using instance selection to remove noisy instances before applying imputation methods.

Although existing cluster-based imputation methods can deal with incomplete data in some extent, they require and strongly depend on a set of complete instances in original data which is not always feasible in many problems where many instances contain missing values. Moreover, clustering has been proven to improve the accuracy of kNN-based imputation, but the effect of clustering on the efficiency of this imputation method has not been investigated. There does not appear to have been any research into applying clustering to multiple imputation methods, although multiple imputation is known to be more accurate than using single imputation.

2.6 Summary

This chapter reviewed the main concepts of classification,ensemble learning, incomplete data, imputation, evolutionary computation, particularly

GP, feature construction, feature selection and clustering. This chapter also reviewed related work of classification with incomplete data.

The limitations of the existing work that form the motivations of this research were also discussed. The overall motivation is that existing approaches to classification with incomplete data mainly focus on improving the effectiveness (accuracy), but do not adequately focus on improving the efficiency. Therefore, how to improve both the effectiveness and the efficiency of classification with incomplete data needs to be investigated.

Specially, the limitation of existing work and the motivations of this research can be summarised as follows.

- Using imputation methods to estimate missing values is the most common approach to classification with incomplete data. Sophisticated imputation methods such as multiple imputation methods are accurate, but expensive, especially for classification tasks. However, no thorough work has been conducted to improve both the effectiveness and efficiency of using imputation for classification with incomplete data.

- Wrapper-based feature selection has been widely used to improve classification, but only with complete data. Several classification algorithms such as C4.5 can directly work with incomplete data, but they often generate inaccurate and complex classifiers when the data contains numerous missing values. However, there is no existing work in investigating the use wrapper-based feature selection to improve these classification algorithms.

- Ensemble learning has been used for classification with incomplete data by building a set of classifiers which can classify unseen incomplete instances without requiring imputation. However, existing ensemble methods are often inaccurate and expensive when faced with data containing numerous missing values. Therefore how to

improve the effectiveness and efficiency of ensemble methods for classification needs to be investigated.

- GP has been widely used to improve classification, but mainly with complete data. Therefore, it is needed to investigate how to effectively and efficiently use GP for classification with incomplete data.

Following Chapters

This thesis aims to address the above-mentioned issues. The following chapters will investigate those issues by developing new algorithms. Chapter 3 will develop new approaches to using clustering and feature selection to improve the effectiveness and efficiency of using imputation for classification with incomplete data. Chapter 4 will develop wrapper-based feature selection to improve classifiers able to directly work with incomplete data. Chapter 5 will develop a GP-based feature construction method which can directly work with incomplete data. Chapter 6 will develop an effective and efficient approach to classification with incomplete data by using imputation, feature selection and ensemble learning. Chapter 7 will develop GP-based classifiers which can directly work with incomplete data.

Chapter 3

Improving Imputation for Classification with Incomplete Data Using Feature Selection and Clustering

3.1 Introduction

One of the most common approaches to classification with incomplete data is to use imputation methods which replace missing values with plausible values. For example, mean imputation replaces all missing values of a feature by the average of the complete values of the feature. Imputation can transform incomplete data into imputed data that is complete, and so can be used by any classification algorithm. Moreover, imputation can improve classification accuracy [45].

However, effective imputation methods take time to estimate missing values. This may not be significant in the training process, but it is not feasible in many classification tasks to spend too much time in the application process to estimate missing values for an incomplete instance,

where the instances to be classified are typically presented one by one, and the application process must be applied to each instance separately. This is especially true for powerful imputation methods such as Multiple Imputation by Chained Equations (MICE) [175], which are computationally intensive when estimating missing values for a single incomplete instance because they must rebuild the whole imputation structure from all the training instances plus the new instance [46, 157]. Though recent research has demonstrated the increased accuracy obtained by using these advanced imputation methods, the high cost of these methods in the application process has seldom been addressed. Therefore, it is important to address the question of how to reduce the computation time in the application process without sacrificing the accuracy achieved by MICE and other advanced imputation methods. This chapter presents three novel methods that dramatically improve the efficiency of classification with incomplete data when using powerful imputation such as MICE.

Clustering is the process of categorising data into clusters such that the instances in a cluster are similar to one another and different from the instances in other clusters [43]. By selecting representative instances from the clusters, it is possible to reduce an original data set to a smaller but representative subset, so clustering has been widely used for data reduction [78]. In the application process of classification with incomplete data using MICE or other powerful imputation methods, the computation time to estimate missing values strongly depends on how many training instances are used by the imputation method. Clustering can be used to reduce the number of training instances needed for the imputation, which in turn can reduce the computation time in the application. This chapter shows how to use clustering to speed up imputation in the application process without losing accuracy.

Feature selection is the process of selecting a subset of relevant features from the original features. It has been widely used to improve classification with complete data [23, 183]. In the application process of classifica-

tion with incomplete data, the computation time of the imputation process also strongly depends on the number of features in the training data. Feature selection can remove redundant and irrelevant features of the training data which may not only improve accuracy, but can also reduce the computational cost of the imputation. This chapter also shows how to use feature selection to speed up imputation and not only retain accuracy, but even improve it.

3.1.1 Chapter Goals

This chapter proposes new methods to improve the effectiveness and efficiency of using imputation for classification with incomplete data. To achieve this goal, we propose three ways of integrating imputation, clustering and feature selection. This chapter will investigate:

1. How to integrate *clustering* into *imputation* to reduce the computation time to estimate missing values while maintaining accuracy; and

2. How to integrate *feature selection* into *imputation* to achieve better accuracy and use less computation time; and

3. How to integrate both *clustering* and *feature selection* into *imputation* to achieve better accuracy and use even less computation time.

3.1.2 Chapter Organisation

The rest of this chapter is organised as follows. Section 3.2 presents the proposed methods: the integration of clustering with imputation, the integration of feature selection with imputation, and the integration of both clustering and feature selection with imputation. Section 3.3 outlines experiment design. Section 3.4 shows empirical results and analysis. Section 3.5 draws conclusions.

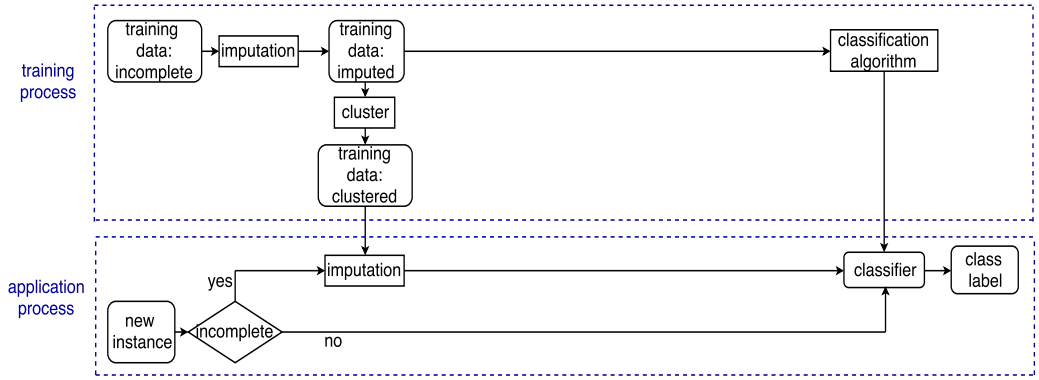


Figure 3.1: Combining *imputation* and *clustering* for classification with incomplete data.

3.2 Proposed Algorithms

This chapter proposes three new methods to improve the effectiveness and efficiency of imputation for classification with incomplete data. The first integrates imputation and clustering. The second integrates imputation and feature selection. The third integrates all three. Each of the combinations includes a training process and an application process. The training process uses training data to build a classifier while the application process then uses it to classify a new instance.

3.2.1 Integrating Imputation with Clustering

The key idea of the first method is to use clustering to produce a smaller set of representative training data to perform imputation in the application process. Consequently, the computation time to estimate missing values in the application process can be reduced.

Figure 3.1 shows the main steps of this method. As usual, in the training process, an imputation method is used to estimate missing values in the incomplete training data, and the imputed training data is used by a classification algorithm to build a classifier. The imputed training data is

also put into a clustering algorithm to construct clusters of data instances. The method then extracts one central instance from each cluster to form a smaller training dataset which will be used to estimate missing values in the application process. When the application process needs to classify a complete new instance, it can be directly classified by the classifier. If it needs to classify an incomplete instance, the instance will be combined with the clustered training data, and put into the imputation method to estimate missing values. Finally, the imputed instance is classified by the classifier.

Although doing the clustering increases the cost during training, training time is not considered so important in many real-world problems. Using the clustered data to estimate missing values might slightly reduce the accuracy of imputation in test data because there is less information, although this may be counteracted by improved quality of the imputation if clustering is able to identify more representative data points. However, the computation time of many imputation methods especially kNN-based imputation strongly depends on the number of instances which are used to estimate missing values. The number of instances in clustered data is much smaller than the number of instances in the original data. Therefore, in the application process, by using clustered data with a smaller number of instances to estimate missing values for incomplete instances, the proposed method can speed up the application process.

3.2.2 Integrating Imputation with Feature Selection

The second method uses feature selection to remove redundant and irrelevant features from the imputed training data. This is known to improve the "quality" of training data and help to construct better classifiers. However, the removed features also need to be applied in the application process, and this can reduce the number of incomplete instances (reducing the need for expensive imputation) and also reduce the cost if the imputation

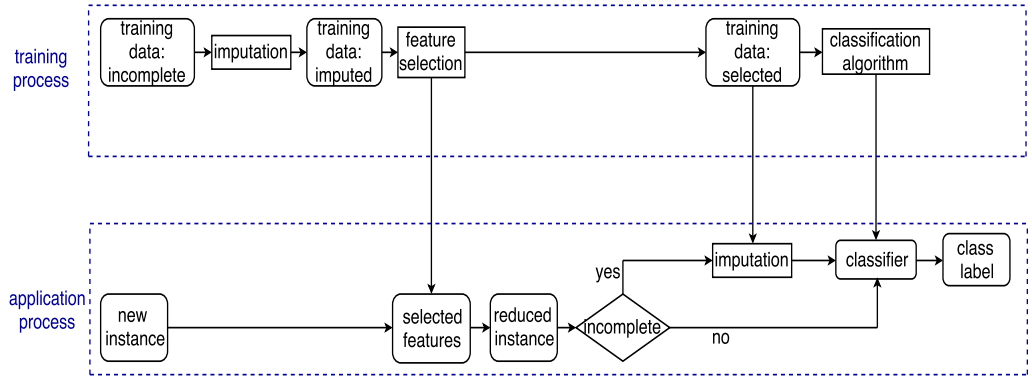


Figure 3.2: Combining *imputation* and *feature selection* for classification with incomplete data.

process when it is required. Consequently, the computation time for the application process can be reduced.

Figure 3.2 shows the main steps of the second method. In the training process, first an imputation method is used to estimate missing values in the incomplete training data. After that, a feature selection method is used to remove redundant and irrelevant features in the imputed training data. The imputed training data with selected features only is then put into a classification algorithm to build a classifier. In the application process, when a new instance needs to be classified, firstly redundant and irrelevant values in the instance are removed by only keeping selected features. If the reduced instance is complete, it is directly classified by the classifier. Otherwise, missing values in the instance are estimated by using the imputation method and the training data with selected features only. Finally, the imputed instance is classified by the classifier.

Feature selection has several advantages. First, feature selection can improve classification accuracy and build more compact classifiers for many classification algorithms. Second, by removing features from new instances (which may have missing values), it can reduce the number of incomplete instances in the application process which in turn will reduce the computation time of the application process. Third, since the computation time

of many imputation methods, especially MICE, strongly depends on the number of features in the data which is used to estimate missing values, using the selected data with a smaller number of features to estimate missing values for incomplete instances, the proposed method can speed up the imputation in the application process.

Performing feature selection in this way may increase the cost during training, though this may be somewhat compensated by decreased cost of the classification algorithm on the reduced data. However, cost during training is not generally the primary concern, and the cost saving during application is much more important.

3.2.3 Integrating Imputation, Feature Selection and Clustering

The third method combines the innovations of the first two methods. It uses feature selection to remove redundant and irrelevant features, and uses clustering to reduce the number of instances for the imputation step of the application process. Unlike the first method, the clustering is applied to the imputed training data with selected features, rather than the full imputed training data. Removing irrelevant and redundant features can improve the clustering quality, as well as improving the classification accuracy, and reducing the computation time of the imputation step in the application process. The computation time is further reduced by the clustering step which reduces the number of instances used in the imputation step. Although clustering and feature selection take time in the training process, the time saved by their results in the application process is more important.

Figure 3.3 shows the main steps of the third method. In the training process, the incomplete training data is put into an imputation method to estimate missing values. The imputed training data is then put into a feature selection method to remove redundant and and irrelevant fea-

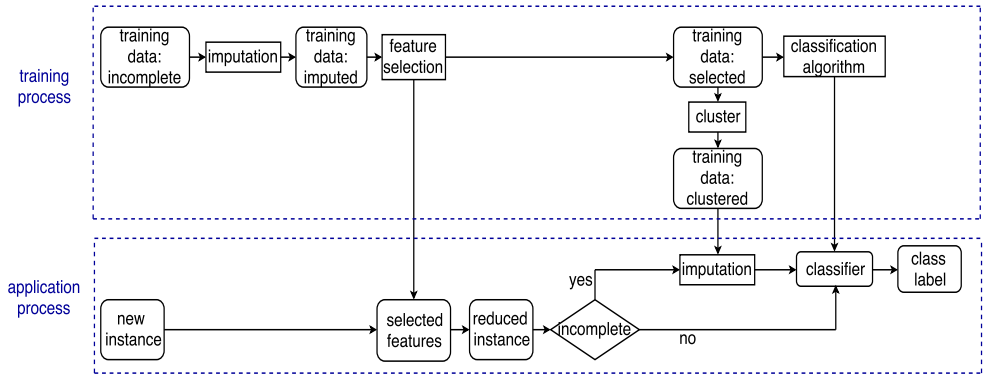


Figure 3.3: Combining *imputation*, *feature selection* and *clustering* for classification with incomplete data.

tures. Following that, the training data with selected features is used by a classification method to build a classifier. The training data with selected features is also used by a clustering method to generate a smaller training dataset which is then used to estimate missing values in the application process. In the application process, when a new instance needs to be classified, firstly, redundant and irrelevant values are eliminated by only keeping the selected features. Subsequently, if the reduced instance is complete, it will be directly classified by the classifier. Otherwise, it is combined with the clustered training data, and then used by the imputation method to estimate missing values. Finally, the imputed instance is classified by the classifier.

3.3 Design of Experiments

This section presents the comparison methods, datasets used in experiments and parameter settings.

3.3.1 The Comparison Methods

We perform experiments to evaluate the effectiveness and efficiency of the proposed methods. To evaluate the combination of imputation and clus-

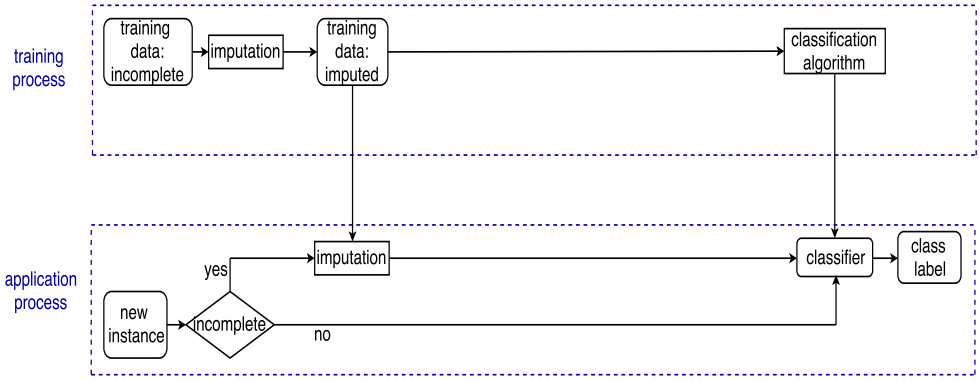


Figure 3.4: A common approach to using imputation for classification with incomplete data.

tering shown in Figure 3.1, and the combination of imputation and feature selection shown in Figure 3.2, both methods are compared with the method that only uses imputation shown in Figure 3.4. The combination of imputation, feature selection and clustering as shown in Figure 3.3 is compared with the method using only imputation shown in Figure 3.4, the combination of imputation and clustering shown in Figure 3.1, and the combination of imputation and feature selection shown in Figure 3.2.

We also perform experiments to compare the proposed methods with recent benchmark methods. The proposed method combing imputation and clustering is compared with three recent benchmark methods from [121, 54, 166], which apply clustering to improve imputation in a different way. The proposed method combing imputation and feature selection is compared with two recent benchmark methods in [36, 128], which directly perform feature selection on incomplete data.

3.3.2 Datasets and Parameter Settings

The proposed methods are tested on 15 real-world incomplete datasets. The main characteristics of the datasets are shown in Table 1.1 in page 18. Ten-fold cross-validation is used to divide the datasets into training and

test datasets. This is particularly appropriate because the number of instances in some datasets is relatively small. The ten-fold cross-validation process is stochastic, so it needs to be performed multiple times and the results averaged. The ten-fold cross-validation ensures fractions of each class in each fold are as close to the fraction in other folds as possible. It is not feasible to also ensure an equal fraction of missing data in each fold because this would constrain the choices too much. Therefore, whether items have missing data is ignored during the allocation to folds. To mitigate the possibility of unfair results due to a fold having too much or too little missing data, the ten-fold cross-validation is repeated 30 times to ensure that on average, there is a reasonable distribution of missing data across the folds.

The experiments use two imputation methods: kNN-based imputation and MICE. These imputation methods are selected to represent two categories of imputation methods: single imputation and multiple imputation, respectively. kNN-based imputation with k=1 is used since it is simple and fast. Multivariate imputation by chained equations in R [22] is used for MICE's implementation. In MICE, random forest [97] is used as a regression method to estimate missing values. The number of cycles is set five and the number of imputed datasets is set 20 following the recommendation in [175].

The k-means clustering algorithm is used to cluster data. WEKA [65] is used to implement the clustering algorithm. The number of k in the clustering algorithm is set to a square root of the number of instances [81].

The experiments use three classification algorithms: C4.5, k-nearest neighbour (kNN) and Naive-Bayes (NB). These classification algorithms are selected to represent three categories of classifiers: rule-based learning, lazy learning and approximate models, respectively. WEKA [65] is used to implement the classification algorithms. The default parameter values in WEKA are used for building classifiers (eg. C4.5 with pruning, NB with kernel density estimator for numeric attributes, and kNN with

k=1 and Euclidean distance). Default parameter settings in WEKA are not always appropriate. However, the thesis uses the same default parameter settings for both the proposed methods and baseline methods. Therefore, the comparison between the proposed methods and the baseline methods are still fair.

The experiments use a wrapper method to perform feature selection. A classifier using in the later classification step (C4.5, kNN and NB) is used to evaluate feature subsets. PSO is used for searching feature subsets. The parameters of *PSO* in the feature selection method are chosen according to common settings proposed by Clerc and Kennedy [27]. The parameters of PSO are set as follows: ω = 0.729844, c_1 = c_2 = 1.49618, the population size is 50, the maximum number of generations is 100.

3.4 Results and Discussions

This section presents comparisons of classification accuracy and computation time between the proposed methods and the benchmark methods. Analysis is also performed to investigate key reasons why the proposed methods are effective and efficient.

Abbreviations are used to refer to different methods. "kNNI" is the baseline method as shown in Figure 3.4 that uses only kNN-based imputation. "kNNICl" is the method as shown in Figure 3.1 that combines kNN-based imputation and clustering. "kNNIFs" is the method as shown in Figure 3.2 that combines kNN-based imputation and feature selection. "kNNIClFs" is the method as shown in Figure 3.3 that combines kNN-based imputation, feature selection and clustering. "MICE", "MICECl", "MICEFs", and "MICEFsCl" refer to methods using MICE in Figure 3.4, Figure 3.1, Figure 3.2 and Figure 3.3, respectively.

Table 3.1 shows mean and standard deviation of classification accuracy of the baseline method in Figure 3.4 and the proposed methods when using kNN-based imputation and MICE. Each classification accuracy is the

average of accuracy over 30 times for doing ten-fold cross-validation on each dataset. The Friedman test [33], which is one of the most commonly used non-parametric tests for multiple comparisons, is used to statistically test whether the differences in classification accuracy are significant. The test shows that for all datasets, there exists significant differences between the methods. Therefore, a post-hoc procedure using the Holm procedure [33] is used to carry out pairwise comparisons. The symbol $\uparrow$ indicates that the proposed method is significantly better than the benchmark method. In contrast the symbol $\downarrow$ shows that the proposed method is significantly worse than the benchmark method.

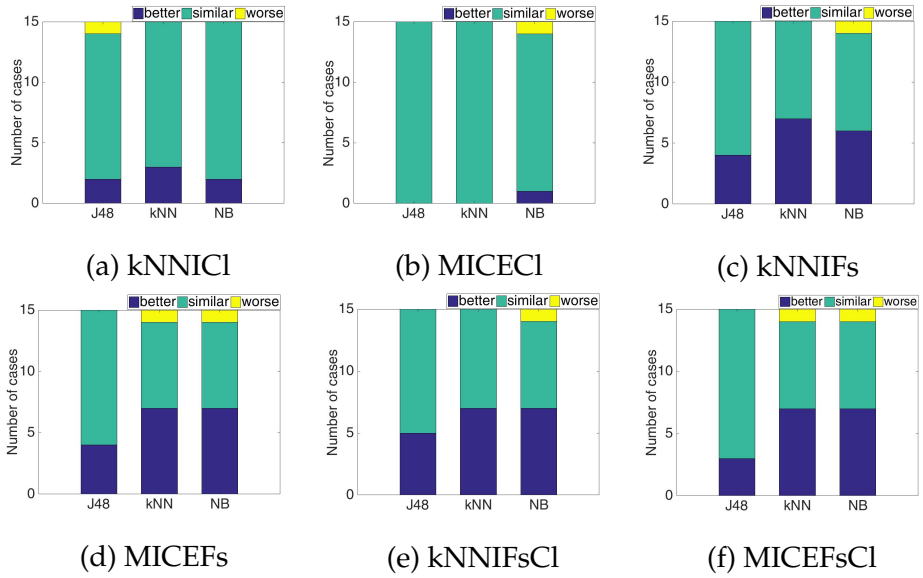
Figure 3.5 summarises the classification accuracy comparison between the proposed methods as shown in Figures 3.1, 3.2 and 3.3 with the baseline method using only imputation as shown in Figure 3.4. In Figure 3.5, each bar shows the fraction of times the accuracy of the proposed method is better (statistically significant), similar (difference is not statistically significant) or worse than the baseline method for each of the three classification algorithms.

Figure 3.6 shows the magnitude of the reduction in computation time of the proposed methods as a fraction of the computation time of the baseline method. For example, Figure 3.6b shows that using kNN classifier and kNN-based imputation, the feature selection method saves 80% of the computation time of the baseline. Note that the absolute time is often higher for MICE than for kNN-based imputation. In Figure 3.6, each bar shows the reduction in computation time relative to the baseline method obtained by using the proposed methods for each imputation method and each classification algorithm.

Table 3.2 shows the classification accuracy of six other benchmark methods. Cl[121], Cl[54] and Cl[166] refer to three benchmark methods in [121, 54, 166] which use clustering to improve imputation. Fs[36] and Fs[128] refer to the benchmark in [36, 128] which directly perform feature selection with incomplete data.

Table 3.1: The mean and standard deviation of classification accuracies of the baseline method in Figure 3.4 and the proposed methods.

Data	Class	kNNI				MICE			
		kNNI	**kNNICl**	**kNNIFs**	**kNNIFsCl**	**MICE**	**MICECl**	**MICEFs**	**MICEFsCl**
Arr	J48	65.58±1.54	65.58±1.54	67.06±2.39	67.06±2.39	65.45±1.53	65.45±1.53	66.34±0.86	66.34±0.86
	kNN	52.98±1.38	53.01±1.33	58.16±2.28↑	58.16±2.28↑	52.73±1.17	52.64±1.22	57.92±1.80↑	57.97±1.87↑
	NB	60.48±1.37	60.69±1.48	66.20±1.21↑	66.18±1.26↑	61.00±1.33	61.00±1.33	65.99±1.61↑	66.04±1.58↑
Aut	J48	67.21±4.11	67.11±4.17	66.64±4.48	66.34±4.43	67.79±3.92	67.74±3.94	67.72±2.90	67.42±3.14
	kNN	68.97±3.39	69.06±3.30	70.28±5.83	69.40±5.74	69.06±3.61	69.02±3.61	69.03±4.31	68.74±4.38
	NB	55.39±3.76	54.95±3.79	61.99±3.20↑	61.12±3.99↑	56.07±3.59	56.12±3.65	59.66±4.22↑	60.00±4.22↑
Ban	J48	67.32±1.84	67.03±1.95	65.75±1.28	65.56±1.14	71.35±2.11	70.24±1.98	70.56±1.29	70.35±1.43
	kNN	68.62±1.10	68.25±1.18	69.93±1.90	69.03±1.93	72.25±0.87	71.79±1.16	71.07±1.86	71.92±1.58
	NB	63.49±1.02	63.34±1.11	62.62±1.50	62.79±1.45	65.10±0.63	63.22±0.82↓	64.51±1.38	64.34±1.25
Bre	J48	95.12±0.31	95.07±0.32	94.92±0.38	94.82±0.37	94.94±0.36	94.92±0.41	94.78±0.54	94.81±0.54
	kNN	95.21±0.23	95.25±0.23	94.93±0.39	94.90±0.43	95.24±0.22	95.24±0.22	94.79±0.54	94.87±0.51
	NB	95.98±0.09	95.98±0.09	95.84±0.42	95.87±0.43	96.01±0.12	96.01±0.12	95.98±0.42	95.96±0.37
Chr	J48	98.52±0.51	98.62±0.50	98.42±0.61	98.32±0.67	97.37±0.56	96.95±0.67	96.92±0.76	96.89±0.58
	kNN	97.97±0.39	98.12±0.20	98.05±0.38	97.80±0.35	98.60±0.37	98.47±0.23	97.97±0.50	97.89±0.71
	NB	93.60±0.42	94.10±0.53	96.62±0.72↑	96.10±0.75↑	95.37±0.43	95.35±0.60	96.77±0.66↑	96.67±0.53↑
Cle	J48	52.75±2.00	52.71±2.06	57.92±0.87↑	57.89±0.87↑	53.05±2.16	53.09±2.16	57.95±1.41↑	57.84±1.41↑
	kNN	54.15±1.04	54.13±1.01	55.22±1.90	55.10±1.89	54.47±1.10	54.36±0.97	55.68±1.20	55.54±1.23
	NB	57.11±0.91	57.17±0.87	56.21±1.38	56.33±1.43	57.50±0.73	57.45±0.71	57.30±0.99	57.39±0.88
Cre	J48	85.38±0.73	85.42±0.72	85.65±0.47	85.66±0.48	85.22±0.56	85.21±0.59	85.43±0.85	85.38±0.86
	kNN	82.42±0.37	82.70±0.47↑	84.07±0.90↑	84.00±0.93↑	82.86±0.41	82.83±0.40	83.54±0.62↑	83.50±0.65↑
	NB	77.18±0.51	77.30±0.51	86.18±0.46↑	86.20±0.50↑	77.26±0.44	77.21±0.46	86.56±0.73↑	86.55±0.73↑
Hea	J48	79.49±0.70	79.47±0.73	81.20±1.16↑	81.76±0.83↑	78.57±1.33	78.73±1.78	80.49±0.80↑	79.89±1.25
	kNN	76.58±1.39	76.72±0.83	80.39±0.99↑	80.10±1.10↑	76.67±1.17	77.11±1.74	80.78±1.40↑	78.73±0.98↑
	NB	82.79±0.57	83.08±0.44↑	81.10±0.93	80.85±1.14	82.79±0.43	82.79±0.59	81.39±1.11	81.32±1.20
Hep	J48	78.11±2.10	78.98±2.12↑	78.23±2.38	78.55±2.85	79.44±2.44	78.81±2.53	81.98±0.69↑	81.43±2.03↑
	kNN	78.16±0.83	79.03±1.04↑	78.48±2.66	79.00±2.19	80.53±1.68	80.61±1.65	77.30±2.26↓	77.80±2.58↓
	NB	83.93±1.31	84.25±0.93	80.46±2.29	81.18±2.80	83.28±0.50	83.33±0.85	82.40±1.81	82.30±2.00
Hor	J48	83.19±0.64	83.21±0.91	83.95±0.84	84.13±0.94↑	84.26±0.78	84.38±0.64	84.34±0.62	84.17±0.64
	kNN	77.66±1.14	77.02±1.85	81.49±1.85↑	81.91±1.55↑	76.14±1.30	75.67±1.12	79.67±1.54↑	79.97±1.06↑
	NB	77.03±1.18	75.93±1.05	83.03±0.88↑	83.50±0.75↑	77.61±0.46	76.92±1.08	81.44±1.05↑	80.89±1.36↑
Hou	J48	96.24±0.62	96.29±0.58	96.26±0.66	96.33±0.61	96.15±0.54	96.15±0.60	96.22±0.57	96.22±0.56
	kNN	92.20±0.66	92.48±0.48	94.58±0.99↑	94.64±1.07↑	92.95±0.41	93.07±0.51	95.10±0.65↑	95.13±0.51↑
	NB	90.20±0.24	90.14±0.31	95.10±0.60	95.14±0.63	91.11±0.20	91.11±0.23	95.72±0.57	95.72±0.43
Mam	J48	82.38±0.39	82.42±0.42	82.42±0.57	82.38±0.61	82.46±0.45	82.33±0.33	82.69±0.51	82.59±0.47
	kNN	75.52±0.60	75.78±0.81	82.14±0.63↑	82.01±0.60↑	75.98±0.64	76.01±0.66	82.43±0.43↑	82.40±0.45↑
	NB	80.63±0.51	80.69±0.40	80.72±0.55	80.82±0.53	80.73±0.37	80.64±0.42	80.47±0.76	80.43±0.80
Mar	J48	30.21±0.55	30.16±0.51	32.97±0.32↑	33.00±0.33↑	30.02±0.39	29.92±0.47	32.86±0.45↑	32.86±0.44↑
	kNN	27.60±0.37	27.65±0.29	32.29±0.47↑	32.28±0.43↑	27.57±0.38	27.50±0.33	31.81±0.46↑	31.84±0.47↑
	NB	30.67±0.34	30.75±0.35	31.87±0.40↑	31.89±0.42↑	30.61±0.29	30.61±0.24	32.27±0.29↑	32.26±0.30↑
Ozo	J48	95.74±0.79	95.93±0.37↑	96.39±0.77↑	96.54±0.35↑	95.89±0.41	95.88±0.42	96.12±0.42	96.12±.42
	kNN	96.69±0.27	96.79±0.15↑	96.63±0.29	96.74±0.16	96.77±0.16	96.79±0.17	96.79±0.12	96.78±0.13
	NB	70.89±1.42	73.06±1.78↑	97.01±0.13↑	97.03±0.10↑	71.46±0.44	72.27±0.50↑	96.18±1.16↑	96.16±1.18↑
Tum	J48	40.61±2.51	39.63±2.64↓	38.78±3.15	38.25±3.79	40.35±2.44	39.78±2.32	39.05±2.10	38.57±1.89
	kNN	37.82±1.55	38.23±1.94	36.71±1.72	37.72±1.31	38.30±2.35	37.82±1.89	37.04±2.11	37.25±2.56
	NB	45.50±1.45	45.62±2.02	42.02±2.35↓	42.82±2.06↓	46.58±1.74	46.28±1.31	43.36±1.69↓	43.12±1.71↓



(a) kNNICl (b) MICECl (c) kNNIFs

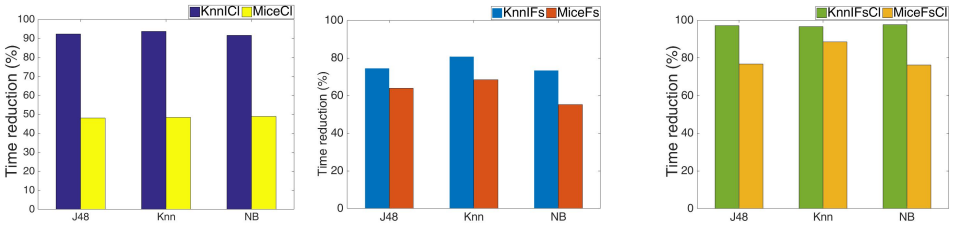
(d) MICEFs (e) kNNIFsCl (f) MICEFsCl

Figure 3.5: Classification accuracy comparison between the proposed methods as shown in Figures 3.1, 3.2 and 3.3 with using *only imputation* as shown in Figure 3.4.

3.4.1 Integrating Imputation and Clustering

Figures 3.5a and 3.5b show the comparison of the classification accuracy between the combination of imputation and clustering and only imputation. It is clear from Figure 3.5a that the combination of kNN-based imputation with clustering can achieve at least similar classification accuracy, and in several datasets significantly better accuracy than using only kNN-based imputation. When kNN is also used as a classifier, clustering can help to improve classification accuracy. Moreover, as demonstrated in Figure 3.5b, the combination of MICE and clustering can achieve comparable accuracy to using only MICE.

Figure 3.6a shows the reduction in computation time relative to the baseline method obtained by combining clustering and imputation. It is clear from Figure 3.6a that clustering can impressively reduce the computation time to estimate missing values for the application process —by



(a) kNNICl, MICECl (b) kNNIFs, MICEFs (c) kNNIFsCl, MICEFsCl

Figure 3.6: The reduction in computation time relative to the baseline method obtained by using the proposed methods.

90% and 50% when using kNN-based imputation and MICE, respectively. Clustering achieves this saving by reducing the number of instances that need to be used by imputation in the application process. For kNN-based imputation, where the cost of imputing missing values in a new instance is linearly dependent on the number of instances to search, so the relative speed gain from using clustering is very significant. In our experiments with MICE, we used random forest for the regression step. This is a particularly effective regression method which appears to have a sublinear dependence on the number of instances. The relative speed gained due to clustering is therefore not as dramatic.

The proposed method integrating clustering into imputation is also compared against of three recent benchmark methods combining clustering and imputation in [121, 54, 166] (detailed results are shown in Table 3.2). Figure 3.7 compares the classification accuracy of the methods.

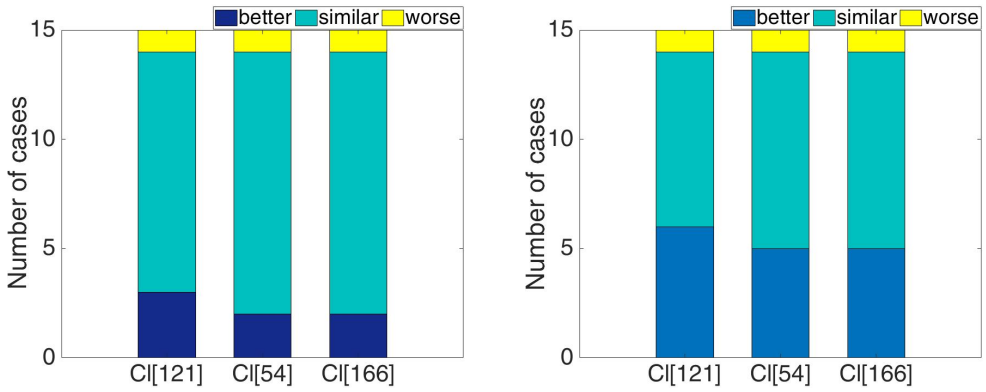
Figure 3.7a shows that the proposed method using kNN-based imputation is very similar to the benchmark methods. Figure 3.7b shows that the proposed method using MICE can frequently achieve better accuracy than the benchmark methods. The underlying reason is that MICE, which is a multiple imputation method, is often better than the single imputation methods which are used in the benchmark methods. Especially, the proposed method is much better than the benchmark methods

Table 3.2: Classification accuracy of some benchmark methods.

Data	**Cl[121]**	**Cl[54]**	**Cl[166]**	**Fs[36]**	**Fs[128]**
Arr	56.28±0.84	58.23±0.78	58.27±0.94	58.15±1.01	58.19±1.12
Aut	52.65± 3.45	53.54±3.25	52.78±3.89	54.85±2.95	58.01±3.01
Ban	68.20±1.23	68.43±1.18	68.37±1.24	69.87±1.12	69.98±1.21
Bre	95.20±0.21	95.26±0.23	95.31±0.19	94.87±0.45	94.95±0.51
Chr	98.31±0.24	98.28±0.27	98.37±0.29	98.12±0.41	98.06±0.43
Cle	54.23±0.97	54.29±1.04	54.24±0.99	55.32±1.82	55.49±1.93
Cre	86.97±0.42	86.87±0.34	87.02±0.37	86.25±0.71	86.57±0.81
Hea	N/A	N/A	N/A	78.50±0.87	79.21±1.02
Hep	81.89±1.24	81.57±1.32	81.65±0.12	82.14±2.12	82.16±2.31
Hor	73.13±2.03	74.95±2.12	73.25±2.21	79.23±1.21	80.01±1.23
Hou	93.60±0.51	93.70±0.42	93.58±0.39	94.61±0.31	94.45±0.35
Mam	78.52±0.61	78.43±0.45	78.87±0.48	82.88±0.65	82.97±0.64
Mar	28.20±0.32	28.34±0.29	28.19±0.31	32.08±0.49	32.01±0.48
Ozo	96.78±0.13	96.77±0.12	96.75±0.18	96.62±0.14	96.65±0.18
Tum	37.92±1.87	38.01±1.89	38.12±1.91	36.81±1.87	36.92±1.82

in datasets which contain many incomplete instances. The key reason is that the benchmark methods strongly depend on the number of complete instances to perform clustering, so they cannot work well when datasets contain a small number of complete instances such as the *Heart Disease* dataset and the *Horse-colic* dataset.

In order to further understand the impact of integrating clustering with imputation, the proposed method is evaluated with five different numbers of clusters—$n^{0.2}, n^{0.4}, n^{0.5}, n^{0.6}$ and $n^{0.8}$, where n is the number of instances in the training data. Figure 3.8 shows the accuracy improvement (negative shows deterioration) by combining kNN-based imputation and clustering with different number of clusters. Figure 3.9 shows the relative time reduction by combining kNN-based imputation and clustering with differ-



(a) kNNICl (b) MICECl

Figure 3.7: Classification accuracy comparison between the proposed method combining clustering and imputation against three benchmark methods using clustering to improve imputation in [121, 54, 166] (using kNN as a classifier).

ent numbers of clusters.

As expected, Figure 3.9 shows that clustering with a smaller numbers of clusters leads to a greater reduction in computation time. Figure 3.8 shows that there is not much effect on accuracy between $n^{0.2}$ and $n^{0.8}$ clusters, but the best accuracy appears to be around $n^{0.5}$ which is the commonly used value of clustering. Therefore, the common setting for the number of clusters should be used when combining imputation and clustering.

In summary, the integration of clustering with imputation can markedly speed up imputation with no loss in accuracy.

3.4.2 Integrating Imputation and Feature Selection

Figures 3.5c and 3.5d show the comparison between the combination of imputation and feature selection and using only imputation. It is clear from Figures 3.5c and 3.5d that feature selection can help to improve clas-

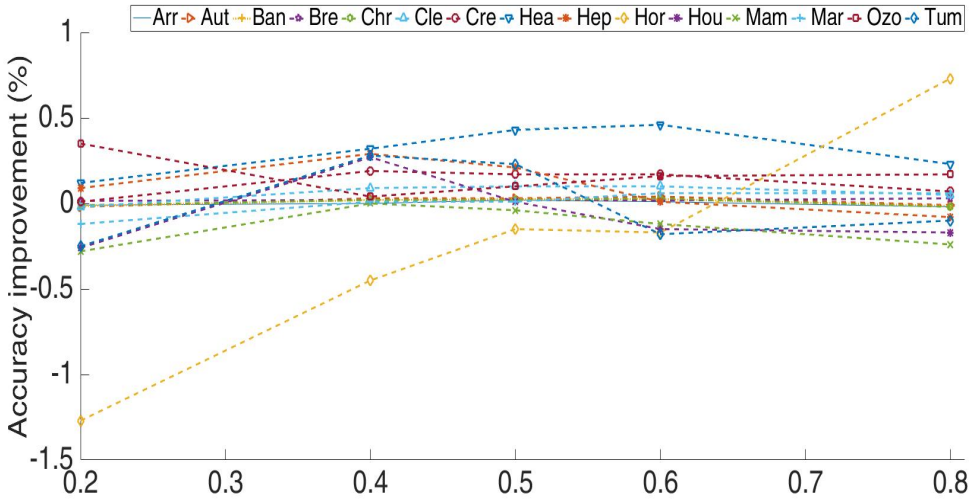


Figure 3.8: The accuracy improvement by integrating clustering into kNN-based imputation with different numbers of clusters(using kNN as a classifier).

sification accuracy when it is combined with imputation. With a Naive Bayes classifier, feature selection significantly improves classification accuracy in over half of the datasets (by about 25% with the Ozone dataset, and by nearly 10% with the Credit dataset). With a kNN classifier, feature selection also improves classification accuracy in at least half the datasets, but not as dramatically as with Naive Bayes. Feature selection may also improve classification accuracy of a J48 classifier, but less often than the Naive Bayes classifier and the kNN classifier. The reason could be that J48 classifier is a decision tree which can implicitly perform feature selection [152], so explicitly performing feature selection cannot improve the J48 classifier very much. However, both Naive Bayes and kNN classifiers are badly affected by irrelevant or redundant features, so explicitly performing feature selection can have a large effect on accuracy.

Figure 3.6b shows the average reduction in time by combining imputation and feature selection relative to using only imputation. As can seen

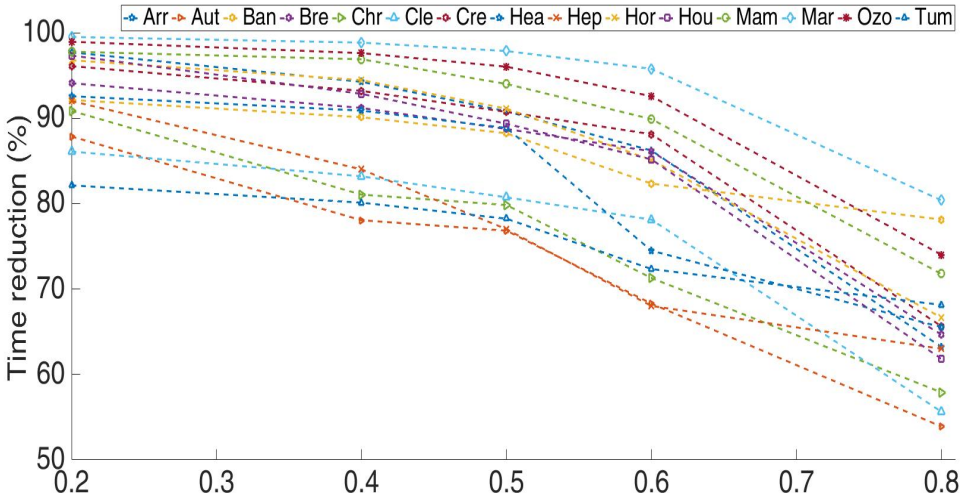
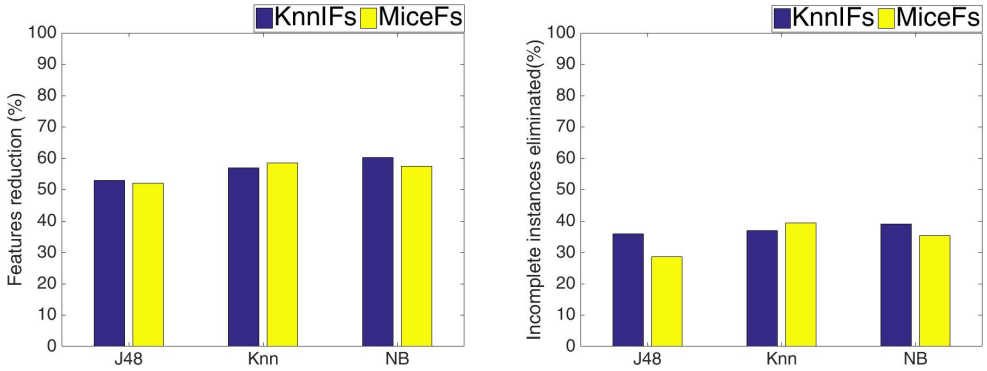


Figure 3.9: The reduction in computation time relative to the baseline method obtained by integrating clustering into kNN-based imputation with different numbers of clusters (using kNN as a classifier).

from Figure 3.6b, feature selection can help remarkably reduce the computation time to estimate missing values for the application process. Feature selection can reduce about 70% and 60% of the computation time when it is combined with kNN-based imputation and MICE, respectively. There are two reasons for this. Reducing the number of features helps speed up the computation process since the cost of the imputation methods depends on the number of features, as well as the number of instances. Figure 3.10a shows that feature selection can eliminate at least half of the original features. Secondly, feature selection can reduce the number of incomplete instances and therefore reduce the number of test instances that require imputation. Figure 3.10b shows that feature selection turns about 30% incomplete instances into complete instances.

The proposed methods integrating feature selection into imputation are also compared against of two recent benchmark methods applying feature selection for incomplete data in [36, 128] (detailed results are shown



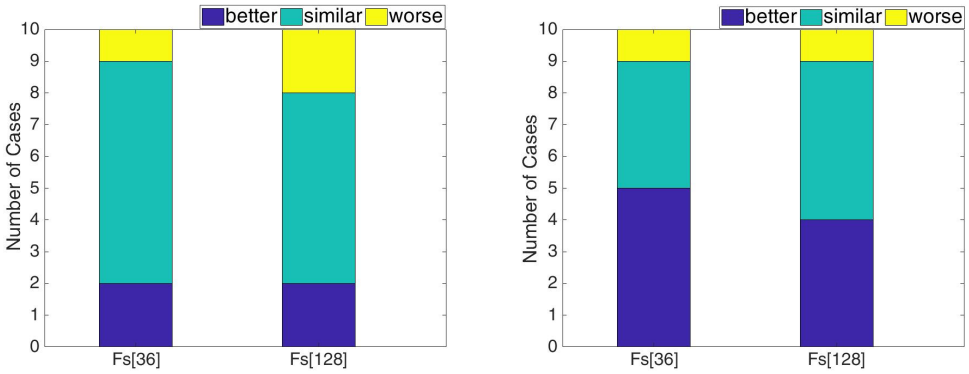
(a) Feature reduction (b) Incomplete instance reduction

Figure 3.10: The reduction in number of features and the fraction of incomplete instances that were turned into complete instances.

in Table 3.2). Figure 3.11 compares the classification accuracy of the methods. Figure 3.11a shows that the proposed method with kNN-based imputation is very similar to the benchmark methods; Figure 3.11b shows that the proposed method with MICE can be more accurate than the benchmark methods. The key reason is that MICE can achieve better accuracy than the single imputation methods in the benchmark methods.

The two main components of feature selection are an evaluation technique and a search technique. Both wrapper methods and filter methods can be used to evaluate feature subsets. Wrapper methods are often more expensive than filter methods, but wrapper methods are often more accurate than filter methods. In the proposed method, feature selection is only applied during the training process, not the application process. Since the primary concern of our approach is the computation time in the application process, it does not matter that wrapper is more expensive, and we use wrapper.

Our approach could use any evolutionary search technique. We uses PSO in our experiments because it is a powerful technique. However, we also perform experiments to compare PSO with two other evolutionary



(a) kNNIFs (b) MICEFs

Figure 3.11: Classification accuracy comparison between the proposed method integrating feature selection into imputation and two benchmark methods in [36, 128] (using kNN as a classifier).

techniques, GAs and DE that have been more commonly used for feature selection to ensure that PSO is a reasonable choice. Table 3.3 shows the classification accuracy by using PSO, GA, DE for feature selection in the proposed method as shown in Figure 3.2. The experiments show that different classification algorithms have the same pattern, so we only report the results of kNN.

Figure 3.12 summarises the classification accuracy of feature selection integrating with imputation using PSO, GA and DE as search techniques (detailed results are shown in Table 3.3). Figure 3.12 shows that PSO achieves very similar results to DE and slightly better results than GA. Therefore, PSO is a suitable choice for search method.

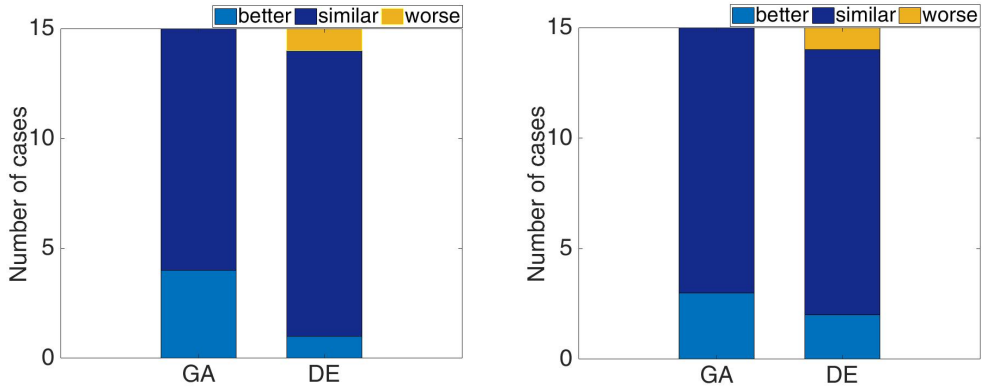
In summary, the integration of feature selection with imputation not only improves classification accuracy, but also reduces computation time.

Table 3.3: The classification accuracy by using DE, PSO and GA for feature selection (kNN is used as a classifier).

Data	kNNI			MICE		
	PSO	GA	DE	PSO	GA	DE
Arr	58.16±2.28	55.12±2.19↓	58.23±2.14	57.92±1.80	55.02±1.96	57.99±1.83
Aut	70.28±5.83	70.12±5.46	66.23±5.67↓	69.03±4.31	68.25±4.28	66.75±4.34↓
Ban	69.93±1.90	69.82±1.93	70.02±1.87	71.07±1.86	71.03±1.89	71.18±1.79
Bre	94.93±0.39	94.91±0.41	94.89±0.38	94.79±0.54	94.63±0.57	94.80±0.52
Chr	98.05±0.38	97.95±0.41	98.02±0.36	97.97±0.50	97.86±0.49	98.01±0.42
Cle	55.22±1.90	55.24±1.86	55.26±1.85	55.68±1.20	55.59±1.22	55.78±1.18
Cre	84.07±0.90	83.28±0.94↓	85.23±0.87↑	83.54±0.62	82.76±0.67↓	84.46±0.58↑
Hea	80.38±0.99	80.47±0.95	80.26±1.03	80.78±1.40	80.97±1.38	80.52±1.44
Hep	78.48±2.66	78.36±2.69	78.17±2.74	77.30±2.26	77.28±2.21	77.15±2.13
Hor	81.49±1.85	79.37±1.89↓	80.95±1.92	79.67±1.54	78.53±1.57↓	78.65±1.87↓
Hou	94.58±0.99	94.46±1.03	94.52±1.04	95.10±0.65	95.15±0.62	95.03±0.69
Mam	82.14±0.63	81.56±0.67↓	82.20±0.61	82.43±0.43	81.57±0.49↓	82.63±0.37
Mar	32.29±0.47	32.16±0.49	32.32±0.41	31.81±0.46	31.78±0.49	31.92±0.42
Ozo	96.63±0.29	96.53±0.31	96.68±0.23	96.79±0.12	96.68±0.19	96.81±0.11
Tum	36.71±1.72	36.65±1.81	36.69±1.73	37.04±2.11	36.98±2.21	37.02±2.14

3.4.3 Integrating Imputation, Feature Selection and Clustering

Figures 3.5e and 3.5f show the comparison between the combination of imputation, feature selection and clustering and only imputation. It is clear from Figures 3.5e and 3.5f that the combination of feature selection and clustering also can help improve classification accuracy when it is combined with imputation. The Naive Bayes classifier still obtains the most benefits from the combination, followed by the kNN classifier and the J48 classifier. Further comparisons show that using both of feature selection and clustering can achieve better accuracy than using only clustering, and



(a) kNN-based Imputation (b) MICE

Figure 3.12: Classification accuracy of GA and DE against PSO for feature selection.

can obtain comparable accuracy to using only feature selection.

Figure 3.6c shows the average reduction in time by combining imputation, feature selection and clustering relative to using only imputation. Figure 3.6c shows that the combination of feature selection with clustering dramatically reduce the computation time for estimating missing values in application process. The combination of feature section and clustering can reduce about 95% and 80% of the computation time when it is combined with kNN-based imputation and MICE, respectively. The combination of feature selection and clustering is also quicker than either feature selection or clustering. The key reason is that the combination of feature selection and clustering not only reduces the number of features, but also reduces the number of instances in training data and the number of incomplete instances in the test set which helps reduce the computation time.

In summary, the integration of clustering and feature selection with imputation not only can improve accuracy, but also can dramatically reduce the computation time in the application process.

Table 3.4: Gene expression datasets used in the experiments.

Name	#Samples	#Genes	#Classes	Incomplete samples (%)	Incomplete genes (%)
alizadeh-2000-v1	42	1095	2	100	65.66
alizadeh-2000-v2	62	2093	3	100	80.21
bredel-2005	50	1739	3	100	33.12
chen-2002	180	85	2	59.77	81.17
garber-2001	66	4553	4	100	44.47
liang-2005	37	1411	3	100	22.39
tomlins-2006	104	2315	5	100	87.65
tomlins-2006-v2	92	1288	4	100	88.76

3.4.4 Further Evaluation on Gene Expression Datasets

Gene expression datasets are often large. Moreover, gene expression datasets usually contain a large number of missing values [30, 31]. Therefore, we evaluate the proposed methods on gene expression datasets to further validate the effectiveness and efficiency of the proposed methods. Table 3.4 shows eight gene expression datasets which were chosen to evaluate the proposed methods.

Table 3.5 shows the classification accuracy and the computation time of the baseline method as shown in Figure 3.4, and the proposed methods when using kNN-based imputation and kNN as a classifier. It is clear from Table 3.5 that the integration of clustering with imputation helps significantly improve the efficiency (around four times faster) while still maintain the accuracy. The integration of feature selection with imputation not only helps improve the efficiency (at least three times faster), but also improve the accuracy (significantly better five in eight cases). Moreover, integrating both clustering and feature selection with imputation can further improve the efficiency (at least 10 times faster) and still improve the accu-

Table 3.5: Classification accuracy and computation time of the baseline method and the proposed methods on gene expression datasets (using kNN as a classifier)

Data	Classification accuracy(%)				Computation time(milisecond)			
	kNNI	**kNNICl**	**kNNIFs**	**kNNIFsCl**	**kNNI**	**kNNICl**	**kNNIFs**	**kNNIFsCl**
alizadeh-2000-v1	77.02±3.04	77.25±2.96	77.44±4.25	77.61±4.24	2.7×10^{-3}	7.3×10^{-4}	7.9×10^{-4}	1.6×10^{-4}
alizadeh-2000-v2	93.77±0.99	93.77±0.99	93.37±1.39	93.43±1.41	9.5×10^{-3}	2.1×10^{-3}	6.5×10^{-4}	1.1×10^{-4}
bredel-2005	83.47±2.54	83.89±2.68	86.77±3.39↑	86.79±3.25↑	4.7×10^{-3}	1.1×10^{-3}	1.3×10^{-3}	3.6×10^{-4}
chen-2002	90.11±0.88	90.10±0.73	92.54±1.48↑	92.48±1.45↑	1.5×10^{-3}	2.1×10^{-4}	3.7×10^{-4}	7.0×10^{-5}
garber-2001	75.83±2.59	75.83±2.59	79.38±3.31↑	79.41±3.78↑	1.9×10^{-2}	4.3×10^{-3}	5.4×10^{-3}	1.0×10^{-3}
liang-2005	95.09±0.89	95.09±0.89	95.09±0.89	95.09±0.89	1.9×10^{-3}	5.7×10^{-4}	2.2×10^{-4}	5.6×10^{-5}
tomlins-2006	73.38±2.37	73.30±2.45	75.87±2.66↑	75.70±2.30↑	7.2×10^{-2}	2.1×10^{-2}	2.1×10^{-2}	5.2×10^{-3}
tomlins-2006-v2	72.01±2.54	72.37±2.69	74.18±2.80↑	74.25±3.12↑	9.8×10^{-3}	2.0×10^{-3}	3.5×10^{-3}	6.7×10^{-4}

racy (significantly better five in eight cases).

In summary, the proposed methods are still able to produce dramatic improvement in efficiency and better accuracy on large datasets.

3.5 Chapter Summary

The goal of this chapter was to propose new methods to improve the effectiveness and efficiency of using imputation for classification with incomplete data. To achieve this goal, we proposed three ways to integrate clustering and feature selection with imputation. The first uses clustering to reduce the number of instances used for imputation in the application process. The second uses feature selection to remove redundant and irrelevant features of the imputed training data which is then used to build a classifier and estimate missing values in the application process. The third

uses both clustering and feature selection. The results showed that the proposed methods can improve the classification accuracy and reduce the computation time of using imputation for classification with incomplete data.

This chapter shows that the integration of clustering with imputation crucially reduces the computation time of the application process with no loss in accuracy. Clustering is used to produce a smaller number of representative instances to perform imputation in the application process. Therefore, the integration speeds up imputation to estimate missing values in the application process while still maintains accuracy.

This chapter shows that the integration of feature selection with imputation also remarkably reduces the computation time, and achieves better accuracy than using only imputation. The first reason is that by removing redundant and irrelevant features, feature selection can enhance the training data which then helps to build better classifiers. The second reason is that feature selection also helps reduce the number of incomplete instances, and provide a smaller number of features to estimate missing values in the application process which then speed up imputation.

This chapter also shows that the integration of both feature selection and clustering with imputation not only further speeds up imputation, but also still improves classification accuracy. Feature selection is used to improve the training data, and reduce the number of features using for imputation in the application process . Clustering is used to further compress data using for imputation in the application process by providing a smaller number of instances. Therefore, the integration of both feature selection and clustering with imputation further reduces the computation time of imputation and still enhances classification accuracy.

The proposed methods in this chapter helps speed up using imputation for classification with incomplete data, but the application process still requires the computation time to estimate missing values. One alternative approach is to use classification algorithms such as C4.5 [129]

which can directly work with incomplete data without the need for imputation. However, these classification algorithms are often not accurate and generate complicated classifiers when datasets contain many missing values. The next chapter will develop a wrapper-based feature selection method to improve the classification accuracy and reduce the complexity of classifiers that are able to directly classify with incomplete data.

Chapter 4

Improve Performance of Classification with Missing Data using Wrapper-based Feature Selection

4.1 Introduction

One common approach to solving classification with incomplete data is to use classification algorithms which can work directly with incomplete data. For example, C4.5 uses a probabilistic approach to tackle missing values in both the training and application processes by making assumption that instances with the missing values are distributed probabilistically according to the relative frequency of known values [129]. The main benefit of this approach is that the classification algorithms do not require any time for estimating missing values. However, when these classification algorithms work with incomplete data, they often generate complex classifiers and result in large classification errors, especially when the incomplete data contains numerous missing values [139]. This chapter shows

how to improve the effectiveness and efficiency of these classification algorithms when faced with incomplete data.

Moreover, classifiers such as decision tress cannot achieve adequate accuracy when the input space contains numerous redundant or irrelevant features. Feature selection that eliminates redundant and irrelevant features and keeps important features while maintaining or improving accuracy is a well known solution to the problem [66, 183]. In feature selection, two main approaches to evaluating feature subsets are the filter approach and the wrapper approach. The filter approach uses measures such as information gain to evaluate the quality of feature subsets [111]. The wrapper approach builds a classifier to evaluate the quality of feature subsets. In recent work [36, 128], filter approaches based on mutual information have been expanded to evaluate feature subsets when datasets contain missing values, and results show that a filter-based feature selection method can help improve classification tasks when faced with missing values. However, a wrapper approach to feature selection for incomplete data has not been investigated. This chapter shows that a wrapper approach to feature selection can improve classification with incomplete data.

Ensemble learning is a machine learning method that builds a set of classifiers instead of a single classifier for classification tasks. Ensemble methods such as bagging and boosting have been demonstrated to enhance classification accuracy [35], but they often generate complex classifiers [119]. Feature selection also has been used to improve ensemble learning [119, 62, 118]. However, feature selection for ensemble learning has been mainly applied to complete data. This chapter shows that the integration of feature selection with ensemble methods can improve classification with incomplete data of the ensemble methods.

4.1.1 Chapter Goals

The overall goal of this chapter is to develop wrapper-based feature selection methods to improve accuracy and reduce complexity of classifiers that are able to work directly with incomplete data. This chapter presents a wrapper-based feature selection method to improve single classifiers for incomplete data. This chapter also presents integration of feature selection with bagging and boosting to improve the ensemble methods for classification with incomplete data. Specially, this chapter will investigates:

1. Whether the proposed wrapper-based feature selection method for a single classifier able to directly classify incomplete data can improve classification accuracy of the classifier.

2. Whether the proposed wrapper-based feature selection method for a single classifier able to directly classify incomplete data can reduce the complexity of the classifier.

3. Whether the integration of feature selection with bagging/boosting can improve classification accuracy of the ensemble methods.

4. Whether the integration of feature selection with bagging/boosting can reduce the complex of classifiers generated by the ensemble methods.

4.1.2 Organisation

The rest of this chapter is organised as follows. Section 4.2 presents the proposed methods. Section 4.3 describes the experiments to evaluate the proposed methods. Section 4.4 presents experimental results and analysis. Finally, section 4.5 states conclusions.

4.2 Proposed Algorithms

This section presents two proposed methods: a wrapper-based feature selection method for a single classifier with incomplete data and a wrapper-based feature selection method for ensemble classifiers with incomplete data.

4.2.1 Wrapper-based Feature Selection for Single Classifier with Incomplete Data

The key idea of the first proposed method is to use a classifier which can work directly with incomplete data to evaluate feature subsets in wrapper-based feature selection. The training data is incomplete, so feature subsets probably contain incomplete features. Therefore, classifiers such as C4.5 which are able to work directly with incomplete features can be used to evaluate the feature subsets.

Figure 4.1 shows the flowchart of the method. The method has two main processes: a training process and an application process. The training process uses a wrapper-based feature method to build an effective and efficient classifier which is then used classify unseen instances in the application process.

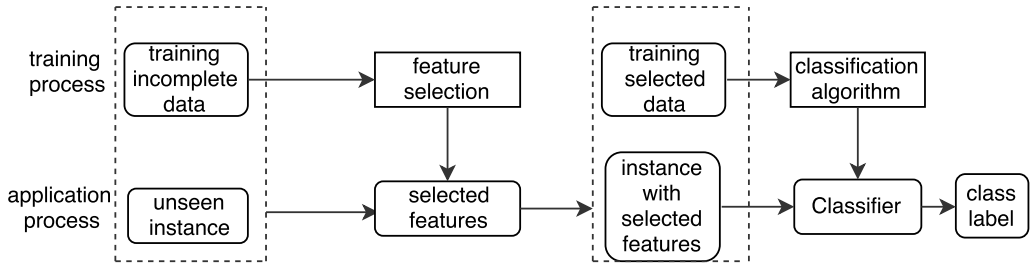


Figure 4.1: Classification with incomplete data using a feature selection method before applying a classifier able to directly classify the incomplete data.

In the training process, firstly, training incomplete data is put into a wrapper-based feature selection method to find important features from original features. In the feature selection method, a search technique is used to search for feature subsets which are evaluated by using a classifier able to work directly with incomplete features. After that, the training data is reduced to the selected features to generate training selected data which is then used by a classification algorithm to build a classifier.

In the application process, an unseen instance is firstly reduced to the selected features generated in the training process. After that, the instance with only selected features is classified by the learnt classifier built in the training process.

4.2.2 Wrapper-based Feature Selection for Ensemble Classifiers with Incomplete Data

The key idea of the second proposed method is to integrate wrapper-based feature selection with bagging or boosting to improve the accuracy and diversity of ensemble classifiers. To construct a set of classifiers, bagging and boosting repeatedly resample the training dataset to build a set of training resampled datasets. The resampled datasets often contain redundant/irrelevant features. Feature selection has been proven capable of remove redundant/irrelevant features. Therefore, feature selection could be applied to each resampled dataset to eliminate redundant/irrelevant features. By eliminating redundant/irrelavant features, this method is expected to improve resampled datasets which in turn can help build more accurate and less complex classifiers.

Figure 4.2 shows the main steps of the training process of the method. In the training process, firstly, the training dataset is put into a resampling procedure several times to generate a set of training resampled datasets. After that, each training resampled dataset is put into a feature selection procedure to select a suitable feature subset which is then used to trans-

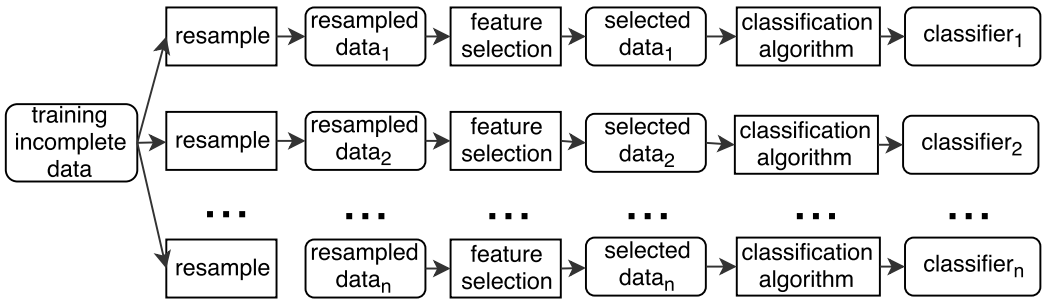


Figure 4.2: The training process of classification with incomplete data by integrating feature selection with ensemble learning methods.

form the training resampled dataset into a training selected dataset. Subsequently, each the training selected dataset is used by a classification algorithm to learn a classifier. As a result, the training process generates a set of classifiers. In the application process, the set of classifiers is combined to classify new instances.

The main steps of the method are presented in the following subsections.

Resampling Data: The purpose of resampling the training dataset is to create a random redistribution of the training dataset. Each training resampled dataset is generated by choosing with replacement the same number of instances as the original training dataset. As a result, many of the original instances might be repeated in the training resampled dataset while others might be left out.

Feature Selection: The key difference between the proposed method and ensemble methods (bagging/boosting) is the feature selection. The ensemble methods use the set of training resampled datasets directly to build a set of classifiers. In contrast, the proposed method applies feature selection to eliminate redundant and irrelevant features in each training resampled dataset before building a set of classifiers.

In order to remove redundant and irrelevant features in incomplete data, any search technique can be used to find feature subsets. Evolutionary techniques such as PSO and GAs (which have been proven to be effective and efficient for feature selection) could be used to search for feature subsets. To evaluate a feature subset which may contain incomplete features, the feature selection procedure requires a feature subset evaluation method which can deal with incomplete data. Therefore, a classifier which is able to directly classify incomplete data such as C4.5 could be used to evaluate feature subsets.

Combining classifiers: A set of classifiers which was built in the training process is combined to classify new instances in the application process. Majority vote which is a simple and powerful voting method [120] chooses a class label with the most votes from the ensemble members as the ensemble output. Therefore, in the proposed method, the majority vote is used to combine classifiers.

4.3 Design of Experiments

This section outlines the design of the experiments. The section then presents comparison methods, datasets and parameter settings.

4.3.1 The Comparison Methods

4.3.1.1 Feature Selection for Single Classifier with Incomplete Data

The first study is designed to empirically evaluate the impact of the wrapper-based feature selection method for a single classifier with incomplete data. To achieve this objective, the proposed method, as shown in Figure 4.1, is compared to two other common methods to tackle classification with incomplete data, as shown in Figure 4.3 and Figure 4.4. Figure 4.3 presents

a common method for classification with incomplete data by using a classifier able to directly classify incomplete datasets. Figure 4.4 presents another common method for classification with incomplete data by using an imputation method to fill missing values with plausible values before using a classifier.

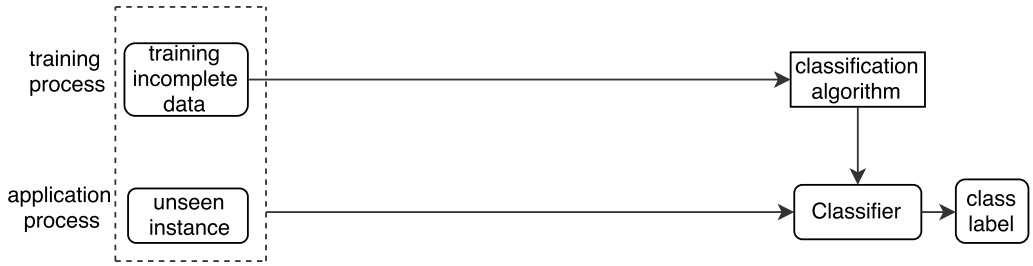


Figure 4.3: Classification with incomplete datasets using a classifier able to directly classify incomplete datasets.

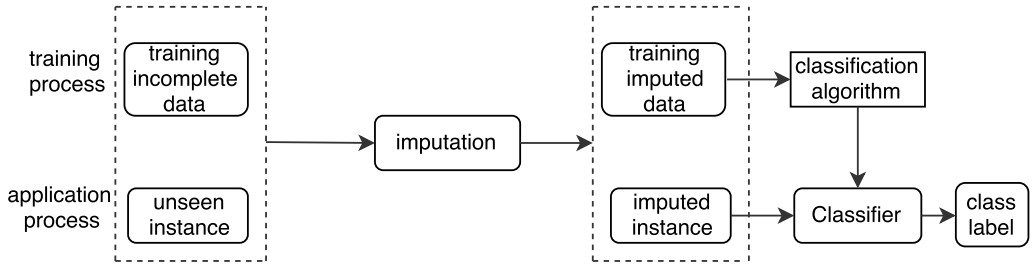


Figure 4.4: Classification with incomplete datasets using an imputation method before using a classifier.

In the proposed setup as shown in Figure 4.1, a feature selection method uses the training incomplete data to select a suitable feature subset. The feature subset is used to transform the training incomplete data into training selected data which is then used by a classification algorithm to build a classifier. To classify a new instance, firstly, the instance is reduced to the selected features, and then classified by the learnt classifier. In the setup shown in Figure 4.3, the training incomplete dataset is directly used by a

classification algorithm to build a classifier which is then used to directly classify new instances. In the setup shown in Figure 4.4, an imputation method is used to transfer the training incomplete data into the training imputed data which is then used by a classification algorithm to build a classifier. To classify a new incomplete instance, firstly, the instance is put into the imputation method to estimate missing values, and then the imputed instance is classified by the learnt classifier.

4.3.1.2 Feature Selection for Ensemble Classifiers with Incomplete Data

Experiments are also conducted to evaluate the impact of the combination of feature selection and ensemble methods for classification with incomplete data. To achieve this goal, the proposed method, as shown in Figure 4.2, is compared with two other common ensemble methods for classification with incomplete data, as shown in Figure 4.5 and Figure 4.6. Figure 4.5 shows the training process of classification with incomplete data by combining an ensemble method with a classifier able to directly classify with incomplete data. Figure 4.6 shows the training process of classification with incomplete data by combining an imputation method and an ensemble method.

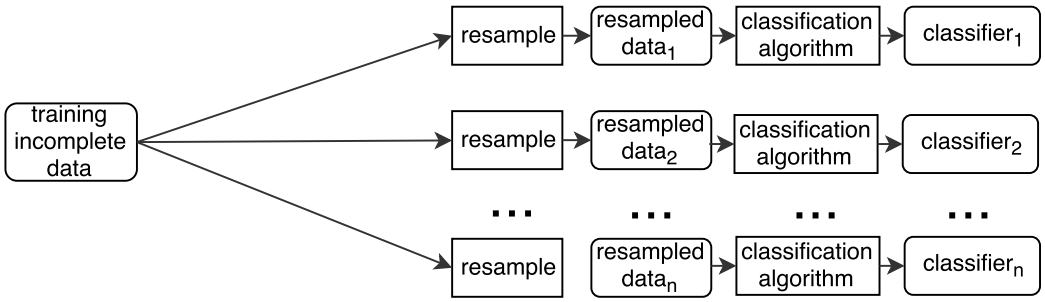


Figure 4.5: The training process of classification with incomplete data by using ensemble learning.

In the proposed method shown in Figure 4.2, the training dataset is used by an ensemble method and feature selection to build a set of classi-

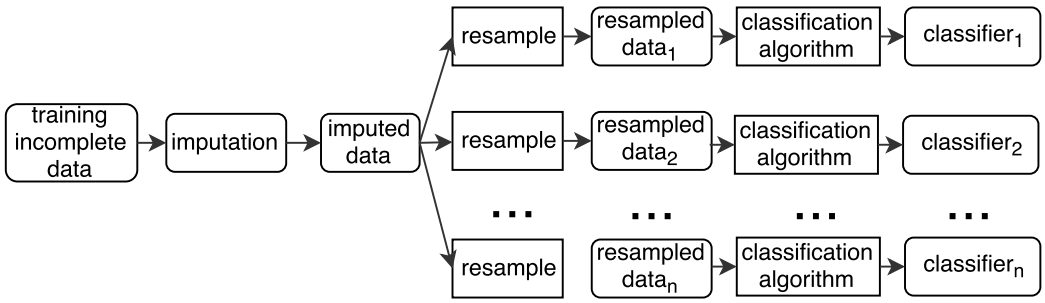


Figure 4.6: The training process of classification with incomplete data by using imputation and ensemble learning.

fiers which is used to classify new instances. In the setup shown in Figure 4.5, the training data is put into a resampling procedure to generate a set of training resampled datasets. After that the set of training resampled dataset is used by a classification algorithm to build a set of classifiers which is then used to classify new instances. In the setup shown Figure 4.6, the training dataset is firstly put into an imputation method to generate a training imputed dataset. After that, the training imputed dataset is put into a resampling procedure to generate a set of training resampled datasets which is used to build a set of classifiers to classify new instances.

4.3.2 Datasets and Parameter Settings

The experiments compare the proposed methods against the benchmark methods on 15 real-world incomplete datasets. Table 1.1 in page 18 shows the main characteristics of these datasets. As in Chapter 3, the ten-folds cross validation procedure is used to divide the datasets into training sets and test sets, and it is performed 30 times to generate 300 pairs of training set and test set for each dataset.

C4.5 is a decision tree which is able to work directly with incomplete datasets. The experiments use *C4.5* to classify data and evaluate feature subsets in feature selection. The experiments use *WEKA* [65] for *C4.5* im-

plementation by setting its parameters as the default values.

Chapter 3 showed that PSO is good at for feature selection. Therefore, PSO is used as a search technique for searching feature subsets in the feature selection method. As in Chapter 3, the parameters of PSO in the feature selection method are chosen according to common settings proposed by Clerc and Kennedy [27]. The detailed settings are shown as follows: ω = 0.729844, c_1 = c_2 = 1.49618, population size is set 50, and the maximum iteration is set 100.

The experiments use two imputation methods: kNN-based imputation and *MICE*. As in Chapter 3, kNN-based imputation is in-house implementations, and a number of neighbors (*k*) is set 1. The implementation of *MICE* using *R* language in [22] is used to run *MICE* where random forest is used as a regression method [175]. The number of cycles is set five and the number of imputed datasets is set 20 following the recommendation in [175].

The experiments use two common ensemble methods: bagging and boosting. In the two ensemble methods, the number of classifiers is set 25 as suggested in [120].

4.4 Results and Discussions

This section presents experimental results and analysis. It first shows the comparison between the first proposed method as shown in Figure 4.1 and the other benchmark methods as shown in Figures 4.3 and 4.4. It then shows the comparison between the second proposed method as shown in Figure 4.2 and the other benchmark methods as shown in Figures 4.5 and 4.6.

4.4.1 Feature Selection for Single Classifier with Incomplete Data

Table 4.1 shows classification accuracy along with standard deviation of the first proposed method and the benchmark methods. "C4.5FS" refers to the first proposed method, as shown in Figure 4.1, using C4.5 as a classification algorithm and an evaluation method in the wrapper-based feature selection. "C4.5" refers to directly classification with incomplete data, as shown in Figures 4.3, by using C4.5 as a classification algorithm. "kNNIC4.5" and "MICEC4.5" refer to the benchmark method, as shown in 4.4, using kNNI and MICE, respectively, for estimating missing values. As in Chapter 3, the *Friedman* test [53] is used to test whether the differences in classification accuracies are statistically significant. The test shows that there exist significant differences between the methods for all datasets; therefore, the *Holm* procedure [72] is used to carry out pairwise comparisons. The symbol $\uparrow$ indicates that the benchmark method is significantly more accurate than the proposed method. In contrast, the symbol $\downarrow$ indicates that the benchmark method is significantly less accurate than the proposed method.

It is clear from Table 4.1 that the proposed method generally achieves comparable, or better accuracy than the benchmark methods. In 15 datasets, C4.5FS is significantly more accurate than C4.5 and kNNIC4.5 in five and six datasets, respectively, and it is only significant less accurate than the two benchmark methods in two datasets. Unsurprisingly, the proposed method is very comparable to MICEC4.5, with roughly equal numbers of datasets better or worse.

Table 4.2 shows the average size of decision trees (the number of nodes in the trees) generated by C4.5FS and the other benchmark methods. Figure 4.7 presents the minimum, average and maximum ratios between the average of tree sizes generated by C4.5, kNNIC4.5 and MICEC4.5 with C4.5FS from Table 4.2.

Table 4.1: The average of accuracy comparison between C4.5FS and the other methods.

Dataset	C4.5FS	C4.5	kNNIC4.5	MICEC4.5
Arr	**66.02±2.25**	65.64±2.13	65.30±2.44	65.43±2.02
Aut	65.98±3.92	67.49±3.85↑	**68.37±3.50**↑	68.09±4.17↑
Ban	68.16±1.88	68.45±1.88	66.05±1.91↓	**71.43±1.80**↑
Bre	94.69±0.50	**94.83±0.46**	94.62±0.50	94.84±0.52
Chr	97.67±0.45	99.08±0.30↑	**99.40±0.47**↑	97.53±0.62
Cle	**56.85±1.38**	54.56±2.10↓	54.16±2.02↓	54.09±1.98↓
Cre	85.14±0.62	85.04±0.66	85.30±0.59	**85.36±0.58**
Hea	**80.46±1.18**	79.38±1.23↓	79.10±1.19↓	78.25±1.39↓
Hep	78.72±1.58	79.21±1.75	78.55±2.05	**80.01±2.25**↑
Hor	**84.26±0.63**	84.24±0.48	82.94±1.22↓	84.06±1.21
Hou	96.35±0.54	**96.52±0.34**	96.31±0.52	96.15±0.54
Mam	**82.39±0.53**	82.12±0.33↓	81.81±0.57↓	82.24±0.65
Mar	**32.76±0.39**	30.91±0.45↓	30.02±0.56↓	30.01±0.41↓
Ozo	**96.88±0.26**	96.25±0.27↓	95.67±0.88↓	95.90±0.40↓
Tum	40.39±2.02	39.67±2.12	40.49±2.64	**41.24±2.05**

It is clear from Table 4.2 and Figure 4.7 that C4.5FS is able to generate smaller decision trees than the other methods in all cases. On averaged on all datasets, C4.5FS generates trees about half the size of the other methods. In some datasets, for example, *Marketing* and *Ozone*, the average tree size generated by C4.5FS is less than one quarter of the average tree size generated by C4.5 and less than one fifth of the average tree size generated by using imputation methods before using C4.5. A possible reason that imputation methods generates particularly large tree is that imputation methods often generate further values for missing features. Therefore, if the incomplete features are selected to make decision trees, the further values can make decision trees bigger.

Note, C4.5 does implicit feature selection when it builds a tree. The

Table 4.2: The average tree sizes of C4.5FS and the other benchmark methods.

Dataset	C4.5FS	C4.5	kNNC4.5	MICEC4.5
Arr	45.86	48.03	48.40	48.60
Aut	35.28	45.90	41.60	39.46
Ban	45.63	85.10	99.02	95.44
Bre	15.75	23.20	23.29	23.47
Chr	13.04	13.73	14.72	14.02
Cle	26.68	78.50	80.74	80.90
Cre	26.44	29.47	31.31	31.67
Hea	7.62	11.37	23.60	26.41
Hep	8.97	17.04	18.98	17.76
Hor	9.56	9.82	18.94	19.83
Hou	9.63	10.64	10.98	10.66
Mam	9.57	10.46	14.43	17.02
Mar	309.01	1361.02	1645.40	1679.65
Ozo	5.50	24.30	28.93	28.66
Tum	29.11	54.56	58.15	60.48

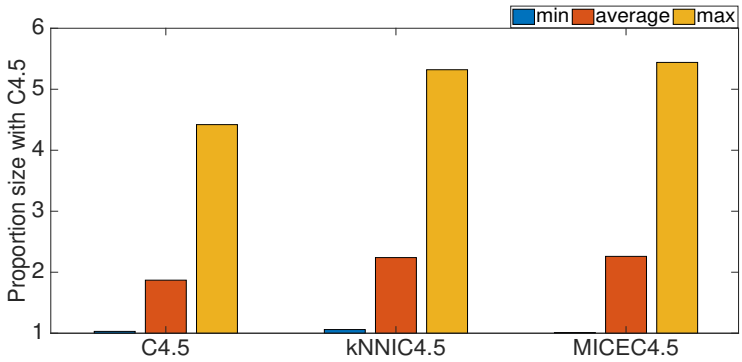


Figure 4.7: The tree sizes ratio between C4.5, kNNIC4.5 and MICEC4.5 over C4.5FS.

results above show that doing explicit feature selection achieves better results than relying on C4.5 built-in feature selection.

In summary, the proposed method using wrapper-based feature selection reduces the complexity of the learnt classifier, and achieves comparable, or even better accuracy than the benchmark methods.

4.4.2 Feature Selection for Ensemble Classifiers with Incomplete Data

Table 4.3 shows the average of classification accuracy along with standard deviation of the second proposed method and the benchmark methods by using bagging and boosting methods to build ensembles of classifiers. Table 4.3 is divided into two vertical partitions. The first partition contains accuracy information for the proposed and benchmark methods where bagging is used to build ensemble classifiers. The second partition contains accuracy information for the proposed and benchmark methods where boosting is used to build ensemble classifiers. The *Friedman* test and *Horm* procedure are used to statistically test the significant differences between the proposed method and the benchmark methods.

It is clear from Table 4.3 that when bagging is used to build ensembles of classifiers, the proposed method generally achieves comparable, or better accuracy than the benchmark methods. BagC4.5FS is significantly more accurate than BagC4.5, BagkNNIC4.5 and BagMICEC4.5 in four, eight and six datasets, respectively. BagC4.5FS is significantly less accurate than BagkNNIC4.5 and BagMICEC4.5 in one and four datasets, respectively, and it is not worse than BagC4.5 in any dataset.

Table 4.3 also shows that when boosting is used to build ensembles of classifiers, the proposed method also achieves comparable, or better accuracy than the benchmark methods. The accuracy of BooC4.5FS is significantly better than BooC4.5, BookNNIC4.5 and BooMICEC4.5 in five, six and two datasets, respectively. The accuracy of BooC4.5FS is significantly

Table 4.3: The average of accuracy comparison between BagC4.5FS, BooC4.5FS and the other benchmark methods.

Data	Bagging				Boosting			
	BagC4.5FS	**BagC4.5**	**BagkNNIC4.5**	**BagMICEC4.5**	**BooC4.5FS**	**BooC4.5**	**BookNNIC4.5**	**BooMICEC4.5**
Arr	**73.01±1.05**	72.85±1.35	72.89±1.32	72.60±1.34	71.83±1.47	**72.02±1.57**	71.36±1.54	72.01±1.49
Aut	**70.44±3.47**	70.07±2.84	70.07±3.49	70.22±3.51	**73.57±3.69**	72.68±3.18	73.29±3.29	73.56±3.54
Ban	71.72±2.17	71.05±1.12↓	68.76±1.29	**73.69±1.67**↑	70.07±1.46	69.22±1.71↓	68.10±2.10↓	**72.08±1.48**↑
Bre	**96.10±0.35**	95.82±0.38↓	95.77±0.36↓	95.92±0.43	**95.74±0.41**	95.68±0.49	95.49±0.38↓	95.65±0.43
Chr	99.18±0.39	99.08±0.45	**99.21±0.24**	97.45±0.63↓	99.41±0.31	**99.76±0.29**↑	99.61±0.27↑	98.49±0.49
Cle	**58.20±1.52**	57.06±1.66↓	57.09±1.17↓	57.13±1.48↓	**56.79±1.21**	55.88±1.46↓	56.16±1.21↓	56.15±1.73
Cre	**86.30±0.52**	86.05±0.60	85.87±0.70↓	86.13±0.63	**85.28±1.02**	84.22±0.59↓	84.24±0.94↓	84.41±0.72↓
Hea	**80.11±1.59**	79.93±1.09	79.62±1.36	79.34±1.34	78.88±1.55	79.40±1.44	**79.84±1.53**↑	79.15±1.46
Hep	82.01±1.57	81.30±1.39	80.80±1.82↓	**83.78±1.68**↑	82.28±1.64	82.66±1.27	81.92±1.96	**84.21±1.70**↑
Hor	85.09±0.68	**85.34±0.48**	83.95±0.84↓	84.69±0.68↓	**82.19±1.26**	81.99±1.27	82.12±1.09	81.93±1.27
Hou	95.75±0.31	95.97±0.43	95.87±0.53	**96.20±0.65**↑	**95.08±0.51**	94.75±0.64↓	94.78±0.58↓	95.02±0.55
Mam	**82.91±0.47**	82.70±0.50	82.23±0.56↓	82.37±0.62↓	77.58±0.89	77.51±0.79	77.53±0.96	**78.00±0.94**
Mar	**31.64±0.41**	31.37±0.36↓	30.89±0.33↓	31.10±0.29↓	**29.90±0.35**	29.86±0.35	29.52±0.40↓	29.69±0.42↓
Ozo	**97.08±0.09**	97.05±0.07	97.00±0.17↓	97.00±0.10↓	**97.06±0.12**	97.05±0.11	97.00±0.16	97.05±0.11
Tum	41.00±1.95	41.27±1.69	42.16±2.48↑	**42.61±2.05**↑	38.91±2.02	36.83±2.42↓	39.56±2.21	**40.32±1.82**↑

worse than BookNNIC4.5 and BooMICEC4.5 in two datasets, and it is significantly worse than C4.5 only in one dataset.

Table 4.4 shows the average size of decision trees(the number of nodes in the trees) of the second proposed method and the other benchmark methods. Figure 4.8 summaries Table 4.4 by showing the average of tree size ratio between the other methods and the proposed method with bagging and boosting (bigger than one means bigger tree, otherwise equal or smaller tree). It is clear from Figure 4.8 that the proposed method also generates smaller decision trees the other benchmark methods.

In summary, the combination of feature selection and bagging/boosting not only improves classification accuracy of the ensemble methods, but also reduces the complexity of ensemble classifiers.

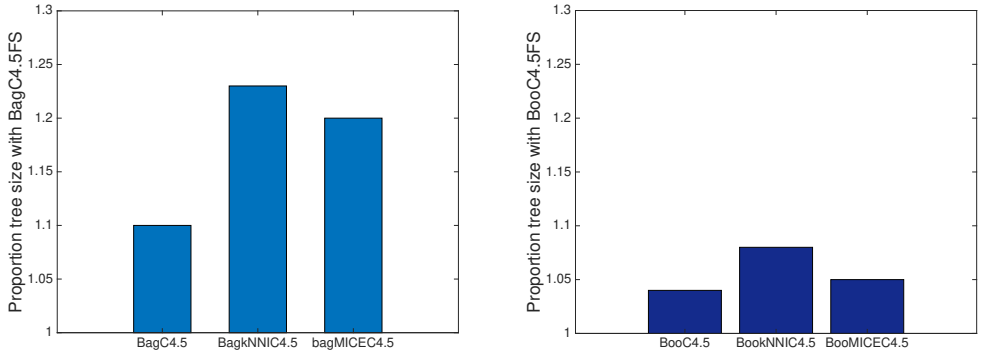
Table 4.4: The average of tree size comparison between BagC4.5FS, BooC4.5Fs and the other methods.

Data	Bagging				Boosting			
	BagC4.5FS	**BagC4.5**	**BagkNNIC4.5**	**BagMICEC4.5**	**BooC4.5FS**	**BooC4.5**	**BookNNIC4.5**	**BooMICEC4.5**
Arr	41.67	42.69	42.70	43.44	48.96	49.78	50.05	50.10
Aut	35.09	38.37	34.30	34.52	35.30	39.83	39.72	39.08
Ban	80.15	83.39	83.72	85.76	79.56	81.15	80.76	81.28
Bre	19.47	22.92	23.29	23.23	39.61	39.02	39.70	39.83
Chr	12.84	13.57	13.93	11.57	15.71	16.35	16.17	16.63
Cle	69.93	73.41	74.35	73.50	79.37	79.29	79.55	78.97
Cre	48.61	48.44	49.13	49.47	92.15	100.0	102.7	101.2
Hea	29.63	28.70	38.50	38.93	43.51	48.97	47.08	46.84
Hep	12.44	15.16	16.92	14.72	19.22	20.66	20.57	18.73
Hor	23.69	21.60	36.46	36.63	46.13	36.54	51.80	48.48
Hou	9.27	11.70	13.01	12.71	26.03	27.08	31.38	27.97
Mam	23.29	32.03	41.52	40.16	75.06	78.86	86.32	81.72
Mar	1691	1710	1873	1874	1689	1703	1813	1821
Ozo	20.05	23.01	25.68	26.68	47.38	48.77	49.28	49.35
Tum	57.98	70.22	70.99	71.77	63.60	72.03	71.86	74.84

4.4.3 Further Analysis

To understand how the integration of feature selection with ensemble methods achieves better accuracy and smaller trees than the ensemble methods with all features, we looked carefully at the trees generated by C4.5 in two cases: bagging using all features and bagging combined with selected feature on *Hepatitis* which has 19 features (*Age*, *Sex*, *Steroid*, *Antivirals*, *Fatigue*, *Malaise*, *Anorexia*, *LiverBig*, *LiverFirm*, *SpleenPalpable*, *Spiders*, *Ascites*, *Varices*, *Bilirubin*, *AlkPhosphate*, *Sgot*, *AlbuMin*, *ProTime*, *Histology*). The *Hepatitis* is chosen since the trees generated on this dataset are not too big to analyse. Figures 4.9 and 4.10 present two typical pattern trees we observed.

It is clear from Figures 4.9 and 4.10 that the combination of bagging and



(a) Bagging (b) Boosing

Figure 4.8: The ratios of tree size between the other benchmark methods and BagC4.5FS, BooC4.5Fs.

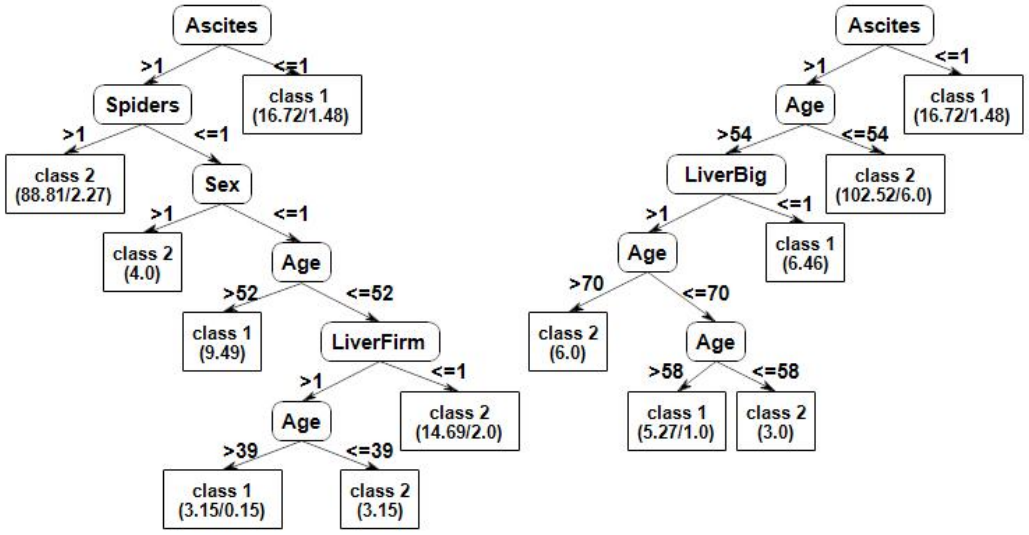


Figure 4.9: Left tree with 90.0% of accuracy generated by C4.5 bagging with all features and right tree with 92.14% of accuracy generated by C4.5 bagging with selected features.

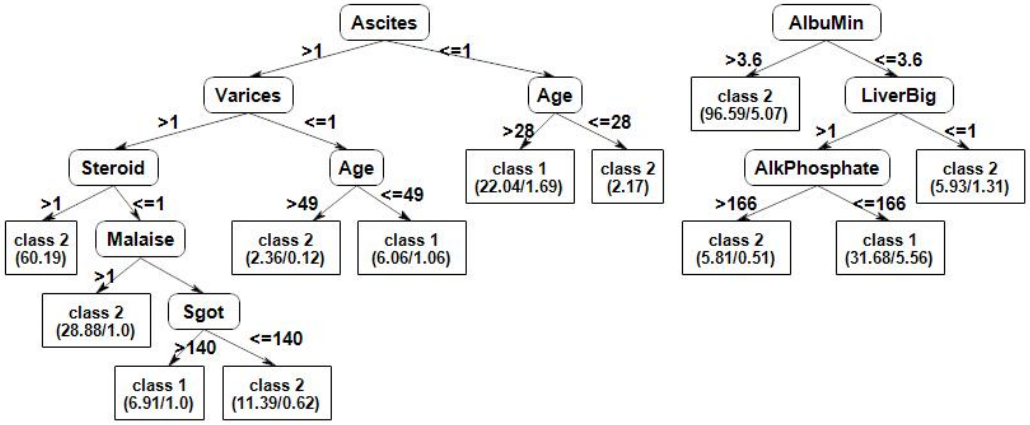


Figure 4.10: Left tree with 86.42 of accuracy generated by C4.5 bagging with all features and right tree with 91.42% of accuracy generated by C4.5 bagging with selected features.

feature selection can generate more accurate and less complex trees than bagging with all features. The reason might be that classifiers like C4.5 are greedy algorithms which make the locally optimal choice at each stage. Therefore, they may provide locally optimal solutions. The purpose of feature selection is to search for more suitable feature subsets. Therefore, feature selection can help reduce the limitation of greedy algorithms. For example, in Figure 4.9, with a training resampled data, when C4.5 bagging uses all features, the information gain of feature Spiders and feature Sex are higher than the information gain of feature Age, so feature Spiders and feature Sex are chosen to build the right tree before choosing feature Age. When feature selection is applied to the training resampled data, only three features Age, Ascite and LiverBig are selected. Consequently, feature Age is chosen to develop the right tree instead of feature Spiders or feature Sex. As a result, bagging with selected features generates more accurate and less complex trees than bagging with all features.

It also can be seen from Figures.4.9 and 4.10 that the combination of bagging and feature selection can generate more diverse trees than bag-

ging with all features. For example, the right tree on Figure 4.9 uses the same feature Ascites in the *first level* as the right tree on Figure 4.10. However, the left tree on Figure 4.9 uses different feature in the *first level* from the right tree on Figure 4.10. By generating more diverse trees, feature selection helps improve the bagging method.

In summary, bagging with selected features can generate more accurate, less complex and more diverse learnt models than bagging with all features. Therefore, the combination of bagging and feature selection helps improve the traditional bagging method.

4.5 Chapter Summary

This chapter attempted to explore the impact of a wrapper feature-based feature selection method for classification with incomplete datasets. To achieve this goal, a wrapper-based feature selection method for incomplete datasets was proposed and compared with the two other common methods coping with incomplete datasets: one using a classifier able to directly classify incomplete datasets and the other using an imputation method to transfer incomplete datasets to complete datasets. The experiments used C4.5 as an evaluation and PSO as a search method for the feature selection approach. The experimental results showed that the proposed wrapper-based feature selection method for incomplete datasets is able to help to enhance the classification accuracy of C4.5 and significantly reduce the complexity of the learned classifier.

This paper also proposed a combination of bagging/boosting and feature selection method to improve classification with incomplete data. In the proposed method, bagging/boosting is firstly used to construct a set of training resampled data. After that, the set of training resampled data is used by a wrapper-based feature selection method to build a set of training selected data which is then used to learn a set of classifiers. The proposed method is compared with the other ensemble methods. The results

showed that the combination of bagging/boosting and feature selection method is more accurate than the other methods. Moreover, the combination of bagging/boosting and feature selection is able to reduce the complex classifiers generated by bagging/boosting.

The proposed method in this chapter can improve the classification accuracy and reduce the computation time of classifiers able to directly classify by removing redundant and irrelevant features. However, many classification algorithms such as decision trees cannot achieve adequate accuracy when faced with difficult tasks because they are not good at transforming their input to gain class separability. Feature construction is a traditional approach to this problem [116]. The next chapter will develop a new feature construction method for classification with incomplete data.

Chapter 5

Genetic Programming-based Feature Construction for Classification with Incomplete Data

5.1 Introduction

Decision trees such as C4.5 and CART can directly work with incomplete data without requiring any imputation method. However, decision trees often cannot work well when faced with difficult classification tasks. One reason is that these classifiers are not able to capture the interaction between features. Feature construction which constructs better new features from original features is a conventional solution to this problem. GP can learn the definition of a function itself from example data, so GP is an excellent choice for feature construction, and has been widely applied [42, 158, 161].

Although there are many GP-based feature construction methods, most of them construct only a single feature, which needs to be combined with

the original features, so will lead to a higher dimensionality [42]. GP-based multiple feature construction (GPMFC) [116] is a recent promising filter approach using GP for feature construction. GPMFC is able to evolve multiple high-level features from the original features. The empirical results show that, in almost all cases, GPMFC can not only improve the classification accuracy, but can also reduce the complexity of decision trees and rule-based classifiers.

However, GPMFC is not able to deal with incomplete data. As a result, using GPMFC for incomplete data requires imputation methods to transform incomplete data into complete data before using GPMFC [158]. In order to obtain good performance, GPMFC has to be combined with sophisticated imputation methods such as MICE [175]. Unfortunately, sophisticated imputation methods such as MICE are often suitable for batch imputation, but computationally intensive for imputing missing values in a single instance in the unseen set for classification [157]. This chapter presents an extended version of GPMFC which can directly construct multiple features from incomplete data without using any imputation methods.

5.1.1 Chapter Goals

This chapter presents a new method that uses GP to directly construct multiple features for classification with incomplete data without using any imputation method. To achieve this goal, we develop an extension of GPMFC, called IGPMFC, that uses GP with a set of interval functions to directly construct multiple features. Specially, this chapter will investigate:

1. How GP can directly perform feature construction for classification with incomplete data; and

2. Whether IGPMFC can improve classification accuracy and reduce

the complexity of classifiers with incomplete data compared to using original features; and

3. Whether IGPMFC can improve classification accuracy and reduce the complexity of classifiers with incomplete data compared to combining imputation and GPMFC.

4. Why GP can effectively and efficiently perform feature construction for classification with incomplete data.

5.1.2 Organisation

The rest of this chapter is organised as follows. Section 5.2 states key ideas of GPMFC. Section 5.3 presents the IGPMFC algorithm which directly performs multiple feature construction for classification with incomplete data using GP with a set of interval functions. Section 5.4 outlines experiment design. Section 5.5 presents empirical results and analysis. Section 5.6 draws conclusions.

5.2 GP-based Multiple Feature Construction

GP-based multiple feature construction (GPMFC) is a filter approach to feature construction that uses GP for constructing multiple features [116]. GPMFC uses GP to evolve new features, and uses the purity of class intervals as a measure to evaluate new features.

Algorithm 4 shows the main body of the GPMFC algorithm. The input data for GPMFC includes two parts. The first part is a matrix containing values of original features. The second part is an array containing class labels corresponding observations in the matrix. For each class label, GPMFC evolves the best constructed feature that maximise the purity of the corresponding class interval. As a result, the output of GPMFC is a set

of constructed features equal to the number of classes in the problem. The detail of GPMFC can be seen in [116].

Algorithm 4: GPMFC [116]

Input:
D: a matrix containing values of original features
C: an array containing class values corresponding observations in D
Output: CF– a set of constructed features

```
1  CF ← {}
2  for c ∈ C do
3      P ← InitialPopulation
4      bestFitness ← +∞
5      while ¬maxGenerations ∧ bestFitness ≠ 0 do
6          for φ ∈ P do
7              φ_fitness ← Fitness(D, φ, c)
8              if φ_fitness < bestFitness then
9                  bestProgram ← φ
10                 bestFitness ← φ_fitness
11             end
12         end
13         Perform selection
14         Perform genetic operators
15     end
16     CF ⟵ CF ∪ {bestProgram}
17 end
18 return CF
```

To apply GPMFC, GPMFC firstly uses the training data to build a set of constructed features that then forms a transformation. The training data is then put into the transformation to generate transformed training data. After that, a classification algorithm uses the transformed training data to build a classifier that is then used to classify the unseen transformed data.

The experimental results show that in almost all cases, GPMFC greatly improves classification accuracy of decision trees and rule-based classifiers. Furthermore, GPMFC helps to reduce the complexity of the learnt

classifiers. However, GPMFC is not able to directly deal with incomplete data. Therefore, the ability of GPMFC to directly deal with incomplete data should be investigated.

5.3 GP with Interval Functions for Multiple Feature Construction on Incomplete Data

Although GPMFC is a powerful feature construction method for complete data, it cannot work directly with datasets containing missing values. To tackle this problem, we proposes IGPMFC which is an extension of GPMFC. IGPMFC uses GP with a set of interval functions—interval GP—to directly evolve multiple new features from incomplete data. The underlying idea of IGPMFC is that interval functions enables GP to operates directly on missing values. If a value for a feature is missing, it is substituted by an interval associated with the feature. If a value for a feature is complete, it is substituted by an interval spanning just the value—the lower bound and upper bound are both equal to the value. The purpose of using interval functions is that missing values are unknown; hence replacing a missing value with an interval reflects the uncertainty associated with the missing value better than using a single value.

The interval associated with each feature must represent the range of possible values of the feature and needs to be estimated by the algorithm. Furthermore, the interval function set of GP also needs to be defined.

5.3.1 Finding the Interval of a Feature

A feature interval is the range which covers a large majority of the actual values of the feature. The interval of a feature should be estimated from the distribution of the feature values. A simple way to find the interval of a feature is to consider that the range between the minimum and maximum

of the feature is the interval of the feature. However, this kind of interval may contain outliers which are not desired [116].

If the values of feature f are normally distributed, the interval $(\mu_f - 3\sigma_f, \mu_f + 3\sigma_f)$ covers 99% of the feature values, where μ_f and σ_f are the mean and the standard deviation of the feature *f*, respectively [116]. Unfortunately, the values of a feature are not necessarily normally distributed; therefore, the interval might include too many values, or too few values. The introselect algorithm in [115] is an advanced method that could be used. We used the method described in [116] which removes a fixed fraction of values from both the top and bottom of the range. Experiments in [116] showed that this worked well for a range of distributions.

5.3.2 Interval Functions

Assuming the interval of feature *a* is represented by the range between the lower bound a_l and the upper bound a_u, and the interval of feature *b* is represented by the range between the lower bound b_l and the upper bound b_u. In IGPMFC, the function set of GP uses four interval arithmetic operations defined as follows [68]:

$$a + b = \begin{cases} lower: & a_l + b_l \\ upper: & a_u + b_u \end{cases}$$

$$a - b = \begin{cases} lower: & a_l - b_u \\ upper: & a_u - b_l \end{cases}$$

$$a * b = \begin{cases} lower: & min(a_l * b_l, a_l * b_u, a_u * b_l, a_u * b_u) \\ upper: & max(a_l * b_l, a_l * b_u, a_u * b_l, a_u * b_u) \end{cases}$$

$$a/b = \begin{cases} lower: & min(a_l/b_l, a_l/b_u, a_u/b_l, a_u/b_u) \\ upper: & max(a_l/b_l, a_l/b_u, a_u/b_l, a_u/b_u) \end{cases}$$

It is important to notice that the division operation is not correct if the lower and upper bounds of denominator have different signs. Therefore, it requires an assumption that the denominator lower bound has the same sign with the denominator upper bound. We expect GP search to eliminate trees which break the assumption.

5.3.3 Estimating the Real Output of an Individual

The output of an individual evolved by GP with interval functions is an interval. Nevertheless, to use constructed features for classification, single values are required. Hence, in IGPMFC, the real output of an individual is calculated as the middle point of the final computed interval. Assuming that $[out_l, out_u]$ is the output of an individual, the real output can be defined as follows:

$$out = \frac{out_l + out_u}{2}$$

5.4 Experiment Design

This section shows detailed experiment design including the comparison methods, datasets, the imputation methods used in the experiments, GP settings and classification algorithms.

5.4.1 Comparison Method

The experiments are designed to evaluate the impact of IGPMFC to construct new features for classification with incomplete data. To achieve this, three experimental setups are designed, as shown in Figure 5.1, Figure 5.2 and Figure 5.3. The Figure 5.1 shows classification with incomplete data by using IGPMFC to construct new features from incomplete data before using a classifier. The Figure 5.2 shows classification with incomplete data by using a classifier such as C4.5 or CART that is able to directly classify

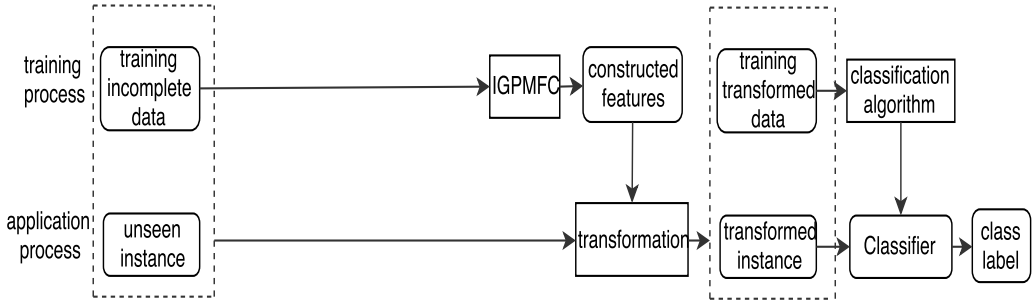


Figure 5.1: Classification with incomplete data by using IGPMFC before using a classifier.

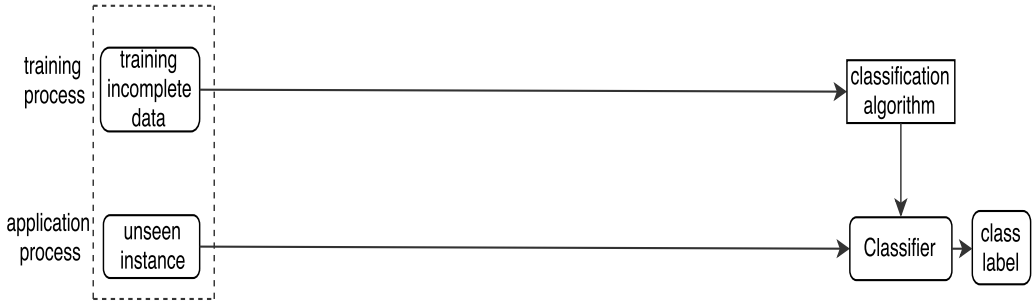


Figure 5.2: Classification with incomplete data by using a classifier able to classify incomplete data.

incomplete data. The Figure 5.3 shows classification with incomplete data by using an imputation method to transform incomplete data into complete data that is then used by GPMFC to construct new features before using a classifier.

In the first setup, as shown in Figure 5.1, IGPMFC directly uses training incomplete data to construct new features that is then used to build a data transformation. The data transformation is used to transform the training incomplete data and unseen incomplete instances into training transformed data and unseen transformed instances, respectively. The training transformed data is used by a classification algorithm to build a classifier which is then used to classify the unseen transformed instances. In the sec-

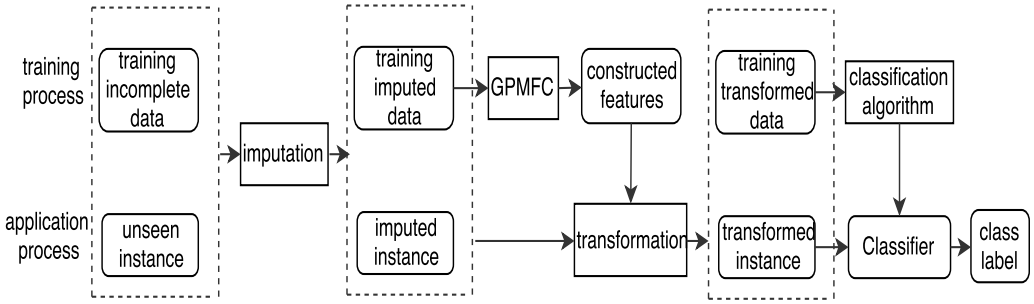


Figure 5.3: Classification with incomplete data by using an imputation method and GPMFC before using a classifier.

ond setup, as shown in Figure 5.2, the training incomplete data is directly put into a classification algorithm to build a classifier that is then used to classify unseen incomplete instances. In the third setup, as shown in Figure 5.3, both training incomplete data and unseen incomplete instances are put into an imputation method to generate training imputed data and unseen imputed instances, and then, the training imputed data is put into GPMFC to build a data transformation. The data transformation is then used to transform the training imputed data and the unseen imputed instances into training transformed data and unseen transformed instances, respectively. After that, as in the first setup, the training transformed data is then put into a classification algorithm to build a classifier that is then used to classify the unseen transformed instances.

5.4.2 Datasets and Parameter Settings

In order to examine the performance of the proposed method, a set of experiments have been conducted on twelve datasets. These datasets only have numerical features because constructed features are mathematical functions of the original features. Table 1.1 in page 18 shows the main characteristics of these datasets. Seven of these datasets contain "natural" missing values, and the other five datasets contain "artificial" miss-

ing values by removing complete values and introducing missing values. To evaluate more precisely the proposed methods on incomplete datasets, missing values are introduced into important features. The correlation-based feature selection method (CFS) [66] is used to select important features. For each of the dataset, we use CFS to select important features, and then randomly introduce 20% of missing values into the selected features. The same as the previous chapters, these datasets are divided into training sets and test sets by using ten-fold cross-validation.

Experiments use two imputation methods which are kNN-based imputation (a single imputation) and MICE (a multiple imputation). As in the previous chapters, for kNN-based imputation, the number of neighbour (k) is set 1. MICE's implementation in [22] with random forest for regression is used for multiple imputation. The number of cycles is set five and the number of imputed datasets is set 20 following the recommendation in [175].

Experiments use the ECJ package [110] to implement GP. The parameters of GP are the same in all experiments and are shown in Table 5.1. For each pair of training set and test set, GPMFC combined with kNN-based imputation, GPMFC combined with MICE and IGPMFC run a number of times, each constructing a new feature for a particular class.

GPMFC is designed to improve symbolic learning classifiers such as decision trees. Therefore, experiments use four decision trees to classify data: C4.5 [129], CART [21], REPTree [17] and BFTree [145]. The classification algorithms can directly work with incomplete data. For all the classifiers, WEKA's implementation is used and all parameters set to WEKA's defaults [65].

5.5 Results and Discussions

This section presents the comparison of the proposed method (IGPMFC) as shown in Figure 5.1 with the *Baseline* as shown in Figure 5.2 and GPMFC

Table 5.1: GP parameters.

Parameter	Value
Function set	Interval functions, +, -, x, / (protected division)
Variable terminals	Interval of the original features $\setminus\{f_1, f_2, .., f_n\}$
Constant terminals	Random float values
Population size	1024
Initialization	Ramped half-and-half
Generations	50
Crossover probability	60
Mutation probability	30
Reproduction rate	10
Selection type	Tournament(size=7)

combined with imputation as shown in Figure 5.3.

5.5.1 Effect of Constructed Features on Classification Accuracy

Table 5.2 presents the average of classification accuracy along with standard deviation of the proposed method and benchmark methods. The averages are calculated on 30 times performing ten-fold cross-validation on each dataset. "IGPMFC " column indicates results from the first experimental setup as shown in Figure 5.1; "*Baseline*" column indicates results from the second experimental setup as shown in Figure 5.2; "kNNGPMFC" and "MICEGPMFC" columns indicate results from the third experimental setup as shown in Figure 5.3 by using kNN-based imputation and MICE combined with using GPMFC, respectively. As in previous Chapters, in order to compare the classification accuracy of IGPMFC with the other methods, the Friedman test [33] is used to test the significance of the results. The test shows that there exits significant differences between

the methods in each dataset and each classifier. Therefore, the Holm procedure [33] is used to perform pair tests between the proposed method and the other methods. In Table 5.2, the symbol $\uparrow$ indicates that the benchmark method is significantly better than IGPMFC. In contrast, the symbol $\downarrow$ shows that the benchmark method is significantly worse than IGPMFC.

Figure 5.4 summarises the accuracy comparison of IGPMFC with *Baseline*, kNNGPMFC and MiceGPMFC. It is clear from Figure 5.4 that IGPMFC achieves significantly better classification accuracy than *Baseline* in over 50% of cases, and IGPMFC is only significantly worse than *Baseline* in about 5% of cases. The underlying reason is that IGPMFC constructs a new feature for each class label which then helps the decision trees separate classes better [116]. The classification accuracy improvement is different among datasets. For example, on the *Balance Scale* dataset, the improvement is much higher than in the other datasets. Moreover, the classification accuracy improvement in each classifier is also different on different datasets.

It is also clear from Figure 5.4 that IGPMFC is significantly more accurate than kNNGPMFC in over half of cases. Moreover, IGPMFC is not significantly worse than kNNGPMFC in any cases. The key reason is that IGPMFC replaces missing values by intervals which can reflect better the uncertainty of missing values than specific imputed values generated by kNN-based imputation.

Figure 5.4 also shows that IGPMFC can achieve comparable accuracy to MICEGPMFC. In 48 cases, IGPMFC achieves significantly better classification accuracy than MICEGPMFC in 7 cases, similar accuracy in 35 cases and significantly worse in 6 cases. The reason is that by replacing missing values by intervals, IGPMFC can reflect the uncertainty of missing values as well as MICEGPMFC which also estimates a set of values for each missing value.

In summary, in almost all cases, IGPMFC not only can achieve better classification accuracy compared to using original features, but also can

Table 5.2: Average of Classification Accuracy on the Test Set

Dataset	Classifier	IGPMFC	*Baseline*	kNNGPMFC	MICEGPMFC
Bands	C4.5	68.66±1.17	68.45±1.88	59.46±1.36↓	64.86±1.99↓
	CART	68.69±1.35	65.99±1.69↓	60.00±1.35↓	67.72±1.48↓
	REPTree	68.10±1.45	65.82±1.71↓	59.31±2.12↓	66.92±1.50↓
	BFTree	67.47±1.79	66.66±1.94	59.41±1.41	64.10±2.05
Breast Cancer	C4.5	95.78±0.59	94.83±0.46↓	95.46±0.71↓	95.71±0.55
	CART	96.19±0.69	94.42±0.44↓	95.94±0.56↓	96.14±0.44
	REPTree	96.11±0.51	94.35±0.65↓	95.83±0.55	96.08±0.44
	BFTree	95.80±0.90	94.49±0.54↓	95.27±0.99↓	95.46±0.95
Cleveland	C4.5	56.77±1.64	54.56±2.10↓	56.93±1.76	57.06±2.10
	CART	58.39±1.46	56.32±1.37↓	58.03±1.44	58.19±1.54
	REPTree	58.04±1.31	56.55±1.36↓	57.62±1.83	57.69±1.62
	BFTree	57.68±1.64	54.88±1.93↓	57.57±1.62	57.29±1.76
Hepatitis	C4.5	80.44±2.59	79.21±1.75↓	79.93±1.85	80.30±1.71
	CART	80.70±2.60	77.47±1.45↓	80.93±2.45	80.71±1.99
	REPTree	80.81±2.51	79.32±2.17↓	80.40±1.67	80.55±1.63
	BFTree	79.56±2.34	78.55±1.80	79.48±2.67	79.85±2.35
Mammographic	C4.5	81.98±0.43	82.12±0.33	79.70±0.90↓	81.63±0.87
	CART	82.02±0.35	82.12±0.50	80.94±0.80↓	82.01±0.99
	REPTree	82.01±0.43	82.11±0.41	80.50±0.82↓	82.14±1.11
	BFTree	81.28±0.62	81.47±0.72	78.88±1.50↓	80.97±1.44
Marketing	C4.5	30.68±0.44	30.78±0.54	29.78±0.54↓	30.55±0.64
	CART	33.43±0.49	33.53±0.39	32.43±0.59↓	33.23±0.59
	REPTree	32.48±0.65	32.68±0.55	31.58±0.45↓	32.78±0.43
	BFTree	31.58±0.45	31.68±0.55	30.48±0.35↓	31.76±0.44
Ozone	C4.5	96.71±0.30	96.25±0.27↓	96.33±0.67↓	96.49±0.29↓
	CART	96.66±0.27	97.09±0.07↑	96.25±0.75↓	96.55±0.33
	REPTree	96.77±0.27	97.04±0.17↑	96.51±0.72	96.71±0.26
	BFTree	96.61±0.30	97.04±0.18↑	96.12±0.77	96.45±0.32
Balance	C4.5	94.50±0.94	77.41±1.26↓	91.91±1.24↓	93.10±1.39↓
	CART	94.51±0.80	77.97±1.19↓	91.83±1.16↓	92.99±1.50↓
	REPTree	94.38±1.06	77.23±1.70↓	91.44±1.10↓	92.95±1.44↓
	BFTree	94.19±1.03	77.44±0.99	91.37±1.28	92.78±1.50
Diabetes	C4.5	69.64±1.58	68.98±1.48	68.82±1.55	69.38±1.16
	CART	69.80±1.91	69.29±1.66	68.98±1.64	69.60±1.76
	REPTree	69.13±1.81	68.92±1.68	68.63±1.84	68.78±1.74
	BFTree	69.02±1.68	67.68±1.46↓	68.35±1.32	68.70±1.55
Iris	C4.5	91.88±2.15	89.35±2.10↓	85.88±2.37↓	93.84±1.15↑
	CART	92.08±2.29	89.62±1.76↓	86.08±2.62↓	93.91±1.47↑
	REPTree	91.95±2.34	86.15±2.26↓	85.42±2.81↓	94.57±1.24↑
	BFTree	92.31±1.94	89.73±1.80↓	86.08±2.82↓	94.28±1.08↑
Liver	C4.5	64.02±1.90	61.56±2.15↓	63.67±2.30	65.80±1.75↑
	CART	64.23±2.51	63.27±2.20	63.84±2.31	65.73±1.99↑
	REPTree	63.78±1.96	63.38±2.61	62.97±2.19	64.08±1.89
	BFTree	63.79±2.21	62.88±2.33	62.96±2.32	64.52±1.56
Statlog	C4.5	74.01±2.56	71.97±2.80↓	73.79±2.35	73.64±2.66
	CART	74.12±2.33	72.45±2.23↓	72.54±2.38↓	73.75±2.38
	REPTree	73.98±2.02	72.27±2.68↓	72.67±2.09↓	73.85±2.12
	BFTree	73.79±2.10	71.86±2.50↓	71.88±2.45↓	73.61±1.89

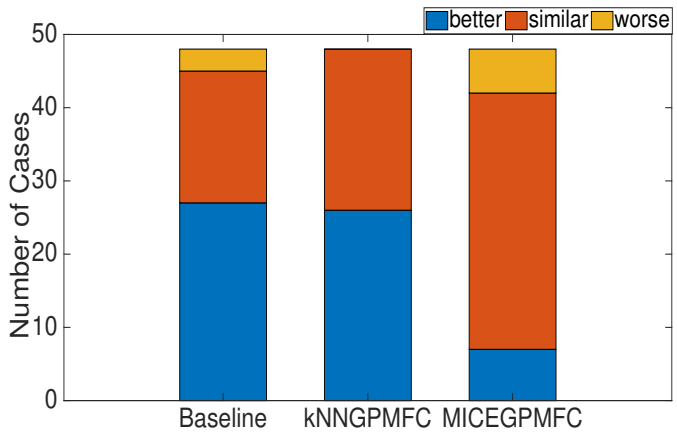


Figure 5.4: Accuracy comparison of IGPMFC with Baseline, kNNGPMFC and MICEGPMFC.

achieve better classification accuracy than using GPMFC combined with kNN-based imputation in most cases. Moreover, IGPMFC is comparable with GPMFC combined with using MICE that is expensive for classification tasks.

5.5.2 Effect of Constructed Features on the Complexity of Classifiers

The complexity of the decision trees is evaluated by the average number of nodes in the decision trees. A decision tree with a small number of nodes is preferred because having too many nodes is often a symptom for poor generalisation, especially in non-rectangular decision spaces [116]. Table 5.3 shows the average of number of nodes in decision trees generated by the proposed method and the benchmark methods.

Figure 5.5 summarises the percentage of size reduction by using the constructed features generated by IGPMFC over using the original features ($reduction$=$\frac{size_{Baseline}-size_{IGPMFC}}{size_{Baseline}}$). As can be seen from Figure 5.5 that using the constructed features generated by IGPMFC helps to decrease

Table 5.3: Average of Size of classifiers (number of nodes in decision trees)

Dataset	Classifier	IGPMFC	Baseline	kNNGPMFC	MICEGPMFC
Bands	C4.5	5.10	85.10	5.76	5.71
	CART	10.20	57.05	24.10	17.78
	REPTree	22.94	43.50	30.71	30.78
	BFTree	34.32	99.38	44.84	26.44
Breast Cancer	C4.5	6.57	23.20	7.08	6.92
	CART	4.49	16.57	5.03	4.84
	REPTree	5.46	13.14	5.64	5.46
	BFTree	12.01	29.82	12.20	12.12
Cleveland	C4.5	53.38	78.50	58.96	59.07
	CART	13.86	14.59	14.37	14.74
	REPTree	17.28	18.57	17.68	17.40
	BFTree	31.66	34.00	32.57	32.52
Hepatitis	C4.5	7.10	17.04	7.28	7.20
	CART	6.97	7.38	7.75	6.98
	REPTree	6.10	7.08	7.02	6.34
	BFTree	11.28	22.23	11.67	10.57
Mamographic	C4.5	10.44	10.46	11.36	10.27
	CART	13.48	17.31	16.76	15.64
	REPTree	23.62	26.68	25.51	24.64
	BFTree	38.78	42.56	44.29	40.19
Marketing	C4.5	2035	2077	2055	2042
	CART	30.10	40.50	35.10	30.00
	REPTree	315.50	360.20	315.52	332.25
	BFTree	325.70	380.40	325.72	352.45
Ozone	C4.5	6.43	24.30	6.20	7.56
	CART	4.43	5.43	3.63	4.96
	REPTree	3.16	5.63	5.16	5.33
	BFTree	4.80	8.30	7.50	9.53
Balance	C4.5	9.89	57.48	11.42	16.49
	CART	8.22	68.70	10.65	13.16
	REPTree	8.42	43.03	10.25	11.60
	BFTree	17.15	156.10	23.46	23.24
Diabetes	C4.5	6.38	18.98	7.16	7.43
	CART	16.57	22.93	20.57	19.96
	REPTree	31.32	35.39	31.80	31.50
	BFTree	26.90	73.04	41.63	35.93
Iris	C4.5	6.83	14.01	8.20	5.09
	CART	5.86	17.10	7.16	5.00
	REPTree	5.50	9.51	6.84	5.01
	BFTree	7.40	23.56	8.88	5.82
Liver	C4.5	5.13	34.80	5.24	5.95
	CART	14.48	39.27	14.10	12.92
	REPTree	19.38	28.58	20.89	19.32
	BFTree	32.37	65.32	32.96	33.52
Stalog	C4.5	4.39	25.94	6.32	6.40
	CART	12.80	25.76	16.61	12.46
	REPTree	10.51	17.99	13.02	11.22
	BFTree	24.52	49.42	22.92	22.04

considerably the complexity of the learned classifiers. In most datasets, except for REPtree achieving about 30% size reduction in some datasets, the complexity of the other three classifiers reduces around 50%. Especially, on the Balance Scale dataset, the decrease in complexity is around 90%.

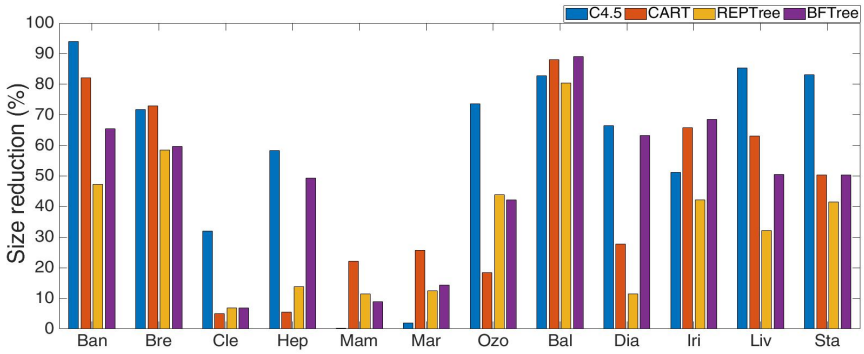


Figure 5.5: Size reduction by using IGPMFC compared to *Baseline*.

Figure 5.6 shows the average of ratio tree size of the other methods over IGPMFC. On average, *Baseline* generates about 4.0 times bigger trees than those using IGPMFC, and both kNNGPMFC and MICEGPMFC also generate bigger trees than IGPMFC.

In summary, in all cases, IGPMFC can dramatically reduce the complexity of the classifiers by using original features. Furthermore, IGPMFC can better reduce the complexity of the classifiers than GPMFC combined with simple or sophisticated imputations.

5.5.3 Computation Time

Table 5.4 shows the average computation time of IGPMFC and the other methods for constructing new features in the testing process. It is clear from Table 5.4 that the proposed method is the fastest method and following kNN-based imputation and MICE. Especially, GPMFC with MICE is million times slower than the other methods. The main reason is that

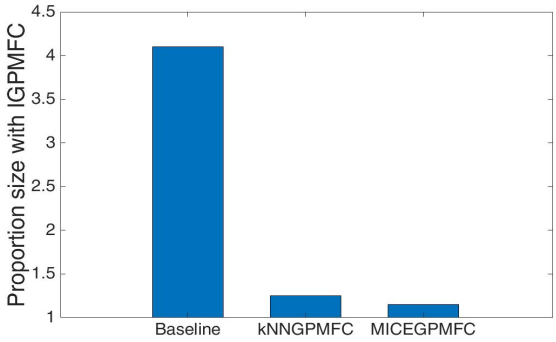


Figure 5.6: The average of ratio tree sizes of Baseline, kNNGPMFC and MICEGPMFC over IGPMFC.

MICE requires rebuilding the regression functions using all the training data and the new instance each time when it needs to estimate missing values in a new instance.

In summary, IGPMFC is faster than GPMFC combined with single imputation, and it is much faster than GPMFC combined with multiple imputation.

5.6 Chapter Summary

The goal of this chapter was to develop feature construction for incomplete data. To achieve this goal, this chapter develops IGPMFC which is a GP-based feature construction for classification with incomplete data. IGPMFC is extended from GPMFC which is a recent promising feature construction method, but it cannot directly work with incomplete data. IGPMFC uses GP with a set of interval functions to tackle with missing values.

This chapter shows that IGPMFC can be more accurate than the combination of GPMFC with single imputation. The reason is that an interval can reflect better the uncertainty of missing values than a single value

Table 5.4: Computation time of different methods for constructing multiple features (millisecond).

Dataset	IGPMFC	kNNGPMFC	MICEGPMFC
Ban	5.3×10^{-6}	1.7×10^{-1}	1.6×10^{4}
Bre	2.7×10^{-6}	8.1×10^{-1}	2.3×10^{2}
Cle	1.6×10^{-6}	4.2×10^{-1}	1.1×10^{2}
Hep	1.2×10^{-5}	3.8×10^{-1}	1.5×10^{4}
Mam	6.3×10^{-5}	2.7×10^{-1}	2.8×10^{4}
Mar	6.3×10^{-5}	5.9×10^{-1}	3.7×10^{4}
Ozo	9.2×10^{-4}	9.8×10^{-1}	8.5×10^{5}
Bal	5.3×10^{-6}	7.2×10^{-2}	2.4×10^{3}
Dia	7.4×10^{-6}	8.3×10^{-2}	3.5×10^{3}
Iri	6.6×10^{-6}	4.6×10^{-2}	3.3×10^{3}
Liv	2.3×10^{-6}	1.9×10^{0}	1.1×10^{3}
Sta	5.5×10^{-6}	3.5×10^{-2}	2.2×10^{3}

generated by single imputation. IGPMFC can achieve comparable accuracy with the combination of GPMFC with sophisticated imputation such as MICE. Moreover, IGPMFC can generate less complex classifiers than the combination of GPMFC with imputation. Furthermore, IGPMFC can construct features more effectively than the combination of GPMFC with imputation because IGPMFC does not require any time to estimate missing values.

This chapter also shows that IGPMFC can enhance the accuracy and reduce the complexity of classifiers compared to using original features. The key reason is that by constructing new features, IGPMFC helps classifiers be able to transform their input to gain class separability.

Chapters 3, 4 and 5 presented three ways to improve input space for classification with incomplete data. Another way to improve classification with incomplete data is to enhance classifiers. Next two chapters will describes new approaches to improving classifiers for incomplete data.

Chapter 6

An Effective and Efficient Ensemble Approach for Classification with Incomplete Data

6.1 Introduction

One of the most common approaches to classification with incomplete data is to use imputation methods to substitute missing values with plausible values [100, 109]. Imputation can provide complete data which can then be used by any classification algorithm. Simple imputation methods such as mean imputation are often efficient but they are often not accurate enough. In contrast, powerful imputation methods such as multiple imputation [175] are usually more accurate, but are computationally expensive [46, 164]. It is not straightforward to determine how to combine classification algorithms and imputation in a way that is both effective and efficient, particularly in the application process.

Ensemble learning is the process of constructing a set of classifiers in-

stead of a single classifier for a classification task, and it has been proven to improve classification accuracy [124]. Ensemble learning has also been applied to classification with incomplete data by building multiple classifiers in the training process and then applicable classifiers are selected to classify each incomplete instance in the application process without requiring any imputation method [24, 125, 185]. However, existing ensemble methods for classification with incomplete data often cannot work well on datasets with numerous missing values [24, 185]. Moreover, they usually have to build a large number of classifiers, which then require a lot of time to find applicable classifiers for each incomplete instance in the application process, especially when incomplete datasets contain a high proportion of missing values [24, 125]. This chapter shows how to construct a compact set of classifiers able to work well even on datasets with numerous missing values.

Feature selection is the process of selecting relevant features from original features, and it has been widely used to improve classification with complete data [183]. Feature selection has also been investigated in incomplete data [36, 162], but the existing methods typically still use imputation to estimate missing values in incomplete instances before classifying them. By removing redundant and irrelevant features, feature selection has the potential of reducing the number of incomplete instances, which could then improve accuracy and speed up classifying incomplete instances. However, this aspect of feature selection has not been investigated. This chapter also shows how to utilise feature selection to improve accuracy and speed up the application process for classification with incomplete data.

6.1.1 Goals

To deal with the issues stated above, this chapter aims to develop an effective and efficient approach to classification with incomplete data, which

use three techniques: imputation, feature selection and ensemble learning. Imputation is used to transform training incomplete data to training complete data which is then further enhanced by feature selection. After that, the proposed method builds a set of specialised classifiers which can classify unseen incomplete instances without the need of imputation. The proposed method is compared with other common approaches for classification with incomplete data to investigate the following main objectives:

1. How to effectively and efficiently use multiple imputation for classification with incomplete data; and

2. How to use feature selection for classification with incomplete data to not only improve classification accuracy but also speed up classifying new instances; and

3. How to build a set of classifiers which can effectively and efficiently classify incomplete instances without the need of imputation; and

4. Whether the proposed method can be more accurate and faster than using imputation both in the training process and the application process; and

5. Whether the proposed method can be more accurate and faster than the existing ensemble methods.

6.1.2 Organisation

The rest of this chapter is organised as follows. Section 6.2 describes the proposed method. Section 6.3 explains experiments to evaluate the proposed methods. Section 6.4 presents the results and analysis. Section 6.5 provides a summary of this chapter.

6.2 Proposed Algorithms

This section presents the proposed method in detail. It starts with showing the definitions used in the method. The section then presents the overall structure and the underlying ideas of the method. After that, it describes the details of the training process and the application process.

6.2.1 Definitions

Let $D = \{(X^i, c^i)|i = 1, .., m\}$ denote a dataset, where each X^i represents an input instance with its associated class label c^i, and m is the number of instances in the dataset. The input space is defined by a set of n features $F = \{F_1, ..., F_n\}$. Each instance X^i is represented by a vector of n values $(x_1^i, x_2^i, ...x_n^i)$, where an x_j^i is either a valid value of the j^{th} feature F_j, or is the value "?", which means that the value is unknown (a missing value).

An instance X^i is called an incomplete instance if it contains at least one missing value. A dataset, D, is called an incomplete dataset if it contains at least one incomplete instance. A feature, F_j, is called an incomplete feature for a dataset if the dataset contains at least one incomplete instance, X^i with a missing value x_j^i. For example, the incomplete dataset shown in Table 6.1 contains five incomplete instances: X^2, X^4, X^5, X^6, and X^7. It has four incomplete features: F_1, F_3, F_4 and F_5.

Table 6.1: An example dataset with missing values.

	F_1	F_2	F_3	F_4	F_5	c
X^1	5	67	3	5	3	1
X^2	4	43	1	1	?	1
X^3	4	28	1	1	3	0
X^4	5	74	1	5	?	1
X^5	4	56	1	?	3	0
X^6	4	70	?	?	3	0
X^7	?	66	?	?	1	1

A subset of features, $S \subset F$, is called a *missing pattern* in a dataset D if there is at least one instance, X^i in D, such that the value in X^i for each feature in S is missing, and the value in X^i for all the other features are known. That is, $S \subset X$ is a missing pattern in D if there exists an instance X^i in D such that if $F_j \in S$, $x^i_j = ?$ otherwise, x^i_j is not missing. For example, the dataset shown in Table 6.1 has five missing patterns: $\{\emptyset\}$, $\{F_5\}$, $\{F_4\}$, $\{F_3, F_4\}$ and $\{F_1, F_3, F_4\}$.

Algorithm 5 shows the steps to identify all missing patterns of a dataset. We use $\mathcal{MP}$ to denote the all missing patterns. At the beginning of the algorithm, $\mathcal{MP}$ is empty. The outer loop in the algorithm iterates over all instances. For each instance, all features with missing values are combined to form a missing pattern. If the missing pattern is not yet in $\mathcal{MP}$, it will be added in $\mathcal{MP}$. By the end of the algorithm, $\mathcal{MP}$ contains all missing patterns.

Given a dataset D and a feature subset S, we use D_S to represent the projected dataset D onto the features in S, i.e. the dataset D reduced to the feature subset S. That is, each instance in D is replaced by the projected instance in which values for features not in S are removed. For example, given the dataset shown in Table 6.1 with five features, the data subset $D_{\{F_1,F_2,F_3\}}$ is shown in Table 6.2.

6.2.2 Overall Proposed Method

The proposed method has two main processes: a training process and an application process. The training process constructs an ensemble of classifiers which is then used to classify new instances in the application process. Figure 6.1 shows the flowchart of the method.

The method is based on three key ideas. The first idea is that the method constructs an ensemble of classifiers to cover possible missing patterns, each classifier being built on one missing pattern. Therefore, new incomplete instances can be classified by the ensemble without re-

Algorithm 5: $MissingPatterns(D)$

Input:
D, a dataset
Output:
$\mathcal{MP}$, a set of missing patterns

```
1  MP ← {}
2  for i ← 1 to m do
3  |  temp ← {∅}
4  |  for j ← 1 to n do
5  |  |  if x_j^i =? then
6  |  |  |  temp ← temp ∪ F_j
7  |  |  end
8  |  end
9  |  if temp ∉ MP then
10 |  |  MP ← MP ∪ temp
11 |  end
12 end
13 return MP
```

quiring imputation. This is not completely novel —it has also been used in [24, 185]. The second idea is to use imputation in the training process, but not in the application process. Using a powerful imputation method to generate high quality complete training data for building classifiers results in more accurate classifiers. In contrast, existing ensemble methods based on missing patterns [24, 185] do not use any imputation, so the training set for each classifier may be as small as a single instances which leads to low accuracy. However, good imputation methods such as multiple imputation are computationally expensive. For the training process, there is no time limit for many applications, and the high cost of multiple imputation is not a problem; in the application process, there may be tight time limits on the process of classifying a new instance, and using multiple imputation may be infeasible. The third idea is to use feature selection to

Table 6.2: The dataset in Table 6.1 reduced to the feature subset $\{F_1, F_2, F_3\}$.

	F_1	F_2	F_3	c
X^1	5	67	3	1
X^2	4	43	1	1
X^3	4	28	1	0
X^4	5	74	1	1
X^5	4	56	1	0
X^6	4	70	?	0
X^7	?	66	?	1

further improve the training data. By removing redundant and irrelevant features, feature selection not only produces a high quality feature set, but also reduces the number of missing patterns and removes missing values of incomplete instances in the application process.

6.2.3 Training Process

The purpose of the training process is to build a set of classifiers, one classifier for each missing pattern. Algorithm 6 shows the main steps of the training process. The inputs of the algorithm are an original dataset D and a classifier learning algorithm $\mathcal{L}$.

The algorithm starts with using multiple imputation to estimate missing values in the original dataset D to generate an imputed dataset $ImpD$ which is complete. After that, feature selection is applied to the complete dataset D to select the subset $\mathcal{SF}$ of important features. The missing patterns algorithm is then used to search for all missing patterns, $\mathcal{MP}$, from the *original* (incomplete) dataset reduced to the selected features $D_{\mathcal{SF}}$. Subsequently, for each missing pattern $\mathcal{MP}_i$, a "complete pattern" $\mathcal{CP}_i$ is generated by selecting features which are in $\mathcal{SF}$, but not in $\mathcal{MP}_i$. After that, the imputed dataset is reduced to the features in the complete pattern and then split into a training dataset and a validation dataset. The training

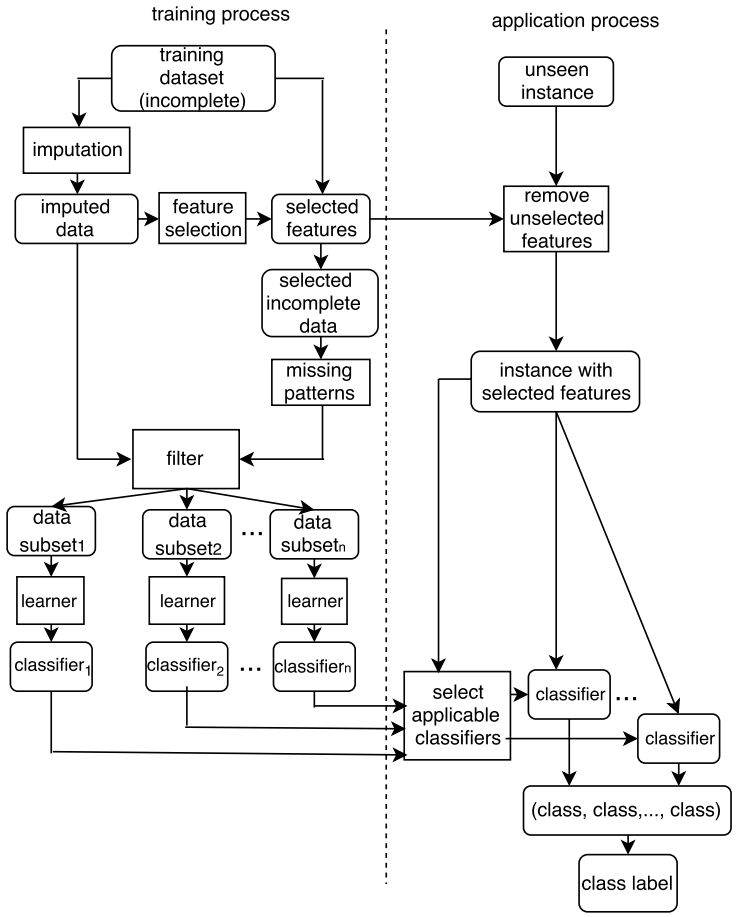


Figure 6.1: The proposed method builds an ensemble of classifiers then used to classify new incomplete instances without imputation.

dataset is used to construct a classifier which is evaluated using the validation dataset. The average accuracy on the validation set becomes the score (or weight) of the classifier. As a result, the application process generates a set of classifiers, one classifier on each missing pattern.

The four main components in the training process are imputation, feature selection, identifying missing patterns and learning the classifiers. Either single imputation or multiple imputation can be used to transform the incomplete training data to the imputed training data. Multiple impu-

Algorithm 6: The training process

Input:
D, an original training dataset
$\mathcal{L}$, a classifier learning algorithm
Output:
$\mathcal{C}$, a set of learnt classifiers
w, the weighting of classifiers
$\mathcal{SF}$, a set of selected features

$ImpD \leftarrow MultipleImputation(D)$
$\mathcal{SF} \leftarrow FeatureSelection(ImpD)$
$\mathcal{MP} \leftarrow MissingPaterns(D_{\mathcal{SF}})$
$\mathcal{C} \leftarrow \{\}$
foreach $\mathcal{MP}_i \in \mathcal{MP}$ **do**
  $\mathcal{CP}_i \leftarrow \mathcal{SF} - \mathcal{MP}_i$
  Divide $ImpD_{\mathcal{CP}_i}$ into $ImpTrain$ and $ImpValidation$
  $classifier_i \leftarrow \mathcal{L}(ImpTrain)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup classifier_i$
  $\mathbf{w}_i \leftarrow classifier_i(ImpValidation)$
end
return $\mathcal{C}$, **w** and $\mathcal{SF}$;

tation is generally more accurate than single imputation, especially when the data contains a large number of missing values [102]. Therefore, a multiple imputation method such as MICE should be used to estimate missing values for the training data where possible. However, multiple imputation is usually much more expensive than single imputation, especially when data contains a large number of features such as in gene expression datasets [31]. Therefore, with datasets containing numerous features, a good single imputation method such as kNN-based imputation to estimate missing values for the training data can be used which makes the imputation cost in the training process feasible.

Feature selection can also be expensive, and the choice of both the evaluation method and the search method must be made carefully. Wrapper evaluation methods are often more accurate than filter methods, but generally more expensive, especially with large training datasets or if the wrapper methods use expensive classifiers such as multiple layer perceptron. There exist fast filter methods such as CFS [66] and mRMR [122] have comparable accuracy to wrapper methods, and these filter methods could be used to evaluate feature subsets efficiently, even when the training data contains a large number of instances and features. For search techniques, evolutionary techniques have been proven to be effective and efficient for feature selection. Therefore, using evolutionary techniques such as GAs and PSO to search for feature subsets and CFS or mRMR to evaluate them enables feature selection to be done efficiently.

Searching for all missing patterns is not time-consuming. The computation time of Algorithm 5 is *O(m*n)* where m is the number of instances, and n is the number of features which is no more than the cost of reading the dataset (assuming that *temp* is represented as a bitset, and $\mathcal{MP}$ as a hash table).

If there are a large number of missing patterns, then the cost of training a set of classifiers for all missing patterns may be very expensive. Existing ensemble methods search for missing patterns in the original training data [24, 184, 185]; therefore, they often get a very large number of missing patterns when the training data contains numerous missing values. In contrast, the proposed method searches for missing patterns in the training data after it has been reduced to the selected features. This reduces the number of missing values, often by a large fraction. Therefore, the proposed method often generates a much smaller number of missing patterns even when the original training data contained numerous missing values. Therefore, the cost of the classifier learning could be much less than in other ensemble methods.

6.2.4 Application Process

The application process is to classify new instances using the learnt classifiers. Algorithm 7 shows the main steps of the application process. The inputs of the algorithm are an instance needed to be classified X, an ensemble of learnt classifiers $\mathcal{C}$ along with their weighs **w** and a set of selected features $\mathcal{SF}$. The algorithm will output the most suitable class label for the instance.

Algorithm 7: The application process

Input:
X, an instance to be classified
$\mathcal{C}$, a set of learnt classifiers
w, weights of classifiers
$\mathcal{SF}$, a set of selected features
Output:
the class of x

```
1  Reduce X to only containing the features in SF
2  AC ← C
3  foreach missing values x_j =? in reduced X do
4  |  foreach classifier ∈ AC do
5  |  |  if classifier requires F_j then
6  |  |  |  AC ← AC − classifier
7  |  |  end
8  |  end
9  end
10 Apply each classifier in AC to reduced X
11 return majority vote of classifiers, weighted by w;
```

The algorithm starts by removing features in the instance X which are not in the set of selected features $\mathcal{SF}$. Next, the algorithm searches for all classifiers which are applicable to the instance—classifiers which do not require any incomplete features in the instance. Subsequently, each applicable classifier is used to classify the instance. Finally, the algorithm

returns a class by taking a majority vote of the applicable classifiers' predictions, weighted by the quality of the classifiers measured on the training process.

Typical methods for classification with incomplete data perform imputation on the new instance. In order get adequate accuracy, it is particularly important to use a high quality imputation method such as MICE, which is very expensive. The proposed method, on the other hand, does not require any imputation method to estimate missing values for unseen incomplete instances. Therefore, the proposed method is expected to be faster than the common approach.

To classify an incomplete instance, the proposed method also reduces the instance to contain only selected features. After this reduction step, the incomplete instance frequently becomes a complete instance, which in turn removes the need to search for applicable classifiers for the instance. Moreover, because of the feature selection, the proposed method often generates a smaller number of classifiers than existing ensemble methods which reduces the cost of the search if the instance is incomplete. Therefore, the proposed method is expected to be considerably faster than existing ensemble methods for classification with incomplete data.

6.3 Design of Experiments

This section discusses the aim and design of the experiments. The discussion consists of methods for comparison, datasets and parameter settings.

6.3.1 Benchmark Methods for Comparison

In order to investigate the effectiveness and efficiency of the proposed method, namely *NewMethod*, its accuracy and computation time are compared with five benchmark methods. The first two methods are common

approaches to classification with incomplete data by using imputation in both training process and application process. The other three methods are ensemble methods for classification with incomplete data without requiring imputation. The details of the five methods are as follows:

- The first benchmark method, namely *kNNI*, is to use kNN-based imputation, which is one of the most common single imputation methods, to estimate missing values for both training data and unseen instances. This benchmark method provides complete training data and complete unseen instances which can be used by any classification algorithm. Comparing the proposed method with this benchmark method can show the proposed method's advantages over one of the most common methods for classification with incomplete data.

- The second benchmark method, namely *MICE*, is to use MICE, which is a powerful multiple imputation method, to estimate missing values for both training data and unseen incomplete instances. Both the proposed method and this benchmark method use multiple imputation to estimate missing values for training data; the key difference is that this benchmark method requires multiple imputation to estimating missing values for incomplete unseen instances in the application process which is very expensive. However, the proposed method can classify unseen instances by constructing a set of classifiers instead of requiring multiple imputation. Therefore, comparing the proposed method with this benchmark method can show the proposed method's effectiveness and efficiency in classifying incomplete instances.

- The third benchmark method, namely *Ensemble[24]*, is a recent ensemble method for classification with incomplete data in [24]. Both the proposed method and this benchmark method search for missing patterns and build one classifier for each missing pattern. One

difference is that this benchmark method does not use any imputation method to fill missing values in the training data, while the proposed method uses a powerful imputation method to estimate missing values and provide complete data for the training process. Another difference is that this benchmark method does not use any technique to reduce the number of missing patterns; hence it may have to build a large number of classifiers. In contrast, the proposed method uses feature selection to reduce the number of missing patterns, so it is expected to speed up classifying unseen instances by building a compact number of classifiers. Therefore, comparing the proposed method with this benchmark method can show the proposed method's benefits due to using multiple imputation and feature selection.

- The fourth benchmark method, namely *Ensemble[185]*, is a very recent extension of the third benchmark method, using the mutual information to reduce the number of missing patterns [185]. Comparing the proposed method with this benchmark method can shows the proposed method's advantages due to using feature selection to not only reduce the number of missing patterns, but also reduce the number of missing values in unseen incomplete instances.

- The final benchmark method, namely *Ensemble[125]*, is an ensemble method for classification with incomplete data in [125]. This benchmark method randomly generates missing patterns rather than exploring missing patterns from the training data; hence it has to build a large number of classifiers. Therefore, comparing the proposed method with this benchmark method can show the proposed method's effectiveness and efficiency thanks to searching for missing patterns from training data.

6.3.2 Datasets and Parameter Settings

In order to examine the performance of the proposed method, a set of experiments have been conducted on 15 real-world incomplete datasets. The details of these datasets can be seen in Table 1.1 in page 18. As in previous chapters, ten-fold cross-validation is used to divide the datasets into training and test datasets.

The experiments use two imputation methods: knn-based imputation and MICE, representing two types of imputation, single imputation and multiple imputation, respectively. As in previous Chapters, with kNN-based imputation, the number of nearest neighbours, k, is set 1 because it is simple and quick. The implementation of MICE using R language in [22] is used to run MICE where random forest is used as a regression method. The number of cycles is set 5 and the number of imputed datasets is set 20 following the recommendation in [175].

The proposed approach is a framework, so any feature selection method can be used to select relevant features. The experiments use a filter-based feature selection method because a filter method is often quicker and more general than a wrapper method. The Correlation Feature Selection (CFS) measure [66] is used to evaluate feature subsets. The main reason is that CFS not only can evaluate the correlation between each feature with the class, but also can evaluate the uncorrelation between features in the feature subset. PSO is used to search for feature subsets because it has been successfully applied to feature selection as shown in Chapter 3. The parameters of PSO for feature selection are set as follows. The number of particles is set to 50, and the maximum number of generations is set to 100. CFS and PSO are implemented under the WEKA [65].

In machine learning, decision trees such as *C4.5* [129], rule-based classifiers such as *PART* [52], and function-based classifiers such as a multilayer perceptron (*MLP*) [67] are suitable to ensemble learning. Therefore, three classification algorithms (C4.5, PART, and MLP) are used to compare the proposed method with the other benchmark methods. The classification

algorithms are implemented under the WEKA [65] with default parameter settings.

The proposed method, *Ensemble[24]* and *Ensemble[185]* automatically identify the number of classifiers from the training data. The number of classifiers in Ensemble[125] is set equally to the number of classifiers explored by *Ensemble[24]*.

6.4 Results and Discussions

This section presents and discusses the experimental results. It first shows the comparison on accuracy between the proposed method and the benchmark methods. It then presents the comparison between them on computation time. Further analysis is also discussed to demonstrate the advantages of the proposed method.

6.4.1 Accuracy

6.4.1.1 Comparison Method

Table 6.3 shows the mean and standard deviation of classification accuracies of the proposed method and the benchmark methods. The first column shows datasets, and the second column shows classification algorithms used in the experiments. "*NewMethod*" refers to the proposed method. The rest five columns are the five benchmark methods. "*kNNI*" and "*MICE*" refer to the benchmark methods using kNN-based imputation and MICE to estimate missing values, respectively. "$Ensemble$[24]", "$Ensemble$[185]" and "$Ensemble$[125]" refer to the three ensemble benchmark methods in [24], [185] and [125], respectively. The values in the table are the average classification accuracy $\pm$ standard deviation resulting from combining a classifier (row) with an approach to classification with incomplete data (column).

It is very important to choose a suitable statistical test to correctly evaluate the significance of the results. A multiple test rather than a pair test should be used to compare the proposed method with the multiple (five) benchmark methods. Moreover, a non-parametric test rather a parametric test should be used because non-parametric tests do not require the normal distribution of data as parametric tests. Therefore, the *Friedman* test [33], which is one of the most popular multiple non-parametric tests, is used to test the significance of the results. The test indicates that there exits significant differences between the methods in each dataset and each classifier. Therefore, the *Holm* procedure [33], which is a post-hoc procedure, is used to perform pair tests between two methods. In Table 6.3, the symbol $\uparrow$ indicates that the benchmark method is significantly better than the proposed method. In contrast, the symbol $\downarrow$ shows that the benchmark method is significantly worse than the proposed method.

6.4.1.2 Compare with Imputation Methods

Figure 6.2 shows the fraction of cases that the proposed method is significantly better or worse than the benchmark methods. It is clear from Fig 6.2 that the proposed method can achieve significantly better accuracy than using imputation (*kNNI* and *MICE*) in most cases. The proposed method is significantly better than both *kNNI* and *MICE* in about 50% cases, and it is only significantly worse than both of them in one out of the 45 cases.

The proposed method is more accurate than the imputation methods because it incorporates feature selection to remove redundant and irrelevant features, which helps to improve classification accuracy. Furthermore, the proposed method can construct multiple classifiers, which can be more comprehensive and generalise better than constructing a single classifier. Therefore, the proposed method can classify new instances better than the imputation methods.

Table 6.3: Mean and standard deviation of classification accuracies.

Data	Classifier	The proposed and the benchmark methods					
		NewMethod	kNNI	MICE	Ensemble[24]	Ensemble[185]	Emsemble[125]
Arr	J48	66.45±2.07	65.30±2.44	65.43±2.02	**66.99**±1.78	59.01±12.31↓	54.46±9.62↓
	PART	64.36±2.82	63.76±2.04	64.06±1.89	**64.46**±1.75	57.69±11.85↓	48.99±12.12↓
	MLP	64.90±2.22	65.49±1.94	**65.85**±1.69	65.02±1.64	59.60±12.68	53.63±11.49↓
Aut	J48	**68.78**±4.28	68.37±3.50	68.09±4.17	66.46±4.19↓	66.34±4.29↓	63.50±4.90↓
	PART	**65.37**±3.87	64.34±4.05	64.54±4.24	65.93±3.84	64.43±3.84	63.36±4.60↓
	MLP	**67.67**±4.07	66.04±3.35↓	66.46±3.42	65.17±3.78↓	64.75±3.13↓	61.76±4.96↓
Ban	J48	69.96±1.53	66.05±1.91↓	**71.43**±1.80↑	69.40±1.13	69.83±1.55	64.49±1.33↓
	PART	69.22±1.39	64.00±1.96↓	68.85±1.40	**69.34**±1.25	68.75±1.29	63.26±1.44↓
	MLP	67.07±1.08	62.91±1.76↓	66.66±2.04	**67.23**±1.22	67.05±1.36	62.19±2.03↓
Bre	J48	**94.30**±0.78	93.88±0.64	94.04±0.74	94.32±0.47	93.85±0.48	90.54±1.07↓
	PART	94.53±0.67	94.66±0.42	**94.61**±0.46	94.44±0.46	94.56±0.51	90.84±1.50
	MLP	95.52±0.64	95.37±0.43	**95.71**±0.41	95.59±0.39	95.17±0.41	90.89±1.44↓
Chr	J48	**99.10**±0.36	99.08±0.47	97.53±0.62↓	94.21±0.46↓	94.18±0.35↓	97.31±0.68↓
	PART	**99.30**±0.29	**99.30**±0.53	98.40±0.68↓	94.28±0.49↓	94.20±0.31↓	97.34±0.84↓
	MLP	96.83±0.50	**99.00**±0.35↑	96.48±0.44	93.93±0.33↓	93.90±0.40↓	96.75±0.72
Cle	J48	**58.19**±1.55	54.16±2.02↓	54.09±1.98↓	55.17±1.74↓	53.10±1.49↓	56.79±1.10↓
	PART	**56.48**±1.88	53.54±2.16↓	53.75±1.45↓	54.00±1.71↓	51.55±2.08↓	56.46±0.98
	MLP	**57.94**±1.32	52.72±1.51↓	54.19±1.96↓	54.21±1.50↓	52.98±1.57↓	56.92±1.29↓
Cre	J48	85.15±0.51	85.30±0.59	**85.36**±0.58	85.06±0.67	84.98±0.76	79.79±1.56↓
	PART	85.44±0.59	83.83±0.94↓	83.82±0.86↓	**85.56**±0.71	84.80±0.77↓	79.65±1.68↓
	MLP	**86.12**±0.50	82.86±0.99↓	83.25±0.78↓	85.18±0.62↓	85.10±0.56↓	76.96±3.07↓
Hea	J48	**78.92**±1.51	78.33±1.56	78.25±1.29↓	76.86±1.41↓	76.76±1.49↓	63.50±6.32↓
	PART	**79.09**±0.96	78.78±1.16	78.80±1.19	77.11±1.44↓	77.61±1.50↓	65.26±4.98↓
	MLP	80.03±1.42	**80.49**±1.58	78.58±1.63↓	76.93±1.43↓	77.23±1.48↓	64.53±5.31↓
Hep	J48	**81.75**±1.42	78.55±2.05↓	80.01±2.25↓	79.80±1.76↓	80.98±1.58	81.74±1.62
	PART	81.28±1.80	79.32±2.75↓	**82.11**±1.85	80.75±1.83	80.69±1.92	81.47±2.16
	MLP	**82.80**±1.57	81.61±1.77↓	82.56±1.24	80.16±1.23↓	78.85±2.18	83.32±1.76
Hor	J48	**85.38**±0.28	82.94±1.22↓	84.06±1.21↓	85.10±0.57	85.25±0.49	63.49±2.60↓
	PART	84.76±0.48	79.23±1.32↓	79.25±1.77↓	84.69±0.62	**85.29**±0.79	62.88±2.67↓
	MLP	84.14±0.51	79.28±1.13↓	79.94±1.74↓	**84.20**±0.75	84.00±0.89	62.93±3.25↓
Hou	J48	96.28±0.30	**96.31**±0.52	96.15±0.54	93.69±0.37↓	93.70±0.39↓	91.25±0.88↓
	PART	95.70±0.36	95.72±0.79	**95.74**±0.63	94.14±0.35↓	94.31±0.44↓	91.08±1.04↓
	MLP	94.64±0.39	**94.88**±0.66	94.72±0.65	93.45±0.53↓	93.98±0.48↓	91.04±0.84↓
Mam	J48	**82.86**±0.52	81.81±0.57↓	82.24±0.65↓	82.57±0.46↓	82.33±0.49↓	79.68±1.53↓
	PART	**82.53**±0.58	81.45±0.59↓	81.52±0.70↓	82.16±0.63↓	82.07±0.62↓	79.76±1.11↓
	MLP	**83.11**±0.43	82.46±0.60	82.97±0.49	82.99±0.36	82.95±0.45	79.73±1.05↓
Mar	J48	**33.93**±0.33	30.02±0.56↓	30.01±0.41↓	33.32±0.40↓	33.43±0.37↓	31.03±0.79↓
	PART	**33.53**±0.29	28.71±0.33↓	28.83±0.42↓	32.82±0.37↓	32.82±0.42↓	31.05±0.60↓
	MLP	**33.56**±0.34	32.15±0.45↓	32.40±0.38↓	33.46±0.22	33.34±0.29	30.20±0.98↓
Ozo	J48	**97.11**±0.03	95.67±0.88↓	95.90±0.40↓	96.44±0.24↓	96.83±0.21↓	83.45±1.04↓
	PART	**97.11**±0.03	95.24±1.15↓	95.86±0.44↓	96.89±0.16↓	96.93±0.17↓	83.26±0.94↓
	MLP	**96.59**±0.58	95.99±0.68↓	96.34±0.27↓	96.48±0.19	96.40±0.17	83.16±1.11↓
Tum	J48	41.26±2.26	40.49±2.64	41.24±2.05	**42.32**±2.12	38.38±2.59↓	30.70±3.02↓
	PART	40.07±1.86	39.68±1.94	**40.49**±1.76	40.55±1.99	37.46±2.41↓	31.50±2.76↓
	MLP	39.65±1.91	38.95±1.74	39.79±2.29	**39.83**±2.09	36.40±1.96↓	33.16±3.18↓

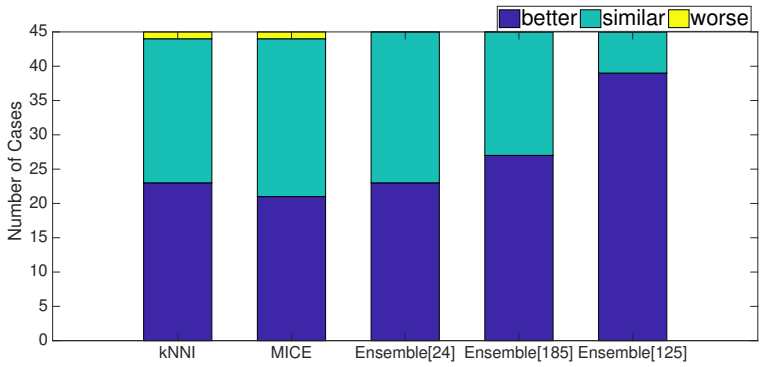


Figure 6.2: The comparison between the proposed method and each of the benchmark methods on all the classification algorithms.

6.4.1.3 Compare with Other Ensemble Methods

As also can be seen from Figure 6.2 that the proposed method also can achieve significantly better accuracy than the benchmark ensemble methods in most cases. The proposed method is significantly more accurate than the benchmark ensemble methods over 50% cases, and it is not significantly worse than the other ensemble methods.

The proposed method is more accurate than the other benchmark ensemble methods because it uses a powerful imputation to provide complete data for the training process rather than working on incomplete training data as *Ensemble[24]* and *Ensemble[185]*. The second reason is that feature selection helps further improve the training data of the proposed method. Moreover, by removing redundant and irrelevant features, feature selection helps reduce the number of incomplete instances in the application process as shown in Figure 6.3. As a result, the proposed method can more frequently choose applicable classifiers to classify incomplete instances than the other ensemble methods.

Table 6.4 shows the percentage of incomplete instances which can be classified by the ensemble methods. It is clear from Table 6.4 that the proposed method can classify all incomplete instances on nine out of fif-

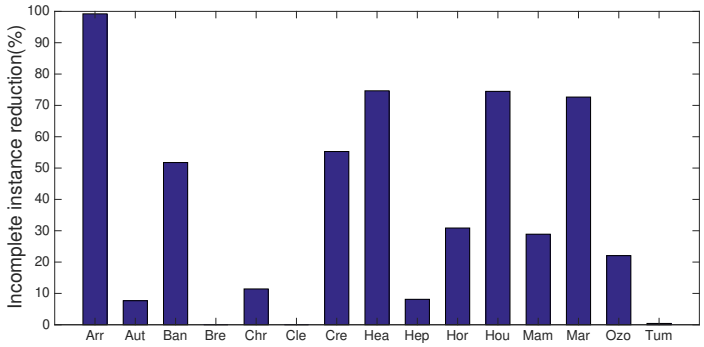


Figure 6.3: The percentage of incomplete instance reduction by using feature slection.

teen datasets and classify almost all incomplete instances on the other six datasets. In contrast, the other ensemble methods cannot well classify incomplete instances. Especially, *Ensemble[125]* only can classify 81.02% and 48.85% incomplete instances on the *Hor* and *Ozo* datasets, respectively, because *Ensemble[125]* randomly generates missing patterns instead of finding missing patterns in the training data as the other ensemble methods.

6.4.1.4 Further Comparison

As can be seen from Table 6.3 that the proposed method can obtain better accuracy than the benchmark methods not only on datasets with a small number of incomplete instances, but also on datasets with a large number of incomplete instances. For instance, the proposed method achieves the best accuracy on *Cle* dataset containing only 1.98% incomplete instances and also on *Hea* dataset with 100% incomplete instances.

Figure 6.4 shows the fraction of cases that the proposed method is significantly better or worse than the other methods on each classifier. Figure 6.4 shows that with any of the classifiers, the proposed method can significantly outperform the other methods in most cases. Moreover, *J48* can get more benefits from the proposed method. The reason is likely that a fil-

Table 6.4: The percentage of incomplete instances are classified by ensemble methods.

Data	NewMethod	Ensemble[24]	Ensemble[185]	Ensemble[125]
Arr	99.23	99.5	87.64	90.19
Aut	97.09	92.21	92.21	98.68
Ban	98.72	93.93	93.93	94.33
Bre	100	100	100	98.34
Chr	100	92.53	92.53	99.68
Cle	100	97.13	97.13	93.59
Cre	100	98.95	98.95	100
Hea	100	99.32	99.32	85.85
Hep	98.66	97.02	97.02	95.69
Hor	99.84	99.71	99.71	81.02
Hou	98.04	96.79	96.79	97.88
Mam	100	99.25	99.25	100
Mar	100	98.95	98.95	100
Ozo	100	99.82	99.82	48.85
Tum	100	99.03	99.03	100

ter feature selection often removes irrelevant and redundant features, but it may keep redundant features. *J48* can perform feature selection while constructing classifiers [135]. Therefore, by further removing redundant and irrelevant features, these classifiers can be more tolerant of missing values [164].

In summary, the proposed method can obtain significantly better accuracy than the benchmark methods in almost all cases when combining with any of the classification algorithms.

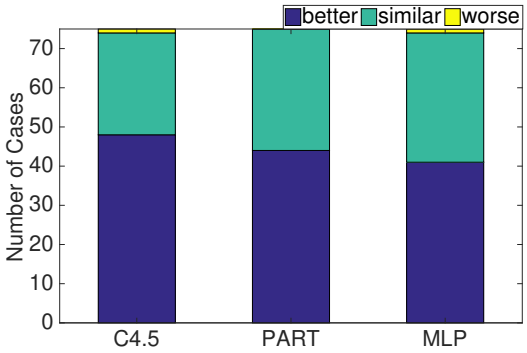


Figure 6.4: The comparison between the proposed method and all the benchmark methods on each of classification algorithms.

6.4.2 Computation Time

For most classification tasks, the training time has no constraint, but the computation time to classify an unseen instance should be feasible. Therefore, we focus on the computation time to classify unseen instances in the application process.

The experiments show that different classification algorithms have the same pattern of computation time. Hence, we only report the computation time of one classification algorithm: J48. Table 6.5 shows the computation time to classify instances in the application process.

6.4.2.1 Compare with Imputation Methods

It is clear from Table 6.5 that the proposed method is considerably more efficient than the methods using imputation (*kNNI* and *MICE*). The proposed method is thousand to million times faster than *MICE* because it does not take any time to estimate missing values in the application process. In contrast, *MICE* takes a long time to estimate missing values in the application process because *MICE* needs to rebuild all regression functions when it estimates missing values for each unseen incomplete instance. The proposed method is also remarkably more efficient than *kNNI* because

Table 6.5: Time to classify instances in the application process (millisecond).

Data	NewMethod	kNNI	MICE	Ensemble[24]	Ensemble[185]	Ensemble[125]
Arr	2.2×10^2	8.1×10^2	7.6×10^7	1.1×10^2	1.3×10^2	3.5×10^2
Aut	8	2.1×10^1	8.6×10^5	2.6×10^1	3.3×10^1	4.5×10^2
Ban	1.1×10^1	3.1×10^1	6.6×10^5	1.6×10^2	2.3×10^2	5.5×10^2
Bre	1.2	9.2	3.7×10^3	2.3	2.6	6.7
Chr	1.2×10^1	3.1×10^1	9.6×10^5	6.1×10^1	2.3×10^1	1.5×10^2
Cle	1.0	6.0	2.7×10^3	1.0	1.0	6.0
Cre	2.0	8.0	5.1×10^4	5.0	3.0	8.0
Hea	1.0	1.9×10^1	2.0×10^5	2.0	1.0	2.0
Hep	1.0	8.0	8.1×10^4	3.0	1.0	2.0
Hor	5.0	2.1×10^1	3.6×10^5	1.7×10^1	1.5×10^1	2.8×10^1
Hou	4.0	3.1×10^1	4.6×10^5	2.7×10^1	1.3×10^1	1.7×10^1
Mam	3.0	6.4×10^1	1.8×10^5	7.0	5.0	8.0
Mar	7.9×10^3	1.3×10^4	7.1×10^8	8.8×10^4	6.7×10^4	4.7×10^4
Ozo	1.2×10^4	2.7×10^4	1.2×10^9	1.3×10^5	2.9×10^4	1.7×10^5
Tum	2.1×10^1	5.3×10^1	8.2×10^5	3.1×10^1	4.0	6.2×10^1

kNNI also takes time to estimate missing values for unseen incomplete instances. Especially with big datasets such as the *Mar* and *Ozo* datasets, the proposed method is much more efficient than both *MICE* and *kNNI* because the two methods take a long time to estimate missing values in datasets with numerous instances and features.

6.4.2.2 Compare with Other Ensemble Methods

As can be seen from Table 6.5 that the proposed method is also more efficient than the benchmark ensemble methods. The first reason is that the proposed method uses feature selection to remove redundant and irrelevant features before building classifiers, so it can generate simpler clas-

sifiers than the other ensemble methods. As demonstrated in Figure 6.5 that feature selection can remove over half of the features in the majority of datasets. Moreover, by removing redundant and irrelevant features, the proposed method also reduces the number of missing patterns; therefore, it only needs to build a small number of classifiers. As is evident from Figure 6.6 that the proposed method can reduce over 50% missing patterns in many datasets. In other words, the proposed method only needs to build half of the number of classifiers compared to other ensemble methods such as the ensemble method in [24]. With a smaller number of classifiers, the proposed method can classify instances quicker than the other methods. Finally, by using feature selection, the proposed method can reduce the number of incomplete instances in the application process. Therefore, the proposed method can save time to search for applicable classifiers for incomplete instances. As can be seen from Figure 6.3 that the proposed method can significantly reduce the number of incomplete instances. For example, it can reduce over 70% incomplete instances in the *Heh*, *Hou* and *Mar* datasets.

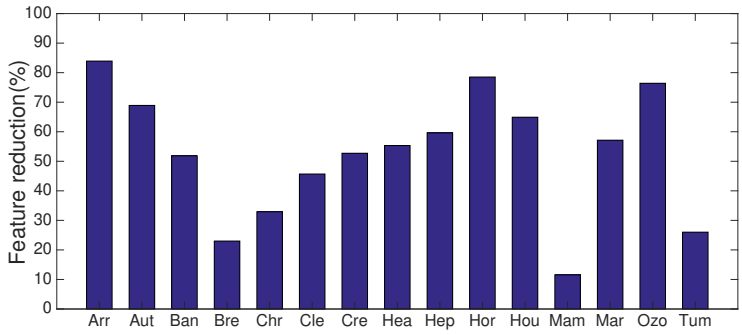


Figure 6.5: Feature reduction by using feature selection.

In summary, the proposed method can not only be more effective, but also more efficient than the other benchmark methods.

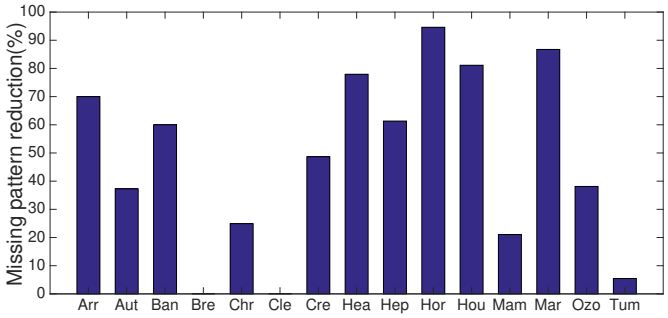


Figure 6.6: Missing pattern reduction by using feature selection.

6.4.3 Further Analysis

This section discusses further analysis to deeply understand the effectiveness and efficiency of the proposed method.

6.4.3.1 Evaluation of the Proposed Method on Different Imputation Methods

One of the important component in the proposed method is imputation. In order to know the impact of imputation on the proposed method, experiments are designed to compare the proposed method on three multiple imputation methods in MICE (using random forest regression (rf), bayesian linear regression (norm) and linear regression (nob)) and kNN-based imputation (kNNI). Table 6.6 shows the classification accuracy and the training time of the proposed methods on different imputation methods.

It is clear from Table 6.6 that the proposed method with multiple imputation is generally more accurate than the proposed method with single imputation. Moreover, the proposed method with multiple imputation using non-linear regression is usually more accurate than the proposed method with multiple imputation using linear regression. However, the training time of the proposed method using multiple imputation is much more expensive than using single imputation.

Table 6.6: Classification accuracy and training time of the proposed method by using different imputation methods.

Data	Classification accuracy				Training time			
	rf	**norm**	**nob**	**kNNI**	**rf**	**norm**	**nob**	**kNN**
Arr	**66.45**	66.32	66.10	66.24	2.1×10^5	4.2×10^4	3.6×10^3	1.8×10^1
Aut	68.78	**68.82**	68.32	68.12	4.2×10^4	4.1×10^3	2.3×10^1	1.4×10^1
Ban	**69.96**	69.03	68.11	67.24	7.2×10^4	6.1×10^3	3.8×10^1	2.4×10^1
Bre	94.30	94.36	**94.38**	93.87	3.8×10^4	3.4×10^3	6.3×10^1	1.3×10^1
Chr	99.10	99.09	**99.12**	99.04	9.1×10^4	8.2×10^3	4.6×10^1	2.8×10^1
Cle	**58.06**	57.82	57.21	57.10	7.7×10^3	6.9×10^2	2.0×10^1	7.6
Cre	85.34	85.34	**85.45**	85.16	4.2×10^4	3.2×10^3	5.2×10^1	1.2×10^1
Hea	**78.92**	78.56	78.43	78.89	7.3×10^4	6.1×10^3	3.4×10^1	1.6×10^1
Hep	**82.19**	81.96	80.88	79.32	5.8×10^4	6.1×10^3	2.1×10^1	9.1
Hor	**85.38**	85.20	84.53	83.37	9.2×10^4	8.1×10^3	4.0×10^1	1.2×10^1
Hou	95.00	94.94	94.66	**95.32**	8.1×10^4	7.3×10^3	3.6×10^1	1.5×10^1
Mam	**82.86**	82.73	82.61	82.05	2.5×10^4	1.7×10^3	4.8×10^1	1.2×10^1
Mar	**33.90**	33.86	32.89	32.21	4.5×10^4	1.7×10^4	1.2×10^3	4.2×10^2
Ozo	**97.10**	**97.10**	97.07	96.45	3.1×10^6	5.2×10^5	7.9×10^2	2.9×10^2
Tum	41.16	**41.27**	40.67	40.34	3.4×10^4	1.8×10^3	5.2×10^1	2.2×10^1

6.4.3.2 Evaluation of the Proposed Method on Gene Expression Datasets

Gene expression datasets usually contain a large number of features, and often contain a large number of missing values [31]. Therefore, we evaluate the proposed method on gene expression datasets to further validate the effectiveness and efficiency of the proposed methods. Table 3.4 in Chapter 3 in page 80 shows eight gene expression datasets which are chosen to evaluate the proposed methods.

Table 6.7 shows the classification accuracy of the proposed method and the other methods on the gene expression datasets. It is clear from the Table 6.7 that the proposed method using kNN-based imputation in

the training processes is more accurate than using kNN-based imputation both in the training and application processes. Moreover, the proposed method is much more accurate than existing ensemble methods. For example, in *tomlins* datasets, the accuracy of the proposed method is double that of the other ensemble methods in [24, 185].

Table 6.7: Classification accuracy (using *J48* as a classifier) of the proposed method (using kNN-based imputation) and the other benchmark methods on the gene expression datasets.

Dataset	NewMethod	kNNI	Ensemble[24]	Ensemble[185]	Ensemble[125]
alizadeh-2000-v1	74.41±6.41	66.81±8.39↑	50.23±5.23↑	51.42±5.54↑	64.83±5.21↑
alizadeh-2000-v2	83.18±5.17	81.24±4.96	67.81±4.24↑	65.87±4.52↑	62.12±3.21↑
bredel-2005	69.43±7.21	66.36±7.48↑	62.01±4.26↑	63.27±4.03↑	59.35±3.69↑
chen-2002	87.72±2.75	82.28±2.93↑	78.86±4.62↑	79.06±4.64↑	88.72±2.45
garber-2001	71.55±4.93	63.62±4.87↑	52.29±4.93↑	53.62±4.78↑	51.17±4.38↑
liang-2005	79.68±6.08	78.42±5.09	75.73±2.84↑	74.65±2.76↑	75.60±3.25
tomlins-2006	65.42±6.03	52.61±5.12↑	30.78±1.42↑	30.78±1.42↑	60.78±3.33↑
tomlins-2006-v2	63.14±3.27	53.78±4.81↑	34.79±1.42↑	35.64±1.38↑	58.72±3.61+

Table 6.8 shows the computation time to classify new instances in the application process of the proposed method and the benchmark methods. It is clear from Table 6.8 that although the proposed method is slightly more expensive than kNN-based imputation, but it is much faster than the ensemble methods.

In summary, the proposed methods are still able to produce dramatic improvement in efficiency and better accuracy on large datasets.

6.4.3.3 Evaluation of the Proposed Method on Specific Problem

In order to demonstrate how the proposed method works and its effectivenesses and efficiencies, we analysed carefully the proposed method on *Heart-h* using *C4.5*. The *Heart-h* dataset was chosen because it has the

Table 6.8: Time to classify instances in the application process on the gene expression datasets (millisecond).

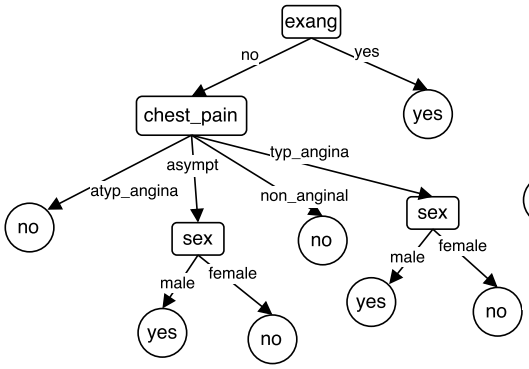
Dataset	NewMethod	kNNI	Ensemble[24]	Ensemble[185]	Ensemble[125]
alizadeh-2000-v1	4.4×10^{1}	9.3	1.4×10^{3}	1.3×10^{3}	1.4×10^{4}
alizadeh-2000-v2	9.9×10^{3}	2.9×10^{1}	1.1×10^{4}	1.0×10^{4}	1.2×10^{5}
bredel-2005	2.1×10^{2}	3.6×10^{1}	2.3×10^{3}	2.1×10^{3}	1.3×10^{5}
chen-2002	5.4	6.8	1.1×10^{2}	9.1×10^{1}	3.4×10^{2}
garber-2001	2.4×10^{3}	8.1×10^{1}	2.9×10^{4}	2.7×10^{4}	8.5×10^{5}
liang-2005	4.9×10^{1}	9.9	6.1×10^{2}	5.8×10^{2}	8.0×10^{3}
tomlins-2006	7.4×10^{3}	8.9	6.3×10^{4}	6.1×10^{4}	5.5×10^{5}
tomlins-2006-v2	8.3×10^{2}	3.7	1.4×10^{4}	1.2×10^{4}	1.2×10^{5}

largest percentage of incomplete instances (100%) compared to the other datasets. *C4.5* was chosen because decision trees generated by *C4.5* are straightforward to interpret.

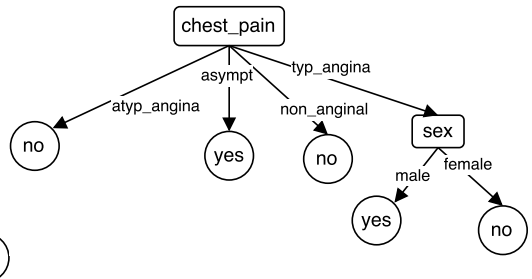
Hear-h describes the contents of the heart-disease collected by Hungarian Institute of Cardiology [3]. The dataset has 13 features: {*age*, *sex*, *chest_pain*, *trestbps*, *chol*, *fbs*, *restecg*, *thalach*, *exang*, *oldpeak*, *slope*, *ca*, *thal*}. The values of the 13 features are used to decide the diagnosis of heart disease shown in a class feature, where 0 indicates less than 50% diameter narrowing and 1 indicates more than 50% diameter narrowing. The original dataset has six incomplete features: {*chol*, *fbs*, *exang*, *slope*, *ca*, *thal* } and contains 15 missing patterns.

In the application process, *MICE* imputation is firstly used to transform the incomplete dataset into an imputed dataset. *CFS* is then applied on the imputed dataset, and it selects a subset of five features $\mathcal{SF} = \{$*sex*, *chest_pain*, *exang*, *oldpeak*, *slope*$\}$ and removes the other eight features. As a result, the original dataset reduced on $\mathcal{SF}$ has only two incomplete features {*exang*, *slope*} and it contains only three missing patterns: {*slope*}, {*exang*} and $\{\emptyset\}$. Therefore, feature selection helps to reduce the number

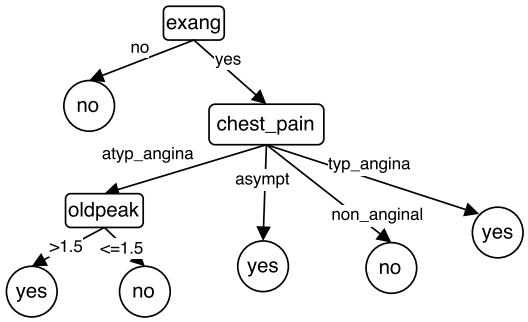
of incomplete features (from 6 to 2) and reduce the number of missing patterns (from 15 to 3).



(a) {*sex, chest_pain, exang, oldpeak*}



(b) $\{sex, chest\_pain, oldpeak, slope\}$



(c) $\{sex, chest\_pain, exang, oldpeak, slope\}$.

Figure 6.7: Decision trees constructed by using different feature subsets.

From the three missing patterns, the proposed method generates three complete patterns: {*sex, chest_pain, exang, oldpeak*}, {*sex, chest_pain, oldpeak, slope*} and {*sex, chest_pain, exang, oldpeak, slope*}. Figs. 6.7a, 6.7b and 6.7c show three decision trees generated by *C4.5* according to the three complete patterns. It is clear from the figures that the decision trees do not require all features, so they are tolerant with missing values because they can be applicable to more than one missing pattern. For example, the decision tree in Figure 6.7b is built on a dataset with four features (*sex, chest_pain, oldpeak, slope*), but it requires only two features (*sex, chest_pain*). As a result, the decision tree in Figure 6.7b is originally designed to classify incomplete

instances with only one missing value in feature *exang*; however, it does not require feature *slope*, so it also can be used to classify any incomplete instance with missing value in feature *slope*.

Table 6.9 shows some incomplete instances in the *Hear-h* dataset, which need to be classified. Table 6.10 presents instances in Table 6.9 reduced on the selected features $\mathcal{SF} = \{sex, chest\_pain, exang, oldpeak, slope\}$. As can be seen from Tables 6.9 and 6.10 that feature selection can help to reduce the number of missing values and reduce the number of incomplete instances. For example, the second and third instances in Table 6.9 are incomplete, but by only keeping the selected features, these instances become complete as shown in Table 6.10.

Table 6.9: Incomplete instances in the original *Hear-h* dataset.

N	**age**	**sex**	**chest_pain**	**trestbps**	**chol**	**fbs**	**restecg**	**thalach**	**exang**	**oldpeak**	**slope**	**ca**	**thal**
1	59	male	asympt	140	?	f	normal	140	no	0	?	0	?
2	46	male	asympt	120	277	f	normal	125	yes	1	flat	?	?
3	54	male	asympt	150	365	f	st_t_wave_abnormality	134	no	1	up	?	?
4	48	male	atyp_angina	100	?	f	normal	100	no	0	?	?	?
5	54	female	atyp_angina	140	309	?	st_t_wave_abnormality	140	no	0	?	?	?
6	48	female	atyp_angina	?	308	f	st_t_wave_abnormality	?	?	2	up	?	?

Table 6.10: Instances in Table 6.9 reduced on the selected features.

N	**age**	**chest_pain**	**exang**	**oldpeak**	**slope**
1	59	asympt	no	0	?
2	46	asympt	yes	1	flat
3	54	asympt	no	1	up
4	48	atyp_angina	no	0	?
5	54	atyp_angina	no	0	?
6	48	atyp_angina	?	2	up

In Table 6.10, the second and third instances are complete so they are quickly classified by all the decision trees without requiring time for ex-

ploring applicable classifiers. The first, fourth and fifth instances contain a missing value in feature *slope*. Although only the decision tree in Figure 6.7a is learned to classify these incomplete instances, the other decision trees in Figure 6.7b and 6.7c also can be used to classify the incomplete instances because they do not require feature *slope* thanks to implicitly performing feature selection of *C4.5*.

In summary, three powerful techniques—multiple imputation, feature selection and ensemble learning—make the proposed method effective and efficient.

6.5 Chapter Summary

The goal of this chapter was to develop a new ensemble approach to classification with incomplete data which can effectively and efficiently classify new incomplete instances without requiring any imputation. To achieve this goal, we integrated imputation, feature selection and ensemble learning to construct an ensemble of classifiers, each tailored to a known pattern of missing data. The imputation creates higher quality training data. The feature selection process reduces the number of missing patterns, which increases the speed of classification, and greatly increases the fraction of new instances that can be classified by the ensemble. Results show that the proposed method is more accurate, and faster than previous common methods for classification with incomplete data.

This chapter shows that integration of imputation with feature selection and ensemble learning can build more effective classifiers than a common approach to using imputation for classification with incomplete data. One reason is that feature selection helps reduce redundant and irrelevant features which in turn provides better training data for building classifiers. Another reason is that ensemble learning constructs multiple classifiers which are generally better than a single classifier generated by the common approach. The proposed method can also classify new incomplete

instances more efficiently than the common approach. The underlying reason is that the proposed method can directly classify new incomplete instances without requiring any time to estimate missing values.

This chapter also shows that the integration of imputation and feature selection with ensemble learning can build more effective classifiers than existing ensemble learning methods for classification with incomplete data. The key reason is that the proposed method uses imputation and feature selection to provide high quality training complete data. In contrast, existing ensemble methods often generate low quality training data, especially when incomplete data contains numerous missing values. The proposed method can also classify new instances faster than existing ensemble methods. The reason is that due to using feature selection to remove incomplete features, the proposed method generates a smaller number of classifiers and only needs to classify a smaller number of incomplete instances than existing the ensemble methods.

This chapter presented an an ensemble approach to directly classifying incomplete data by building a set of classifiers. Another approach such as C4.5 and CART is to build single classifiers which can work directly with incomplete data. The next chapter will describe using GP with interval functions to directly evolving classifiers for incomplete data.

Chapter 7

Directly Evolving Classifiers for Incomplete Data using Genetic Programming

7.1 Introduction

In classification tasks, discriminant functions are a popular method for representing classifiers. A discriminant function is a mathematical expression that represents a combination of the features of an instance which needs be classified. The value returned by the discriminant function determines the predicted class by using a single threshold (binary classification) or a set of thresholds (multiple classification) [42].

Genetic programming (GP) is an evolutionary technique which constructs computer programs [89]. The capability of GP to learn the definition of a function from examples makes it a very good choice for constructing discriminant functions for classification tasks. Therefore, GP has been widely used to construct discriminant functions for classification tasks [42].

Although GP has been successfully used to construct classifiers, it has been mainly applied to complete data. In order to use traditional GP

to construct classifiers for incomplete data, imputation methods are required to transform incomplete data into complete data before using GP. The combination of GP and a simple imputation method such as mean imputation often leads to big classification error. Therefore, to construct good classifiers, GP should be combined with sophisticated imputation methods such as multiple imputation by chained equations (MICE) [175] or multiple imputation for missing data using GP (GPMI) [159]. Unfortunately, sophisticated imputation methods like MICE and GPMI are only appropriate for batch imputation, and are too computationally intensive to estimate missing values for individual incomplete instances in the unseen data [157]. This chapter shows how to use GP to construct classifiers which can work directly with incomplete data without requiring an imputation method.

7.1.1 Chapter Goals

This chapter describes two methods that we use interval GP developed in Chapter 5 to construct effective and efficient classifiers able to work directly with incomplete data without requiring any imputation method. The first method construct a single classifier. The second method combines interval GP and ensemble learning to construct an ensemble of classifiers. Specially, this chapter will investigate:

1. How GP can directly construct effective and efficient classifiers for incomplete data without combining with imputation.

2. Whether a classifier constructed by interval GP can achieve better classification accuracy than a classifier constructed by combining imputation and traditional GP; and

3. Whether a classifier constructed by interval GP can achieve better classification accuracy than a classifier build by using a classification algorithm able to work directly with incomplete data; and

4. Whether a set of classifiers constructed by the combination of ensemble learning and interval GP can achieve better classification accuracy than a set of classifier constructed by the combination of ensemble learning, imputation and traditional GP; and

5. Whether a set of classifiers constructed by the combination of ensemble learning and interval GP can achieve better classification accuracy than a set of classifier build by using the combination of ensemble learning and classification algorithm able to work directly with incomplete data.

6. Why interval GP can construct effective and efficient classifiers for incomplete data.

7.1.2 Chapter Organisation

The rest of the paper is organised as follows. Section 7.2 discusses the proposed algorithms including interval GP to construct a single classifier for each incomplete data, and the combination of ensemble learning and interval GP to construct a set of classifiers for incomplete data. Section 7.3 outlines experiment design. Section 7.4 presents results and analysis. Section 7.5 draws conclusions.

7.2 Proposed Algorithms

This section presents two proposed algorithms. The first algorithm uses interval GP, namely IGP, to construct a single classifier for each incomplete dataset. The second algorithm combines ensemble learning with IGP, namely EIGP, to construct a set of classifiers for each incomplete dataset.

7.2.1 Interval Genetic Programming to Directly Construct a Single Classifier for Incomplete Data

The key idea of IGP is to use interval GP to directly construct a classifier for each incomplete data. With the constructed classifier, each missing value is replaced by an interval that expresses the uncertainty associated with the missing value.

The Figure 7.1 shows the main steps of IGP. The proposed algorithm consists of the training process and the application process. The training process uses interval GP to construct a single classifier for each incomplete dataset. The application process then uses the constructed classifier to classify new instances.

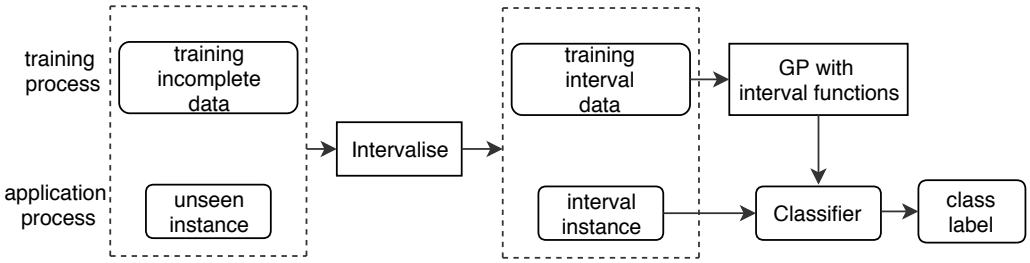


Figure 7.1: interval GP to construct a classifier for incomplete data

In the training process, firstly, all features in the training data are rescaled to build the training rescaled data. Next, the interval of feature is estimated, and then each feature value is replaced by an interval. Consequently, the training rescaled data is transformed into the training interval data. Finally, interval GP uses the training interval data to construct a classifier.

In the application process, an instance that needs to be classified is firstly rescaled to build a rescaled instance. After that, each feature value in the rescaled instance is replaced by an interval to construct an interval instance. Finally, the interval instance is classified by the constructed classifier.

The following subsections present main steps of the proposed algorithm: rescaling data, finding interval, interval functions and class boundary determination.

7.2.1.1 Rescaling Data

In order to construct a discriminant function for classification task, GP usually combines different features by using some operators. However, in a dataset, different features often have different scales. It is beneficial to make all features have the same scale before putting them into GP to build a classifier. Two common ways to rescale data are normalisation and standardization [28].

Normalization rescales all numeric features into the range of [0,1]. One formula for normalisation can be given below:

$$f_{new} = \frac{f_{ori} - f_{min}}{f_{max} - f_{min}}$$

where f_{min} and f_{max} are minimum value, maximum value of original feature *f*, respectively; f_{ori} and f_{new} are original value and new rescaled value of feature *f*, respectively.

Standardization rescales mean and variance of a feature into zero mean and unit variance, respectively. One formula for standardization can be given below:

$$f_{new} = \frac{f_{ori} - \mu_f}{\sigma_f}$$

where μ_f and σ_f are the mean and standard deviation of original feature *f*, and f_{ori} and f_{new} are original value and new rescaled value of feature *f*, respectively.

Each of these techniques has its drawbacks. If data contains outliers, normalization is likely to rescale the data to a very small interval. Stan-

dardizing data can lead to a poor result if the data has a very non-normal distribution [28].

7.2.1.2 Feature Interval and Interval Functions

As in Chapter 5, in order to make interval GP able to work directly with missing values, we need to find an interval for each incomplete feature, and then replace missing values for the feature by the interval. We used the method shown in Section 5.3.1 in page 111 to find the feature interval.

Along with transforming incomplete data to interval data, we also need to build interval functions to work with interval data. We used the four interval arithmetic operations shown in Session 5.3.2 in page 112.

7.2.1.3 Class Boundary Determination

The classifier outputs an interval [l,u], which must be assigned to a particular class. IGP uses a sequence of static boundaries ($T_1 < T_2 < ... < T_{n-1}$) to define n class regions [190]. It then determines the class by working out which which region the center of the interval ($mid = \frac{l+u}{2}$) fall in:

$$class = \begin{cases} class_1 : & mid \leq T_1 \\ class_2 : & T_1 < mid \leq T_2 \\ ... & \\ class_i : & T_{i-1} < mid \leq T_i \\ ... & \\ class_n : & mid > T_{n-1} \end{cases}$$

where $T_1, T_2, ..., T_{n-1}$ are pre-defined static class boundaries.

7.2.2 The Combination of Interval Genetic Programming and Ensemble Learning to Evolve a Set of Classifiers for Incomplete Data

The problem with IGP is that the output of a classifier constructed by IGP is an interval which can span more than one class boundary, but IGP determines a single class label by using the middle point of the output interval. This decision method does not select the highest probability class when the middle point belongs to one class region, but the biggest overlap with the output interval belongs to another class region. For example, in Figure 7.2, the middle point belongs to *class 2*, but the highest overlap with the interval output is the region of *class 1*. The decision method also makes an unjustified decision when there exist two or more class regions with the same or similar overlap with the interval output. For example, in Figure 7.3, *class 2*, *class 3* and *class 4* have the same overlap with the interval, but IGP only outputs *class 3*.

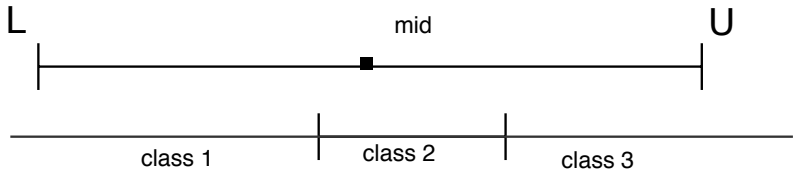


Figure 7.2: A problem with IGP because of using the middle of interval output to decide a final class.

In order to overcome the limitations of IGP, the second proposed algorithm, EIGP, combines ensemble methods and IGP to construct a set of classifiers for incomplete data. To construct a set of classifiers, firstly, training data is put into a resample procedure such as in bagging/boosting to build a set of training resampled data. After that, each training resampled data is used by IGP to build a single a classifier. As a result, a set of classi-

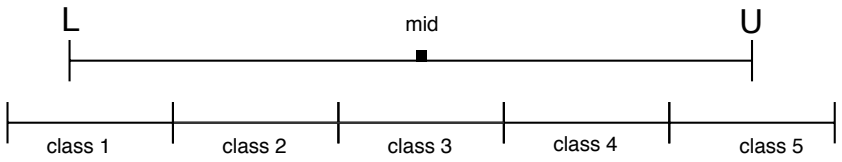


Figure 7.3: A problem with IGP because of building only one classifier.

fiers is generated. When a new instance needs be classified, each classifier in EIGP estimates the probability of the instance belonging to each class instead of determining a single class for the instance as IGP. After that, the final class of the instance is the class which achieves the highest total probability over all classifiers.

Assuming $[l, u]$ is the output of a classifier constructed by GP with interval function, and $[T_{i-1}, T_i)$ is the class region of the i^{th} class label, then the probability of the i^{th} class label is chosen by the classifier is defined as followed:

$$prob([l,u] \in class_i) = \begin{cases} 0: & \text{if } u < T_{i-1} \quad or \quad Ti \leq l \\ \frac{min(u,T_i) - max(l,T_{i-1})}{u-l} : & otherwise \end{cases}$$

where $T_1, T_2, ..., T_{n-1}$ are the pre-defined static class boundaries.

7.3 Design of Experiments

This section presents the comparison method between the proposed methods with benchmark methods. It also shows datasets and parameter settings which are used in the experiments.

7.3.1 The Comparison Methods

The experiments are designed to achieve two main objectives. The first objective is to evaluate the impact of IGP on evolving a single classifier for each incomplete dataset. The second objective is to evaluate the impact of ensemble learning on IGP for evolving a set of classifiers for each incomplete dataset.

To achieve the first objective, the experiments are designed to compare IGP with two other common classification methods for classifying incomplete datasets. The Figure 7.1 shows the process of IGP for building a single classifier for each incomplete dataset. The Figure 7.4 shows one common classification method for classifying incomplete datasets by combining an imputation method and GP for evolving a single classifier for each incomplete dataset. The Figure 7.5 shows other common classification method for classifying incomplete datasets by using a classifier which is able to work directly with incomplete datasets. In the first setup, as shown in Figure 7.1, the training incomplete data is used by IGP to build a classifier which is then used to classify unseen incomplete instances. In the second setup, as shown in Figure 7.4, an imputation method is firstly used to transfer the training incomplete data and unseen incomplete instances into the training imputed data and unseen imputed instances, respectively. After that, the training imputed data is used by GP to construct a classifier which is then used to classify unseen imputed instances. In the third setup, as shown in Figure 7.5, the training incomplete data is directly used by a classification algorithm which is able to classify incomplete datasets to build a classifier which is then used to classify unseen incomplete instances.

To achieve the second objective, in the three setups, the training incomplete dataset in Figures 7.1 and 7.5 and the training imputed dataset in Figure 7.4 are put into the bagging method to generate a set of training datasets. After that, each training dataset is used to build a classifier. Consequently, with the set of training datasets, a set of classifiers is generated.

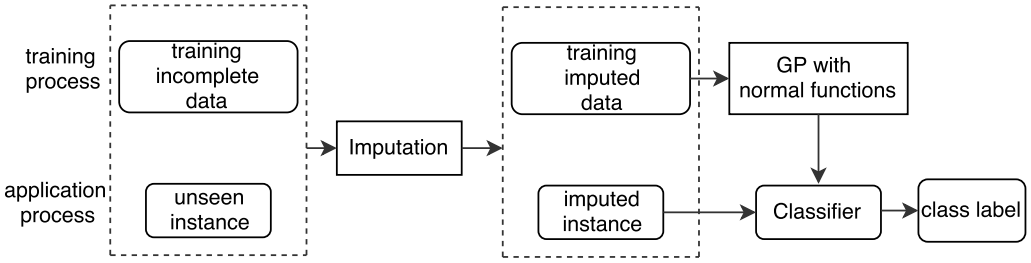


Figure 7.4: Classification with incomplete data by combining imputation and GP

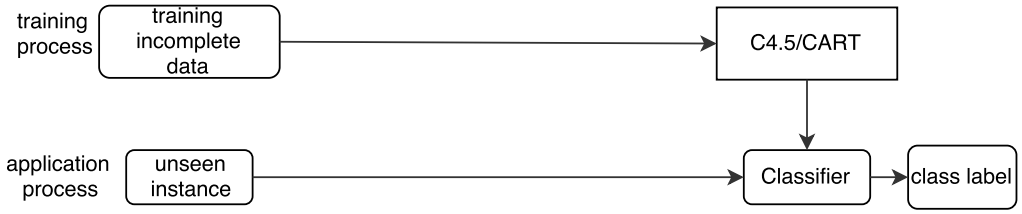


Figure 7.5: Classification with incomplete data by using a classifier able to work directly with incomplete data.

Finally, the set of classifiers is used to classify the testing dataset.

7.3.2 Datasets and Parameter Settings

The experiments compare the proposed methods with the other methods on 12 datasets which were used in Chapter 5. The main characteristics of these datasets are shown in Table 1.1. These datasets only contain numeric features because classifiers which are evolved by GP are the mathematical expressions. As in previous chapters, ten-fold cross-validation is used to divide these datasets into training sets and test sets.

The first seven of these datasets contain "natural" missing values. To test the proposed methods on datasets with various levels of missing values, from the last five complete datasets, some complete values are randomly removed to generate "artificial" incomplete datasets. Six levels of

missing values: 5%, 10%, 15%, 20%, 25% and 30% are introduced into each complete dataset. The same as Chapter 5, missing values are introduced into important features which are selected by CFS [66].

As in Chapter 5, ECJ package [110] is used to implement GP. The parameters of GP in imputation methods combined with GP are similar to the parameters of GP in the proposed methods, except using normal function set and normal terminal set instead of using interval function set and interval terminal set. The parameters of GP is set the same as in Chapter 5 shown in Table 5.1 in page 117. Bagging is used to generate a set of classifiers. The number of classifiers in ensemble learning evolving by GP is also 25 as suggested by [120].

The experiments use two imputation methods to combine with GP are kNN-based imputation and MICE. As in previous Chapters, with kNN-based imputation, the number of k is set 1; MICE in [22] with random forest is used to estimate missing values for incomplete features. Each incomplete feature is repeatedly regressed 20 times on other features. With each incomplete dataset, the multiple imputation is repeatedly done 5 times to generate 5 imputed datasets before averaging them to generate a final imputed dataset.

The proposed methods are compared with two decision trees which can directly classify incomplete data: C4.5 [129] and CART [105]. WEKA [65] with default parameter settings is used to implement the classification algorithms. Following the suggestion in [120], the number of classifiers in ensemble learning evolving by C4.5 and CART is set 25.

7.4 Results and Discussions

This section presents the comparison of the proposed methods with benchmark methods and performs analysis.

7.4.1 The comparison between IGP with other single methods

Table 7.1 presents the average of classification accuracy along with standard deviation of IGP and the other single methods on the first seven datasets. The average of classification accuracy in Table 7.1 is calculated on accuracies of each method on 30 times performing ten-fold cross-validation on each dataset. Table 7.2 shows the average of classification accuracy along with standard deviation of IGP and the other single methods on the last five datasets with six levels of missing values. With each dataset and each level of missing values, the averages of classification accuracy in Table 7.2 is calculated on accuracies of each method on 30 generated incomplete datasets by using ten-fold cross-validation on each incomplete dataset.

Table 7.1: The comparison between IGP with the other single methods on natural incomplete datasets

Dataset	IGP	kNNGP	MICEGP	C4.5	CART
Ban	**69.97±1.23**	69.89±1.0	69.91±0.85	68.45±1.88↓	65.99±1.69↓
Bre	**96.53±0.28**	95.66±0.35↓	95.75±0.39↓	94.83±0.46↓	94.42±0.44↓
Cle	**58.12±1.50**	56.56±1.35↓	57.27±1.16↓	54.56±2.10↓	56.32±1.37↓
Hep	**81.05±1.77**	79.72±2.30↓	80.36±2.14	79.21±1.75↓	77.47±1.45↓
Mam	**83.13±0.35**	80.07±0.82↓	80.29±0.92↓	82.12±0.33	82.12±0.50
Mar	30.74±0.47	30.77±0.52	30.57±0.66	30.91±0.45	**33.57±0.35**↑
Ozo	**97.12±1.42**	96.91±0.13	96.89±0.11	96.25±0.27↓	97.09±0.07

In Table 7.1 and Table 7.2, "IGP" refers to the first proposed method, as shown in Figure 7.1. "kNNGP" and "MICEGP" refer to the second experimental setup, as shown in Figure 7.4, combining GP with kNN-based imputation and MICE, respectively. "C4.5" and "CART" refer to the third experimental setup as shown in Figure 7.5 using *C4.5* and *CART* as classi-

Table 7.2: The comparison between IGP with the other single methods on each artificial incomplete dataset.

Dataset	Missing values (%)	IGP	kNNGP	MICEGP	C4.5	CART
Bal	5	**98.98±0.41**	98.79±0.41	98.92±0.40	78.95±0.77↓	78.51±0.84↓
	10	**97.75±0.64**	97.70±0.62	97.41±0.57	78.66±0.86↓	78.14±0.80↓
	15	**97.10±0.60**	96.93±0.72	96.73±0.62	78.23±0.94↓	77.95±0.78↓
	20	**95.81±0.53**	95.32±0.73	95.59±0.50	77.69±0.87↓	77.31±0.95↓
	25	**94.95±0.76**	94.22±0.90	94.65±0.86	77.49±1.10↓	76.82±1.09↓
	30	**93.70 ±0.71**	92.87±0.88	93.41±0.81	77.02±0.96↓	76.37±0.96↓
Dia	5	**74.89±0.88**	67.37±0.87↓	67.61±0.79↓	74.86±0.93	74.52±0.80
	10	74.20±0.79	67.15±1.00↓	67.36±0.95↓	**74.81±0.72**	73.71±1.26
	15	73.42±0.68	66.87±0.63↓	67.13±0.85↓	**74.34±1.22**	73.08±1.09
	20	72.61±0.98	66.81±0.80↓	66.93±1.04↓	**72.62±1.21**	72.59±1.21
	25	72.08±1.22	66.61±0.61↓	66.67±0.95↓	**72.48±0.96**	71.64±1.24
	30	70.97±1.16	66.75±0.86↓	67.13±1.01↓	**72.24±1.06**↑	70.88±1.35
Iri	5	**95.06±0.93**	93.42±1.58↓	93.77±1.06↓	91.33±1.24↓	91.17±1.27↓
	10	**95.11±0.93**	92.11±1.90↓	93.42±1.35↓	91.17±1.51↓	91.11±1.46↓
	15	**94.19±1.37**	90.06±2.22↓	92.11±1.73↓	90.62±1.73↓	90.06±1.63↓
	20	**93.91±1.48**	87.91±2.34↓	91.57±1.80↓	89.20±1.61↓	89.86±1.81↓
	25	**93.51±1.58**	86.28±2.43↓	90.24±2.26↓	88.62±1.95↓	88.57±2.33↓
	30	**91.33±2.2**	84.91±2.83↓	87.60±2.62↓	87.24±2.20↓	87.62±1.87↓
Liv	5	67.98±1.95	67.58±2.03	**68.20±2.24**	62.54±2.21↓	64.23±2.28↓
	10	67.87±2.01	67.58±2.02	**68.21±2.25**	62.46±2.12↓	64.28±2.21↓
	15	67.48±1.89	67.08±2.02	**67.70±2.31**	62.06±2.23↓	63.77±2.21↓
	20	66.97±1.97	66.58±2.00	**67.20±2.27**	61.56±2.15↓	63.27±2.20↓
	25	65.86±1.86	65.47±1.98	**66.19±1.97**	60.46±2.34↓	62.16±2.19↓
	30	64.98±1.96	65.47±2.01	**66.19±2.25**	60.46±2.13↓	62.37±2.25↓
Sta	5	**83.04±1.31**	78.50±1.63↓	78.46±1.55↓	81.39±1.25↓	77.49±1.49↓
	10	**81.81±1.70**	77.60±1.89↓	78.14±1.56↓	80.04±2.05↓	76.38±1.54↓
	15	**81.45±1.38**	76.92±1.40↓	77.71±1.93↓	78.77±1.67↓	75.70±1.51↓
	20	**80.92±1.83**	76.46±2.04↓	77.32±2.13↓	77.06±2.19↓	75.04±1.75↓
	25	**80.22±1.60**	75.80±1.66↓	76.80±2.01↓	76.02±1.96↓	74.12±1.96↓
	30	**79.14±1.51**	74.67±1.82↓	75.99±1.98↓	75.35±1.76↓	73.48±1.65↓

fication algorithm, respectively.

As in previous chapter, for each incomplete dataset, *Friedman* test [53], which is a non-parametric test for multiple comparisons, is used to statistical test the null hypothesis in classification accuracies. The test shows that for all tasks, there is a significant difference in classification accuracies for the five methods, so null hypothesis rejected. Therefore, a post hoc multiple comparisons test using the *Holm* method [72] is used to determine the statistically significant differences between group means. "↑" means that the benchmark method is significantly more accurate than the proposed method; "↓" means that the benchmark method is significantly worse than the proposed method.

It can be seen clearly from Table 7.1 that with the "natural" incomplete datasets, IGP is significantly more accurate than the other methods in most cases. IGP is not significantly worse than kNNGP, MICEGP and *C4.5* in any case, and it is only significantly worse than *CART* in *Marketing* dataset.

It can be seen clearly from Table 7.2 that with "artificial" incomplete datasets, in most cases, IGP are also significantly more accurate than the other benchmark methods. Except *Balance* and *Liver* datasets with the same accuracy, the classification accuracies of IGP are significantly better than both kNNGP and MICEGP in three other datasets with all levels of missing values. Moreover, for all tasks except *Diabetes*, the classification accuracies of IGP are significantly better than both *C4.5* and *CART*. For *Diabetes* task, the classification accuracies of IGP are not statistically different from both C4.5 and CART with the first five levels of missing values, and significantly worse than C4.5 with 30% of missing values.

In summary, with both natural and artificial incomplete datasets, in most cases, IGP are generally significantly more accurate than the other benchmark methods.

7.4.2 The Comparison between EIGP with Other Ensemble Methods

Table 7.3 presents the average of classification accuracy along with standard deviation of *EIGP* and the other ensemble methods on the first seven "natural" incomplete datasets. It is clear from Table 7.3 that with the "natural" incomplete datasets, the classification accuracies of *EIGP* are also significantly more accurate than the other benchmark methods in most cases.

Table 7.3: The comparison between bagging IGP with the other bagging methods on natural incomplete datasets.

Dataset	EnIGP	EnkNNGP	EnMICEGP	EnC4.5	EnCART
Ban	**72.15±0.84**	70.53±0.77↓	70.74±0.62↓	71.18±1.26↓	70.10±1.85↓
Bre	**96.69±0.21**	96.22±0.28↓	96.30±0.32↓	95.86±0.31↓	95.84±0.35↓
Cle	**59.85±1.17**	57.88±0.84↓	57.69±1.27↓	57.08±1.29↓	57.73±1.64↓
Hep	**82.39±1.69**	82.47±1.84	82.85±1.47	81.56±1.63↓	81.27±1.77↓
Mam	**83.44±0.27**	80.77±0.54↓	80.77±0.59↓	82.68±0.50↓	81.63±0.54↓
Mar	31.40±0.28	30.83±0.39↓	30.89±0.37↓	**31.56±0.42**	31.26±0.41
Ozo	97.09±0.06	**97.12±0.01**	**97.12±0.01**	97.05±0.08	97.07±0.09

Table 7.4 shows the average of classification accuracy along with standard deviation of *EIGP* and the other ensemble methods on the last five datasets with six levels of missing values. It is also clear from Table 7.4 that with "artificial" missing values, in almost all cases, the classification accuracies of *EIGP* are significantly better than the other methods.

To confirm whether the proposed methods are really significantly better than the other methods, we perform the *Friedman* test on the average of accuracies of all the algorithms in all incomplete datasets as shown in Table 7.1, Table 7.2, Table 7.3 and Table 7.4. The test indicates that there is a significant difference in classification accuracies in the ten methods, so

Table 7.4: The comparison between bagging IGP with the other bagging methods on artificial incomplete datasets.

Dataset	Missing values (%)	EnIGP	EnkNNGP	EnMICEGP	EnC4.5	EnCART
Bal	5	**98.98±0.41**	98.79±0.41	98.92±0.40	82.95±0.77↓	82.51±0.84↓
	10	**97.75±0.64**	97.70±0.62	97.41±0.57	82.66±0.86↓	82.14±0.80↓
	15	**97.10±0.60**	96.93±0.72	96.73±0.62	82.23±0.94↓	81.95±0.78↓
	20	**95.81±0.53**	95.32±0.73	95.59±0.50	81.69±0.87↓	81.31±0.95↓
	25	**94.95±0.76**	94.22±0.90	94.65±0.86	81.49±1.10↓	80.82±1.09↓
	30	**93.70±0.71**	92.87±0.88	93.41±0.81	81.02±0.96↓	80.37±0.96
Dia	5	**74.89±0.88**	67.37±0.87	67.61±0.79	74.86±0.93	74.52±0.80
	10	74.20±0.79	67.15±1.00	67.36±0.95	**74.81±0.72**	73.71±1.26
	15	73.42±0.68	66.87±0.63	67.13±0.85	**74.34±1.22**	73.08±1.09
	20	72.61±0.98	66.81±0.80	66.93±1.04	**73.62±1.21**	72.37±1.21
	25	72.08±1.22	66.61±0.61	66.67±0.95	**73.48±0.96**↑	71.64±1.24
	30	70.97±1.16	66.75±0.86	67.13±1.01	**72.24±1.06**↑	70.88±1.35
Iri	5	**95.06±0.93**	93.42±1.58↓	93.77±1.06↓	93.33±1.24↓	94.17±1.27↓
	10	**95.11±0.93**	92.11±1.90↓	93.42±1.35↓	93.17±1.51↓	94.11±1.46↓
	15	**94.19±1.37**	90.06±2.22↓	92.11±1.73↓	92.62±1.73↓	93.06±1.63↓
	20	**93.91±1.48**	87.91±2.34↓	91.57±1.80↓	91.20±1.61↓	91.86±1.81↓
	25	**93.51±1.58**	86.28±2.43↓	90.24±2.26↓	90.62±1.95↓	90.57±2.33↓
	30	**91.33±2.20**	84.91±2.83↓	87.60±2.62↓	88.24±2.20↓	88.62±1.87↓
Liv	5	**69.53±1.62**	69.24±1.49	69.47±1.62	69.12±1.79	68.56±1.72
	10	**69.05±1.73**	68.73±1.54	68.96±1.43	68.59±1.93	68.07±1.70
	15	68.54±1.63	68.21±1.63	**68.65±1.53**	68.10±1.83	67.58±1.73
	20	68.04±1.72	67.72±1.55	**68.06±1.42**	67.59±1.92	67.07±1.69
	25	67.53±1.66	67.22±1.64	**67.66±1.57**	67.12±1.82	66.59±1.72
	30	67.05±1.74	66.73±1.56	**67.09±1.43**	66.58±1.97	66.03±1.65
Sta	5	**83.04±1.31**	78.50±1.63↓	78.46±1.55↓	81.39±1.25↓	80.49±1.49↓
	10	**81.81±1.70**	77.60±1.89↓	78.14±1.56↓	80.04±2.05↓	79.38±1.54↓
	15	**81.45±1.38**	76.92±1.40↓	77.71±1.93↓	78.77±1.67↓	78.70±1.51↓
	20	**80.92±1.83**	76.46±2.04↓	77.32±2.13↓	79.06±2.19↓	78.04±1.75↓
	25	**80.22±1.60**	75.80±1.66↓	76.80±2.01↓	78.02±1.96↓	77.12±1.96↓
	30	**79.14±1.51**	74.67±1.82↓	75.99±1.98↓	77.35±1.76↓	76.48±1.65↓

null hypothesis rejected. Therefore, the *Holm* method [72] is used to determine the statistically significant differences between pairs of algorithms in all the incomplete datasets.

Table 7.5 shows the significant comparison between some pairs of algorithms. As demonstrated from Table 7.5, in all the incomplete datasets, IGP is significantly more accurate than the other single methods, and *EIGP* is also significantly more accurate than the other ensemble methods. As also can be seen from Table 7.5 that each ensemble method is significantly better than the corresponding single method. Table 7.6 shows the ranking of the algorithms using the Friedman test *(smaller means better)*. As is evident from Table 7.6 that the proposed methods are the best algorithms.

Table 7.5: The significant comparison between some methods on all incomplete datasets *(Holm's procedure rejects those hypotheses that have a p-value <= 0.00294)*.

Algorithms	**Holm**
IGP vs. kNNGP	0.0012
IGP vs. CART	0.0013
IGP vs. MICEGP	0.0014
IGP vs. C4.5	0.0015
EIGP vs. EkNNGP	0.0013
EIGP vs. ECART	0.0016
EIGP vs. EMICEGP	0.0019
EIGP vs. EC4.5	0.0023
IGP vs. EIGP	0.0028
C4.5 vs. EC4.5	0.0016
MICEGP vs. EMICEGP	0.0018
CART vs. ECART	0.0020
kNNGP vs. EkNNGP	0.0024

In summary, with both "natural" and "artificial" incomplete datasets, in most cases, the proposed method is generally significantly better than the other benchmark methods.

Table 7.6: The ranking of the methods on all incomplete datasets using Friedman test.

EIGP	IGP	EC4.5	EMICEGP	ECART	EkNNGP	C4.5	MICEGP	CART	kNNGP
1.214	3.414	3.757	4.300	5.000	5.971	7.385	7.628	7.857	8.471

7.4.3 Further Analysis

To investigate how the proposed methods work, we further analysed trees generated by using interval GP on *Diabetes* dataset. *Diabetes* is chosen because generated trees in this case are small enough to be analysed by humans. *Diabetes* is a binary classification problem (two classes: *tested_negative*, *tested_positive*), and it has eight features $\{f_1,f_2,..,f_8\}$ and we put 10% missing values in four features $\{f_2, f_6, f_7, f_8\}$, which are selected by CFS [66].

In training process, the training data is firstly normalised. After that, the interval of each feature is estimated from complete values of the feature. Table 7.7 shows the interval of each feature in *Diabetes*. Subsequently, in the training data, each complete field is replaced by an interval where lower bound and upper bound are set the complete value, and each missing field is replaced by an interval of the feature containing the field. For example, from Table 7.7, if a field in f_2 is missing, it is replaced by [-2.09, 2.40]. Finally, the training data is put into interval GP to build a classifier. Figure 7.6 show a tree generated by interval GP on *Diabetes*.

Table 7.7: Interval of features in *Diabetes* dataset.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
lower	-1.14	-2.09	-3.44	-1.28	-0.68	-3.96	-1.17	-1.04
upper	2.98	2.40	2.03	2.46	5.07	3.13	5.07	3.12

In application process, when an instance needs to be classified, it is firstly normalised, and then intervalised. For example, to classify an instance: *(5, 99, 9.74, 27, 0, ?, ?, ?)* (? means missing value), complete values are firstly normalised: *(0.34, -0.68, 0.25, 0.41, -0.69, ?, ?, ?)*. After that, each

field is replaced by an interval: *([0.34, 0.34], [-0.68, -0.68], [0.25, 0.25], [0.41, 0.41], [-0.69, -0.69], [-3.96, 3.13], [-1.17, 5.07], [-1.04, 3.12])* (each missing field is replaced by an interval of corresponding feature). The interval instance is then put into a classifier such as shown in Figure 7.6 to result in an interval *[-9.42, 4.77]*. Subsequently, the middle point of the interval is calculated: $\frac{-9.42+4.77}{2} = -2.32$; therefore, the instance is classified into *tested_negative* class.

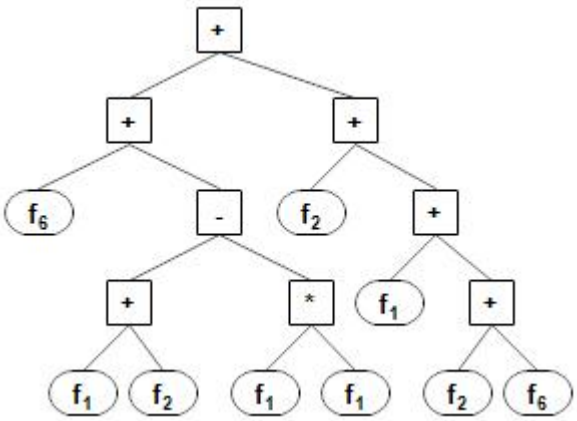


Figure 7.6: The first tree generated by GP with interval. functions

In the second proposed algorithm, namely EIGP, instead of constructing one classifier, a set of classifier is contructed and combined to classify new instances. For example, Figure 7.7 shows another classifier generated by using interval GP on *Diabetes*. When the instance is put into the second classifier, the output is *[-7.28, -0.18]* . Instead of using middle point to decide class label, the proportion of the instance belonging to each class. For example, with the output *[-9.42, 4.77]*, $\frac{|-9.42|}{|-9.42|+4.77} * 100 = 66.38\%$ of the instance belongs *tested_negative* class, and $\frac{|4.77|}{|-9.42|+4.77} * 100 = 33.62\%$ belonging *tested_positive* class. With the output *[-7.28, -0.18]*, both lower bound and upper bound are less than zero; therefore, 100% of the instance belongs *tested_negative* class. As a result, if the two classifiers are used to classify the instance, on average, $\frac{66.38+100}{2} = 83.19\%$ of the in-

stance belongs *tested_negative* class, and $\frac{33.62+0}{2} = 16.81\%$ of the instance belongs *tested_positive* class. Consequently, the instance is classified to *tested_negative* class.

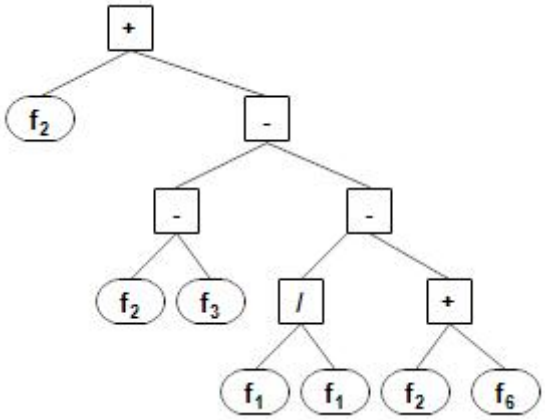


Figure 7.7: The second tree generated by interval GP.

In summary, replacing missing values with intervals help reflect very well the uncertainty of incomplete data. Moreover, the combination of a set of classifiers generated by interval GP can improve the accuracy for classification with incomplete data.

7.5 Chapter Summary

The goal of this chapter was to develop GP-based classifiers that can directly classify new incomplete instances effectively and efficiently without using any imputation method. To achieve this goal, GP with a set of interval functions as a function set is used to directly evolve a single classifier for incomplete data, where each missing value is replaced by an interval. Interval GP is also combined with ensemble learning to directly evolve a set of classifiers for each incomplete data.

This chapter shows that interval GP can evolve more effective classifiers than the combination of GP and imputation methods. One reason is

that an interval can reflect better the uncertainty of a missing value than a single value generated by imputation methods. As shown in Chapter 5, interval GP is more efficient than the combination of GP and imputation methods; therefore, the proposed methods can classify new incomplete instances faster than the combination of GP and imputation methods.

This paper also shows that interval GP can evolve more effective classifiers than classification algorithms such as C4.5/CART that are able to work directly with incomplete data. The key reason is that GP has capability of evolving more accurate classifiers than decision trees in many cases.

Chapter 8

Conclusions

This thesis focuses on classification with incomplete data. The overall goal was to improve the effectiveness and efficiency of classification with incomplete data by using evolutionary machine learning techniques. This goal was successfully achieved by developing a number of new methods based on evolutionary machine learning techniques for feature selection, feature construction, clustering, ensemble learning and constructing classifiers in order to build effective and efficient classifiers for incomplete data. The proposed methods were evaluated and compared with existing methods on a range of incomplete classification tasks of varying difficulty. Results show that the proposed methods are able to construct effective and efficient classifies for incomplete data.

The rest of this chapter first presents conclusions for each of research objective of this thesis. After that, it provides the main findings and highlights from each individual chapter. Finally, it discusses potential research areas for future work.

8.1 Achieved Objectives

This thesis has achieved the following objectives:

- *Develop new methods to improve the effectiveness and efficiency of using imputation for classification with incomplete data by integrating clustering and feature selection with imputation.*

  One method uses clustering to reduce the training set of imputed instances into a smaller, but representative, subset which is then used for estimating missing values in the application process. The second method uses feature selection to remove redundant and irrelevant features from the training imputed data to provide better training data for building classifiers and imputing missing values. Integrating clustering with imputation can speed up the imputation of missing values, and still obtain comparable accuracy to using only imputation. Integrating feature selection with imputation also can reduce the computation time of imputation, and improve classification accuracy compared to using only imputation. Integrating both feature selection and clustering with imputation not only improves the effectiveness, but also improves the efficiency of using imputation for classification with incomplete data. The proposed methods also outperform recent approaches to using clustering and feature selection for classification with incomplete data.

- *Develop new wrapper-based feature selection methods to improve classifiers that are able to directly work with incomplete data.*

  The first proposed method uses wrapper-based feature selection for a single classifier, where the feature selection is used to remove redundant and irrelevant features of the original training data. The second proposed method uses wrapper-based feature selection for ensemble classifiers, where the feature selection is used to remove redundant and irrelevant features of resampled datasets generated by bagging/boosting. In the wrapper-based feature selection methods, a classifier that can directly classify incomplete data without requiring any imputation is used to evaluate the quality of feature subsets.

By removing redundant/irrelevant features from the original training data, the wrapper-based feature selection for a single classifier can improve the accuracy and reduce the complexity of the classifier. By removing redundant and irrelevant features from resampled datasets generated by bagging/boosting, the wrapper-based feature selection for the ensemble classifiers can generate more accurate and less complex classifiers than classifiers generated by the original bagging/boosting. The proposed methods also outperform using imputation for classification with incomplete data.

- *Develop a new feature construction method to evolve multiple complete features from incomplete data for classifiers that are able to directly work with incomplete data.*

  The proposed method uses GP with interval functions as a function set to evolve multiple complete features, where a constructed feature is evolved for each class to maximise the purity of the class. The proposed method replaces each missing value for a feature with the interval of possible values for the feature to better reflect the uncertainty of missing data. By constructing multiple complete features from incomplete data, the proposed method can improve the accuracy and reduce the complexity of the classifiers. It can achieve better accuracy and less complex classifiers than using imputation to estimate missing values before constructing new features.

- *Develop an effective and efficient ensemble method to classification with incomplete data by integrating imputation, feature selection and ensemble learning.*

  The method uses an imputation method to estimate missing values in the original training set to generate an imputed dataset which is complete. A feature selection method is then used to improve the imputed dataset by removing redundant and irrelevant features. After that, the training process identifies all missing patterns and then

builds a set of classifiers, one classifier for each missing pattern. To classify a new instance, the algorithm starts by removing the redundant and irrelevant features, and then searches for classifiers applicable to the instance. Subsequently, each applicable classifier is used to classify the instance, and the algorithm returns a class by taking a majority vote of the applicable classifiers' predictions. By integrating imputation, feature selection and ensemble learning to build a set of classifiers and avoiding the need for imputation in the application process, the proposed method is more effective and more efficient than other common methods for classification with incomplete data.

- *Develop new methods to use interval GP for directly evolving classifiers for incomplete data without requiring any imputation method.*

  The first proposed method uses interval GP to evolve a single classifier for incomplete data. The second proposed method combines interval GP with ensemble methods to build a set of classifiers for incomplete data. In the proposed methods, GP with a set of interval functions is used evolve classifiers, where each feature missing value is substituted by an interval. By using interval functions as the function set in GP, the proposed method can evolve more accurate classifiers than using imputation to estimate missing values before using traditional GP to evolve classifiers. The proposed methods can also achieve better accuracy than existing classifiers that are able to directly work with incomplete data.

8.2 Main Conclusions

This section discuss the contributions from the five main chapters (Chapter 3 to Chapter 7)

8.2.1 Integrating Clustering, Feature Selection and Imputation

Chapter 3 presented new methods to improve imputation for classification with incomplete data by using clustering and feature selection.

Integrating Clustering with Imputation

It was found that using clustering to produce a smaller set of representative training data to perform imputation can dramatically reduce the computation time of the application process, and still maintain the classification accuracy.

The computation time of many imputation methods such as kNN-based imputation strongly depends on the number of instances which are used to impute missing values. The number of instances in clustered data is much smaller than the number of instances in the original data. Therefore, using clustered data with a smaller number of instances to estimate missing values for unseen incomplete instances can speed up the application process.

Using the clustered data to estimate missing values might be expected to slightly reduce the accuracy of imputation in the application process because there is less information. However, this may be counteracted by improved quality of the imputation if clustering is able to identify more representative data points. Results show that integrating clustering with imputation is generally as accurate as using only imputation.

Integrating Feature Selection with Imputation

It was found that using feature selection to remove redundant and irrelevant features from the imputed training data for building classifiers and performing imputation can not only improve the classification accuracy but also reduce the computation time of the application process in classification with incomplete data.

Feature selection improves the quality of training data by eliminating redundant and irrelevant features, which in turn helps to construct more accurate classifiers. By removing redundant and irrelevant features, feature selection also reduces the number of missing values in testing data, which in turn also improves the classification accuracy.

The computation time of many imputation methods such as MICE strongly depend on the number of features in the data which is used to impute missing values. Due to using the selected data with a smaller number of features to estimate missing values for unseen incomplete instances, the integration of feature selection with imputation can reduce the computation time of the imputation in the application process. Moreover, by removing redundant and irrelevant features from unseen instances, feature selection is able to reduce the number of incomplete instances in the application process which then reduces the computation time of the application process.

Integrating both Feature Selection and Clustering with Imputation

It was found that integrating both feature selection and clustering with imputation can further speed up imputation in the application process and still achieve better accuracy than using only imputation. The reason is that the innovations of the two methods are complementary. Both feature selection and clustering help reduce the computation time of imputation in the application process but in different and complementary ways. By removing irrelevant and redundant features, feature selection reduces the number of features in the data; clustering reduces the number of instances in the data. Both of these factors reduce the computation time of imputation in the application process. Moreover, due to removing redundant and irrelevant features, feature selection not only improves classification accuracy, but also provides better data for clustering which produces better clusters and therefore better representative instances, which then further helps maintain classification accuracy.

8.2.2 Wrapper-based Feature Selection for Classification with Incomplete

Chapter 4 presented wrapper-based feature selection methods to improve classifiers such as C4.5 and CART that can directly work with incomplete data.

Wrapper-based Feature Selection for a Single Classifier

This thesis showed that a wrapper-based feature selection method can improve the accuracy and reduce the complexity of a classifier able to directly classify incomplete data. One reason is that although classification algorithms such as C4.5 and CART can directly work with incomplete data, they often generate inaccurate and more complex classifiers, especially when data contains a large number of missing values. By eliminating redundant and irrelevant features, feature selection helps to generate more accurate and simpler classifiers. Moreover, decision tree algorithms such as C4.5 tend to select complete features and ignore incomplete features to build classifiers. However the bias of selecting complete features to build decision trees is not always good. Therefore, by considering different feature subsets, the feature selection is able to reduce the bias towards selecting complete features to build classifiers.

Wrapper-based Feature Selection for Ensemble Classifiers

This thesis showed that a wrapper-based feature selection method can also improve the accuracy and reduce the complexity of ensemble classifiers generated by bagging/boosting. To construct a set of classifiers, bagging and boosting repeatedly resample the training dataset to build a set of training resampled datasets. The resampled datasets often contain redundant/irrelevant features. Applying feature selection to each resampled dataset can eliminate redundant/irrelevant features and hence improve the resampled datasets which in turn can help build more accurate and

less complex classifiers. Moreover, feature selection provides different feature subsets for different resampled datasets; therefore, it helps to generate more diverse classifiers which then result in better accuracy.

8.2.3 Interval GP-based Feature Construction for Classification with Incomplete Data

Decision trees such as C4.5 and CART can directly work with incomplete data. However, decision trees are not always effective classifiers—they often do not achieve adequate accuracy when faced with difficult tasks because they cannot capture the complex interaction between features. Feature construction which builds new features from original features is a conventional solution for this problem because such features can capture the relationships between features.

GP has been widely used for feature constructions with complete data, but it cannot directly work with incomplete data because its functions cannot work when faced with missing values. This means that decision trees with GP-based feature construction no longer have the desirable of original decision trees. To deal with this problem, Chapter 5 presented interval GP which uses a set of interval functions to replace the normal function set in traditional GP. To use a feature constructed by interval GP, first, each missing value for a feature is replaced by an interval associated with the feature. The interval functions then operate on the intervals as well as regular value to compute an output value for the feature. The result is that decision trees using these constructed features can still work on incomplete data.

Using interval GP is an alternative to using traditional GP with imputation to deal with the missing data. It was found that interval GP can construct complete features from incomplete data which help to generate more accurate classifiers. The key reason is that replacing a missing value with an interval can better reflect the uncertainty associated with the miss-

ing value than the value generated by single imputation. This idea is to some extent similar to multiple imputation where a set of values is estimated for each missing value which also captures the uncertainty well. Although multiple imputation is accurate, it often takes a long time to estimate the set of values. In contrast, calculating interval of each feature is fast, and only needs to be done one time in the training process. Therefore, interval GP-based feature construction can generate more efficient classifiers than the combination of imputation with traditional GP-based feature construction.

Chapter 5 also showed that interval GP-based feature construction can generate less complex classifiers than the combination of imputation and traditional GP-based feature construction. The underlying reason is that imputation generates more values for the features which then leads to more complex classifiers while the proposed method avoids the need for multiple values.

8.2.4 Ensemble Approach to Classification with Incomplete Data

Chapter 6 presented a new approach to constructing effective and efficient ensemble classifiers for incomplete data by combining imputation, feature selection and ensemble learning.

Building Ensemble Classifiers for Incomplete Data

The thesis confirmed that constructing an ensemble of classifiers to cover all possible missing patterns is an effective and efficient approach to classification with incomplete data. By constructing a set of classifiers, each classifier being built on one missing pattern, unseen incomplete instances can be classified by selecting applicable classifiers without requiring imputation in the application process. As a result, this approach is more efficient than a common approach using imputation both in the training and

application processes. Moreover, there usually exist more than one applicable classifier to classify an unseen incomplete instance which makes the ensemble approach more accurate than the common approach which uses imputation to estimate missing values for unseen incomplete instances, but builds only a single classifier.

Integrating Imputation with Ensemble Learning

The thesis shown that it is important to use imputation in the training process to provide high quality complete training data for constructing ensemble classifiers which then results in more accurate classifiers. Existing ensemble methods for classification with incomplete data also build a set of classifiers, but they do not use any imputation. However, when the original training data contains numerous missing values, the training set for each classifier can be as small as a single instance which results in low accuracy classifiers.

Either single imputation or multiple imputation can be used to estimate missing value for the training data. Multiple imputation is usually more accurate than single imputation, especially when the data contains numerous missing values. Hence, a multiple imputation method should be used to impute missing values for the training data where possible. However, multiple imputation is generally computationally more expensive than single imputation, especially when data contains numerous features. Therefore, with datasets containing a large number of features, a good single imputation method such as kNN-based imputation can be used to impute missing values for the training data and still obtain better accuracy than existing ensemble methods while having a feasible imputation cost in the training process.

Integrating Feature Selection with Ensemble Learning

The thesis showed that feature selection can further improve the effectiveness and efficiency of ensemble classifiers for incomplete data. By removing redundant and irrelevant features, feature selection produces a high quality feature set which then helps to build accurate classifiers. More importantly, feature selection also reduces the number of missing patterns, so fewer classifiers are required which then saves time when identifying applicable classifiers and classifying an unseen instance. Moreover, removing redundant and irrelevant features from unseen instances before classifying them can reduce the number of unseen incomplete instances which also saves time in the application process.

The thesis showed that feature selection can be done effectively and efficiently by using fast and powerful filter methods such as CFS and mRMR and using evolutionary techniques such as PSO.

8.2.5 Directly Evolving Classifiers with GP

Chapter 7 presented interval GP-based methods to directly evolve single classifiers and ensemble classifiers for incomplete data.

GP with Interval Functions to Build Single Classifiers

It was shown that GP using a set of interval functions can evolve an effective and efficient classifier for incomplete data. A classifier evolved by interval GP is often more effective and efficient than a classifier evolved by the combination of imputation and traditional GP. The underlying reason is the same reasons as given in Chapter 5 for the advantage of interval GP-based feature construction.

GP with Interval Function to Build Ensemble Classifiers

Chapter 7 also showed that interval GP also can evolve effective and efficient ensemble classifiers for incomplete data. The output of a classifier evolved by interval GP is an interval, and the class boundary is also an interval. Therefore, the output interval probably spans more than one class boundaries so that more than one class label could be determined by the classifier. However, if interval GP only constructs one classifier for each classification task, it has to use the middle point of the output interval to determine one class label. Therefore, constructing a set of classifiers for each incomplete data can lessen the uncertainty of an interval classifier which then improve the classification accuracy. The combination of interval GP with ensemble learning to evolve classifiers is also more effective than the combination of GP, imputation and ensemble learning because interval GP can directly work with incomplete data without requiring any time to estimate missing values.

In conclusion, the proposed methods are generally significantly more accurate than the existing benchmark methods. Although the accuracy improvement on some of the benchmark datasets is not really large, the improvement is still meaningful in real applications. For example, if we improve a cancer prediction system with 1% of accuracy, the system is used to test million patients, we can save life for thousand patients. Furthermore, the thesis proposed genetic methods which can be applied to another problems. For example, in chapter 6, the proposed method was applied to gene expression datasets and made very meaningful differences with baseline methods (improve more than 10% of accuracy). Moreover, the thesis also focused on improving the efficiency of classification with incomplete data. It is clear from the contribution chapters that the proposed methods in the thesis can dramatically improve the efficiency of classification with incomplete data (speed up thousand times of computation time). Therefore, the proposed methods are really meaningful for classification with incomplete data.

8.3 Future Work

There are a number of directions for future work suggested by the thesis.

8.3.1 Improve Input Space for Classification with Incomplete Data

This thesis proposed several ways to improve the input space for classification with incomplete such as improving imputation by using feature selection and clustering as shown in Chapter 3, improving feature space for classifiers able to directly work with incomplete data by using wrapper-based feature selection and interval GP-based feature construction as shown in Chapter 4 and Chapter 5, respectively. There are other ways to improve the input space, we could investigate.

GP-based Imputation

Regression-based imputation is one of main approaches to estimate missing values. GP has been successfully applied to symbolic regression. In [157, 159], we showed that GP is good at estimating missing values. Along with standard GP for symbolic regression, advanced GP such as semantic GP [173] has been proven to improve symbolic regression. Therefore, advanced GP for symbolic regression should be investigated for effectively and efficiently estimating missing values.

Filter-based Feature Selection

This thesis investigated wrapper-based feature selection for classification with incomplete data. Other researchers have investigated filter-based feature selection methods for incomlete data, but they often do not work well when the data contains a large number of missing values. Therefore, it would be interesting to further investigate filter-based feature selection

for incomplete data. Modifying evaluation measures such as mutual information, Pearson product-moment correlation coefficient and the pointwise mutual information would be more necessary for working directly with incomplete data.

8.3.2 Further Investigate GP to Evolve Classifiers for Incomplete Data

Chapter 7 proposed new methods for using GP to directly evolve classifiers for incomplete data. GP also have been widely used for evolving decision trees and rule-based classifiers , but only for complete data. Therefore, how to use GP to evolve decision trees and rule-based classifiers for incomplete could be a productive direction for further work.

GP for Evolving Decision Trees

Decision trees are one of the most popular representation methods for classifiers. Some decision trees such as C4.5 and CART can directly classify incomplete data, but they often generate inaccurate and more complex classifiers when faced with data containing numerous missing values. Future work could explore using GP to evolve more accurate and less complex decision trees. Existing GP technique to evolve decision trees for complete data [19, 20, 70, 117, 137, 167] could be combined with techniques for handling missing values in C4.5 and CART to evolve decision trees for incomplete data.

GP for Evolving Rule-based Classifiers

Rules are one of the simplest and most interpretable ways to present classifiers. Some rule-based classifiers such as CN2 can classify incomplete data, but they often build inaccurate and complex rules when faced with data containing a large number of missing values. GP has been widely

applied to evolve rule-based classifiers, but only for complete data [18, 60, 77, 154, 168]. Therefore, in future, it could be interesting to use GP to evolve rule-based classifiers that can classify incomplete data better than CN2. Existing GP techniques to evolve rule-based classifiers for complete data could be combined with techniques for handling missing values in CN2 to evolve rules for incomplete data.

8.3.3 Regression with Incomplete Data

Missing values are also a common issue in many regression problems [186]. However, there has been much less work on handling missing data in regression than in classification. In future work, we would like to investigate how the ideas in this thesis can be used for regression with incomplete data.

Imputation for Regression with Incomplete Data

As with classification, the most common approach to regression with incomplete data is to use imputation to estimate missing values before doing regression. Simple imputation such as mean imputation is fast, but not accurate. In contrast, powerful imputation such as MICE is accurate, but slow. It would be useful to investigate how to effectively and efficiently use imputation for regression with incomplete data. Based on ideas in Chapter 3, one possible idea is to integrate clustering and feature selection with imputation to improve the effectiveness and efficiency of using imputation for regression with incomplete data.

Feature Manipulation for Regression with Incomplete Data

Feature manipulation including feature selection and feature construction has been used to improve regression. However, existing feature manipulation techniques cannot directly work with incomplete data, and they have to be combined with imputation to estimate missing values. Therefore,

how to directly perform feature manipulation for regression with incomplete data could be investigated. It would be interesting to investigate how apply ideas of feature selection in Chapter 4 and feature construction in Chapter 5 for regression with incomplete data.

Interval GP for Regression with Incomplete Data

GP is well-known for symbolic regression, but it cannot directly work with incomplete data —GP has to be combined with imputation to estimate missing values before doing regression. One possible idea that could be explored is using interval GP, as Chapter 7, to directly do regression with incomplete data.

Bibliography

[1] ACUNA, E., AND RODRIGUEZ, C. The treatment of missing values and its effect on classifier accuracy. *Classification, clustering, and data mining applications* (2004), 639–647.

[2] AGNELLI, D., BOLLINI, A., AND LOMBARDI, L. Image classification: an evolutionary approach. *Pattern Recognition Letters 23* (2002), 303–309.

[3] AHA, D., AND KIBLER, D. Instance-based prediction of heart-disease presence with the cleveland database. Tech. rep., University of California, 1980.

[4] AMIRI, M., AND JENSEN, R. Missing data imputation using fuzzy-rough methods. *Neurocomputing 205* (2016), 152–164.

[5] ANDERSON-COOK, C. M. Practical genetic algorithms, 2005.

[6] ANDRIDGE, R. R., AND LITTLE, R. J. A review of hot deck imputation for survey non-response. *International Statistical Review 78* (2010), 40–64.

[7] ARMINA, R., ZAIN, A. M., ALI, N. A., AND SALLEHUDDIN, R. A review on missing value estimation using imputation algorithm. In *Journal of Physics: Conference Series* (2017), vol. 892, p. 012004.

[8] ASUNCION, A., AND NEWMAN, D. UCI machine learning repository, 2013.

[9] AUSSEM, A., AND DE MORAIS, S. R. A conservative feature subset selection algorithm with missing data. *Neurocomputing 73* (2010), 585–590.

[10] AZIM, S., AND AGGARWAL, S. Hybrid model for data imputation: using fuzzy c means and multi layer perceptron. In *Advance Computing Conference (IACC), 2014 IEEE International* (2014), pp. 1281–1285.

[11] BACKER, G. Learning with missing data using genetic programming. In *Proceedings of the First Online Workshop on Soft Computing (WSC1)* (1996), pp. 279–283.

[12] BATISTA, G. E., AND MONARD, M. C. A study of k-nearest neighbour as an imputation method. In *Hybrid Intelligent Systems - HIS* (2002), pp. 251–260.

[13] BATISTA, G. E., AND MONARD, M. C. An analysis of four missing data treatment methods for supervised learning. *Applied artificial intelligence 17* (2003), 519–533.

[14] BHOWAN, U., JOHNSTON, M., AND ZHANG, M. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42* (2012), 406–421.

[15] BHOWAN, U., JOHNSTON, M., ZHANG, M., AND YAO, X. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation 17* (2013), 368–386.

[16] BHOWAN, U., JOHNSTON, M., ZHANG, M., AND YAO, X. Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation 18* (2014), 893–908.

[17] BIFET, A., HOLMES, G., PFAHRINGER, B., AND FRANK, E. Fast perceptron decision tree learning from evolving data streams. In *Advances in knowledge discovery and data mining*. 2010, pp. 299–310.

[18] BOJARCZUK, C. C., LOPES, H. S., AND FREITAS, A. A. Genetic programming for knowledge discovery in chest-pain diagnosis. *Engineering in Medicine and Biology Magazine, IEEE 19* (2000), 38–44.

[19] BOT, M. C., AND LANGDON, W. B. Application of genetic programming to induction of linear classification trees. In *Genetic Programming*. Springer, 2000, pp. 247–258.

[20] BOT, M. C., AND LANGDON, W. B. Improving induction of linear classification trees with genetic programming. *space 10*, 1.25 (2000), 2.

[21] BREIMAN, L., FRIEDMAN, J., STONE, C. J., AND OLSHEN, R. A. *Classification and regression trees*. CRC press, 1984.

[22] BUUREN, S., AND GROOTHUIS-OUDSHOORN, K. mice: Multivariate imputation by chained equations in R. *Journal of statistical software 45* (2011).

[23] CHANDRASHEKAR, G., AND SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering 40* (2014), 16–28.

[24] CHEN, H., DU, Y., AND JIANG, K. Classification of incomplete data using classifier ensembles. In *2012 International Conference on Systems and Informatics (ICSAI2012)* (2012), pp. 2229–2232.

[25] CHEN, Y., WANG, A., DING, H., QUE, X., LI, Y., AN, N., AND JIANG, L. A global learning with local preservation method for microarray data imputation. *Computers in biology and medicine 77* (2016), 76–89.

[26] CHUANG, L.-Y., CHANG, H.-W., TU, C.-J., AND YANG, C.-H. Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry* (2008), 29–38.

[27] CLERC, M., AND KENNEDY, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on 6* (2002), 58–73.

[28] COLANTUONI, C., HENRY, G., ZEGER, S., AND PEVSNER, J. Snomad (standardization and normalization of microarray data): web-accessible gene expression data analysis. *Bioinformatics 18* (2002), 1540–1541.

[29] CONVERSANO, C., AND SICILIANO, R. Incremental tree-based missing data imputation with lexicographic ordering. *Journal of classification 26* (2009), 361–379.

[30] DE SOUTO, M. C., COSTA, I. G., DE ARAUJO, D. S., LUDERMIR, T. B., AND SCHLIEP, A. Clustering cancer gene expression data: a comparative study. *BMC bioinformatics 9* (2008), 497.

[31] DE SOUTO, M. C., JASKOWIAK, P. A., AND COSTA, I. G. Impact of missing data imputation methods on gene expression clustering and classification. *BMC bioinformatics 16* (2015), 64.

[32] DEB, R., AND LIEW, A. W.-C. Missing value imputation for the analysis of incomplete traffic accident data. *Information sciences 339* (2016), 274–289.

[33] DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* (2006), 1–30.

[34] DI ZIO, M., SCANU, M., COPPOLA, L., LUZI, O., AND PONTI, A. Bayesian networks for imputation. *Journal of the Royal Statistical Society: Series A (Statistics in Society) 167* (2004), 309–322.

[35] DIETTERICH, T. G., ET AL. Ensemble methods in machine learning. *Multiple classifier systems 1857* (2000), 1–15.

[36] DOQUIRE, G., AND VERLEYSEN, M. Feature selection with missing data using mutual information estimators. *Neurocomputing 90* (2012), 3–11.

[37] DORIGO, M., BIRATTARI, M., AND STUTZLE, T. Ant colony optimization. *IEEE computational intelligence magazine 1* (2006), 28–39.

[38] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern classification*. John Wiley & Sons, 2012.

[39] DUMA, M., MARWALA, T., TWALA, B., AND NELWAMONDO, F. Partial imputation of unseen records to improve classification using a hybrid multi-layered artificial immune system and genetic algorithm. *Applied Soft Computing 13* (2013), 4461–4480.

[40] ELTER, M., SCHULZ-WENDTLAND, R., AND WITTENBERG, T. The prediction of breast cancer biopsy outcomes using two cad approaches that both emphasize an intelligible decision process. *Medical physics 34* (2007), 4164–4172.

[41] ENDERS, C. K. *Applied missing data analysis*. Guilford Press, 2010.

[42] ESPEJO, P. G., VENTURA, S., AND HERRERA, F. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40* (2010), 121–144.

[43] FAHAD, A., ALSHATRI, N., TARI, Z., ALAMRI, A., KHALIL, I., ZOMAYA, A. Y., FOUFOU, S., AND BOURAS, A. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing 2* (2014), 267–279.

[44] FANG, F., AND SHAO, J. Iterated imputation estimation for generalized linear models with missing response and covariate values. *Computational Statistics & Data Analysis 103* (2016), 111–123.

[45] FARHANGFAR, A., KURGAN, L., AND DY, J. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition 41* (2008), 3692–3705.

[46] FARHANGFAR, A., KURGAN, L. A., AND PEDRYCZ, W. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 37* (2007), 692–709.

[47] FESSANT, F., AND MIDENET, S. Self-organising map for data imputation and correction in surveys. *Neural Computing & Applications 10* (2002), 300–310.

[48] FOGEL, D. B. *Evolutionary computation: toward a new philosophy of machine intelligence*, vol. 1. John Wiley & Sons, 2006.

[49] FOLCH-FORTUNY, A., ARTEAGA, F., AND FERRER, A. Pca model building with missing data: New proposals and a comparative study. *Chemometrics and Intelligent Laboratory Systems 146* (2015), 77–88.

[50] FOLGUERA, L., ZUPAN, J., CICERONE, D., AND MAGALLANES, J. F. Self-organizing maps for imputation of missing data in incomplete data matrices. *Chemometrics and Intelligent Laboratory Systems 143* (2015), 146–151.

[51] FOLINO, G., AND PISANI, F. S. Evolving meta-ensemble of classifiers for handling incomplete and unbalanced datasets in the cyber security domain. *Applied Soft Computing 47* (2016), 179–190.

[52] FRANK, E., AND WITTEN, I. H. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning* (1998), pp. 144–151.

[53] FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association 32*, 200 (1937), 675–701.

[54] GAJAWADA, S., AND TOSHNIWAL, D. Missing value imputation method based on clustering and nearest neighbours. *International Journal of Future Computer and Communication 1* (2012), 206.

[55] GARCÍA, J. C. F., KALENATIC, D., AND BELLO, C. A. L. Missing data imputation in multivariate data by evolutionary algorithms. *Computers in Human Behavior 27* (2011), 1468–1474.

[56] GARCÍA-LAENCINA, P. J., ABREU, P. H., ABREU, M. H., AND AFONOSO, N. Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values. *Computers in biology and medicine 59* (2015), 125–133.

[57] GARCÍA-LAENCINA, P. J., SANCHO-GÓMEZ, J.-L., AND FIGUEIRAS-VIDAL, A. R. Pattern classification with missing data: a review. *Neural Computing and Applications 19* (2010), 263–282.

[58] GARCÍA-LAENCINA, P. J., SANCHO-GÓMEZ, J.-L., FIGUEIRAS-VIDAL, A. R., AND VERLEYSEN, M. K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing 72* (2009), 1483–1493.

[59] GAUTAM, C., AND RAVI, V. Data imputation via evolutionary computation, clustering and a neural network. *Neurocomputing 156* (2015), 134–142.

[60] GILBERT, R. J., ROWLAND, J. J., AND KELL, D. B. Genomic computing: explanatory modelling for functional genomics. In *GECCO* (2000), pp. 551–557.

[61] GRAHAM, J. W. Missing data analysis: Making it work in the real world. *Annual review of psychology 60* (2009), 549–576.

[62] GUERRA-SALCEDO, C., AND WHITLEY, D. Feature selection mechanisms for ensemble creation: a genetic search perspective. In *Data Mining with Evolutionary Algorithms: Research Directions. Papers from the AAAI Workshop* (1999).

[63] GUO, H., JACK, L. B., AND NANDI, A. K. Feature generation using genetic programming with application to fault classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 35* (2005), 89–99.

[64] GUO, H., AND NANDI, A. K. Breast cancer diagnosis using genetic programming generated feature. *Pattern Recognition 39* (2006), 980–987.

[65] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter 11* (2009), 10–18.

[66] HALL, M. A. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning* (2000), pp. 359–366.

[67] HAN, J., PEI, J., AND KAMBER, M. *Data mining: concepts and techniques*. Elsevier, 2011.

[68] HANSEN, E., AND WALSTER, G. W. *Global optimization using interval analysis: revised and expanded*, vol. 264. CRC Press, 2003.

[69] HAREL, O., AND ZHOU, X.-H. Multiple imputation: review of theory, implementation and software. *Statistics in medicine 26* (2007), 3057–3077.

[70] HARUYAMA, S., AND ZHAO, Q. Designing smaller decision trees using multiple objective optimization based gps. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on* (2002), vol. 6, pp. 5–pp.

[71] HENNESSY, K., MADDEN, M. G., CONROY, J., AND RYDER, A. G. An improved genetic programming technique for the classification of raman spectra. *Knowledge-Based Systems 18* (2005), 217–224.

[72] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[73] HRUSCHKA, E. R., HRUSCHKA, E. R., AND EBECKEN, N. F. Bayesian networks for imputation in classification problems. *Journal of Intelligent Information Systems 29* (2007), 231–252.

[74] HUANG, C.-L., AND DUN, J.-F. A distributed pso–svm hybrid system with feature selection and parameter optimization. *Applied Soft Computing 8* (2008), 1381–1391.

[75] JEREZ, J. M., MOLINA, I., GARCÍA-LAENCINA, P. J., ALBA, E., RIBELLES, N., MARTÍN, M., AND FRANCO, L. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial intelligence in medicine 50* (2010), 105–115.

[76] JIANG, K., CHEN, H., AND YUAN, S. Classification for incomplete data using classifier ensembles. In *2005 International Conference on Neural Networks and Brain* (2005), vol. 1, pp. 559–563.

[77] JOHNSON, H. E., GILBERT, R. J., WINSON, M. K., GOODACRE, R., SMITH, A. R., ROWLAND, J. J., HALL, M. A., AND KELL, D. B. Explanatory analysis of the metabolome using genetic programming

of simple, interpretable rules. *Genetic Programming and Evolvable Machines 1* (2000), 243–258.

[78] JOSE-GARCIA, A., AND GOMEZ-FLORES, W. Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing 41* (2016), 192–213.

[79] JUSZCZAK, P., AND DUIN, R. P. Combining one-class classifiers to classify missing data. In *International Workshop on Multiple Classifier Systems* (2004), pp. 92–101.

[80] KANG, P. Locally linear reconstruction based missing value imputation for supervised learning. *Neurocomputing 118* (2013), 65–78.

[81] KANUNGO, T., MOUNT, D. M., NETANYAHU, N. S., PIATKO, C. D., SILVERMAN, R., AND WU, A. Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence 24* (2002), 881–892.

[82] KARABOGA, D., AND BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization 39* (2007), 459–471.

[83] KEIJZER, M. Improving symbolic regression with interval arithmetic and linear scaling. *Genetic programming* (2003), 275–299.

[84] KENNEDY, J. Particle swarm optimization. In *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

[85] KENNEDY, J., KENNEDY, J. F., AND EBERHART, R. C. *Swarm intelligence*. Morgan Kaufmann, 2001.

[86] KISHORE, J., PATNAIK, L. M., MANI, V., AND AGRAWAL, V. Application of genetic programming for multicategory pattern classification. *Evolutionary Computation, IEEE Transactions on 4* (2000), 242–258.

[87] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[88] KOZA, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and computing 4*, 2 (1994), 87–112.

[89] KOZA, J. R. *Genetic programming III: Darwinian invention and problem solving*, vol. 3. 1999.

[90] KRAUSE, S., AND POLIKAR, R. An ensemble of classifiers approach for the missing feature problem. In *Proceedings of the International Joint Conference on Neural Networks, 2003.* (2003), vol. 1, pp. 553–558 vol.1.

[91] KRAWIEC, K. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines 3* (2002), 329–343.

[92] KWON, T. Y., AND PARK, Y. A new multiple imputation method for bounded missing values. *Statistics & Probability Letters 107* (2015), 204–209.

[93] LAKSHMINARAYAN, K., HARP, S. A., AND SAMAD, T. Imputation of missing data in industrial databases. *Applied Intelligence 11* (1999), 259–275.

[94] LANE, M. C., XUE, B., LIU, I., AND ZHANG, M. Gaussian based particle swarm optimisation and statistical clustering for feature selection. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (2014), pp. 133–144.

[95] LENSBERG, T., EILIFSEN, A., AND MCKEE, T. E. Bankruptcy theory development and classification via genetic programming. *European Journal of Operational Research 169*, 2 (2006), 677–697.

[96] LI, D., DEOGUN, J., SPAULDING, W., AND SHUART, B. Towards missing data imputation: a study of fuzzy k-means clustering method. In *Rough sets and current trends in computing* (2004), vol. 3066, pp. 573–579.

[97] LIAW, A., WIENER, M., ET AL. Classification and regression by randomforest. *R news 2* (2002), 18–22.

[98] LIN, S.-W., YING, K.-C., CHEN, S.-C., AND LEE, Z.-J. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications 35* (2008), 1817–1824.

[99] LIN, W.-C., KE, S.-W., AND TSAI, C.-F. When should we ignore examples with missing values? *International Journal of Data Warehousing and Mining (IJDWM) 13* (2017), 53–63.

[100] LITTLE, R. J., AND RUBIN, D. B. *Statistical analysis with missing data*. John Wiley & Sons, 2014.

[101] LIU, C.-C., DAI, D.-Q., AND YAN, H. The theoretic framework of local weighted approximation for microarray missing value estimation. *Pattern Recognition 43* (2010), 2993–3002.

[102] LIU, Y., AND BROWN, S. D. Comparison of five iterative imputation methods for multivariate classification. *Chemometrics and Intelligent Laboratory Systems 120* (2013), 106–115.

[103] LIU, Z.-G., PAN, Q., DEZERT, J., AND MARTIN, A. Adaptive imputation of missing values for incomplete pattern classification. *Pattern Recognition 52* (2016), 85–95.

[104] LOBATO, F., SALES, C., ARAUJO, I., TADAIESKY, V., DIAS, L., RAMOS, L., AND SANTANA, A. Multi-objective genetic algorithm

for missing data imputation. *Pattern Recognition Letters 68* (2015), 126–131.

[105] LOH, W.-Y. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1* (2011), 14–23.

[106] LONG, Q., AND JOHNSON, B. A. Variable selection in the presence of missing data: resampling and imputation. *Biostatistics 16* (2015), 596–610.

[107] LONGFORD, N. T. *Missing data and small-area estimation: Modern analytical equipment for the survey statistician*. Springer Science & Business Media, 2006.

[108] LOU, Q., AND OBRADOVIC, Z. Margin-based feature selection in incomplete data. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (2012), pp. 1040–1046.

[109] LUENGO, J., GARCÍA, S., AND HERRERA, F. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems* (2012), 1–32.

[110] LUKE, S., PANAIT, L., BALAN, G., PAUS, S., SKOLICKI, Z., POPOVICI, E., SULLIVAN, K., HARRISON, J., BASSETT, J., HUBLEY, R., ET AL. A java-based evolutionary computation research system. *Online (March 2004) http://cs. gmu. edu/˜ eclab/projects/ecj* (2004).

[111] MACKAY, D. J. *Information theory, inference, and learning algorithms*, vol. 7. Citeseer, 2003.

[112] MARWALA, T., AND CHAKRAVERTY, S. Fault classification in structures with incomplete measured data using autoassociative neural networks and genetic algorithm. *Current Science* (2006), 542–548.

[113] MEESAD, P., AND HENGPRAPROHM, K. Combination of knn-based feature selection and knn-based missing-value imputation of microarray data. In *Innovative Computing Information and Control, 2008. ICICIC'08. 3rd International Conference on* (2008), pp. 341–341.

[114] MUHARRAM, M., AND SMITH, G. D. Evolutionary constructive induction. *Knowledge and Data Engineering, IEEE Transactions on 17* (2005), 1518–1528.

[115] MUSSER, D. R. Introspective sorting and selection algorithms. *Softw., Pract. Exper. 27* (1997), 983–993.

[116] NESHATIAN, K., ZHANG, M., AND ANDREAE, P. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation 16* (2012), 645–661.

[117] OKA, S., AND ZHAO, Q. Design of decision trees through integration of c4. 5 and gp. In *Proc. 4th Jpn.-Australia Joint Workshop Intell. Evol. Syst* (2000), pp. 128–135.

[118] OLIVEIRA, L. S., MORITA, M., AND SABOURIN, R. Feature selection for ensembles applied to handwriting recognition. *International Journal of Document Analysis and Recognition (IJDAR) 8* (2006), 262–279.

[119] OPITZ, D. W. Feature selection for ensembles. *AAAI/IAAI 379* (1999), 384.

[120] OPITZ, D. W., AND MACLIN, R. Popular ensemble methods: An empirical study. *J. Artif. Intell. Res.(JAIR) 11* (1999), 169–198.

[121] PATIL, B. M., JOSHI, R. C., AND TOSHNIWAL, D. Missing value imputation based on k-mean clustering with weighted distance. In *International Conference on Contemporary Computing* (2010), pp. 600–609.

[122] PENG, H., LONG, F., AND DING, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence 27* (2005), 1226–1238.

[123] POLI, R., LANGDON, W. B., MCPHEE, N. F., AND KOZA, J. R. *A field guide to genetic programming*. Lulu. com, 2008.

[124] POLIKAR, R. Ensemble based systems in decision making. *IEEE Circuits and systems magazine 6* (2006), 21–45.

[125] POLIKAR, R., DEPASQUALE, J., MOHAMMED, H. S., BROWN, G., AND KUNCHEVA, L. I. Learn++. mf: A random subspace approach for the missing feature problem. *Pattern Recognition 43*, 11 (2010), 3817–3832.

[126] PRIYA, R. D., AND KUPPUSWAMI, S. A genetic algorithm based approach for imputing missing discrete attribute values in databases. *WSEAS Transactions on Information Science and Applications 9* (2012), 169–178.

[127] PURWAR, A., AND SINGH, S. K. Hybrid prediction model with missing value imputation for medical data. *Expert Systems with Applications 42* (2015), 5621–5631.

[128] QIAN, W., AND SHU, W. Mutual information criterion for feature selection from incomplete data. *Neurocomputing 168* (2015), 210–220.

[129] QUINLAN, J. R. *C4. 5: programs for machine learning*. Elsevier, 2014.

[130] RAHMAN, G., AND ISLAM, Z. A decision tree-based missing value imputation technique for data pre-processing. In *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121* (2011), pp. 41–50.

[131] RAHMAN, M. G., AND ISLAM, M. Z. Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. *Knowledge-Based Systems 53* (2013), 51–65.

[132] RAHMAN, M. G., AND ISLAM, M. Z. Fimus: A framework for imputing missing values using co-appearance, correlation and similarity analysis. *Knowledge-Based Systems 56* (2014), 311–327.

[133] RANA, S., JOHN, A. H., AND MIDI, H. Robust regression imputation for analyzing missing data. In *Statistics in Science, Business, and Engineering (ICSSBE), 2012 International Conference on* (2012), pp. 1–4.

[134] RANCOITA, P. M., ZAFFALON, M., ZUCCA, E., BERTONI, F., AND DE CAMPOS, C. P. Bayesian network data imputation with application to survival tree analysis. *Computational Statistics & Data Analysis 93* (2016), 373–387.

[135] RATANAMAHATANA, C. A., AND GUNOPULOS, D. Feature selection for the naive bayesian classifier using decision trees. *Applied artificial intelligence 17* (2003), 475–487.

[136] RAYMER, M. L., PUNCH, W. F., GOODMAN, E. D., AND KUHN, L. A. Genetic programming for improved data mining: application to the biochemistry of protein interactions. In *Proceedings of the 1st annual conference on genetic programming* (1996), pp. 375–380.

[137] ROUWHORST, S., AND ENGELBRECHT, A. Searching the forest: using decision trees as building blocks for evolutionary search in classification databases. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* (2000), vol. 1, pp. 633–638.

[138] RUBIN, D. B. *Multiple imputation for nonresponse in surveys*, vol. 81. John Wiley & Sons, 2004.

[139] SAAR-TSECHANSKY, M., AND PROVOST, F. Handling missing values when applying classification models. *Journal of machine learning research 8* (2007), 1623–1657.

[140] SCHAFER, J. L. *Analysis of incomplete multivariate data*. CRC press, 1997.

[141] SCHAFER, J. L., AND GRAHAM, J. W. Missing data: our view of the state of the art. *Psychological methods 7* (2002), 147.

[142] SHARPE, P. K., AND SOLLY, R. J. Dealing with missing values in neural network-based diagnostic systems. *Neural Computing & Applications 3*, 2 (1995), 73–77.

[143] SHERRAH, J., BOGNER, R. E., AND BOUZERDOUM, A. Automatic selection of features for classification using genetic programming. In *Intelligent Information Systems, 1996., Australian and New Zealand Conference on* (1996), pp. 284–287.

[144] SHERRAH, J. R., BOGNER, R. E., AND BOUZERDOUM, A. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. *Genetic Programming* (1997), 304–312.

[145] SHI, H. Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton, NZ, 2007. COMP594.

[146] SILVA, S., AND TSENG, Y.-T. Classification of seafloor habitats using genetic programming. In *Applications of Evolutionary Computing*. 2008, pp. 315–324.

[147] SILVA-RAMÍREZ, E.-L., PINO-MEJÍAS, R., AND LÓPEZ-COELLO, M. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing 29* (2015), 65–74.

[148] SILVA-RAMÍREZ, E.-L., PINO-MEJÍAS, R., LÓPEZ-COELLO, M., AND CUBILES-DE-LA VEGA, M.-D. Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks 24* (2011), 121–129.

[149] SMITH, M. G., AND BULL, L. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines 6* (2005), 265–281.

[150] SOVILJ, D., EIROLA, E., MICHE, Y., BJÖRK, K.-M., NIAN, R., AKUSOK, A., AND LENDASSE, A. Extreme learning machine for missing data using multiple imputations. *Neurocomputing 174* (2016), 220–231.

[151] STERNE, J. A., WHITE, I. R., CARLIN, J. B., SPRATT, M., ROYSTON, P., KENWARD, M. G., WOOD, A. M., AND CARPENTER, J. R. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *Bmj 338* (2009), b2393.

[152] SUGUMARAN, V., MURALIDHARAN, V., AND RAMACHANDRAN, K. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing 21* (2007), 930–942.

[153] TACKETT, W. A. Genetic programming for feature discovery and image discrimination. In *ICGA* (1993), pp. 303–311.

[154] TAN, K., TAY, A., LEE, T., AND HENG, C. Mining multiple comprehensible classification rules using genetic programming. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (2002), vol. 2, pp. 1302–1307.

[155] TAN, X., BHANU, B., AND LIN, Y. Fingerprint classification based on learned features. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 35* (2005), 287–300.

[156] TIAN, J., YU, B., YU, D., AND MA, S. Clustering-based multiple imputation via gray relational analysis for missing data and its application to aerospace field. *The Scientific World Journal 2013* (2013).

[157] TRAN, B., XUE, B., AND ZHANG, M. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing 8*, 1 (2016), 3–15.

[158] TRAN, C. T., ANDREAE, P., AND ZHANG, M. Impact of imputation of missing values on genetic programming based multiple feature construction for classification. In *Evolutionary Computation (CEC), 2015 IEEE Congress on* (2015), pp. 2398–2405.

[159] TRAN, C. T., ZHANG, M., AND ANDREAE, P. Multiple imputation for missing data using genetic programming. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference* (2015), pp. 583–590.

[160] TRAN, C. T., ZHANG, M., AND ANDREAE, P. A genetic programming-based imputation method for classification with missing data. In *European Conference on Genetic Programming* (2016), pp. 149–163.

[161] TRAN, C. T., ZHANG, M., ANDREAE, P., AND XUE, B. Directly constructing multiple features for classification with missing data using genetic programming with interval functions. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion* (2016), pp. 69–70.

[162] TRAN, C. T., ZHANG, M., ANDREAE, P., AND XUE, B. Improving performance for classification with incomplete data using wrapper-based feature selection. *Evolutionary Intelligence 9* (2016), 81–94.

[163] TRAN, C. T., ZHANG, M., ANDREAE, P., AND XUE, B. Genetic programming based feature construction for classification with incom-

plete data. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), pp. 1033–1040.

[164] TRAN, C. T., ZHANG, M., ANDREAE, P., AND XUE, B. Multiple imputation and genetic programming for classification with incomplete data. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), pp. 521–528.

[165] TROYANSKAYA, O., CANTOR, M., SHERLOCK, G., BROWN, P., HASTIE, T., TIBSHIRANI, R., BOTSTEIN, D., AND ALTMAN, R. B. Missing value estimation methods for dna microarrays. *Bioinformatics 17* (2001), 520–525.

[166] TSAI, C.-F., AND CHANG, F.-Y. Combining instance selection for better missing value imputation. *Journal of Systems and Software 122* (2016), 63 – 71.

[167] TSAKONAS, A. A comparison of classification accuracy of four genetic programming-evolved intelligent structures. *Information Sciences 176* (2006), 691–724.

[168] TSAKONAS, A., DOUNIAS, G., JANTZEN, J., AXER, H., BJERREGAARD, B., AND VON KEYSERLINGK, D. G. Evolving rule-based systems in two medical domains using genetic programming. *Artificial Intelligence in Medicine 32* (2004), 195–216.

[169] TSIKRIKTSIS, N. A review of techniques for treating missing data in om survey research. *Journal of Operations Management 24* (2005), 53–62.

[170] TUTZ, G., AND RAMZAN, S. Improved methods for the imputation of missing data by nearest neighbor methods. *Computational Statistics & Data Analysis 90* (2015), 84–99.

[171] USHARANI, Y., AND SAMMULAL, P. A novel approach for imputation of missing attribute values for efficient mining of medical datasets-class based cluster approach. *arXiv preprint arXiv:1605.01010* (2016).

[172] VAN BUUREN, S., AND OUDSHOORN, K. Flexible multivariate imputation by MICE. Tech. rep., PG/VGZ/99.054: TNO Prevention and Health, Leiden, 1999.

[173] VANNESCHI, L. An introduction to geometric semantic genetic programming. In *NEO 2015*. 2017, pp. 3–42.

[174] VATEEKUL, P., AND SARINNAPAKORN, K. Tree-based approach to missing data imputation. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on* (2009), pp. 70–75.

[175] WHITE, I. R., ROYSTON, P., AND WOOD, A. M. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine 30* (2011), 377–399.

[176] WHITE, T. K., REITER, J. P., AND PETRIN, A. Imputation in us manufacturing data and its implications for productivity dispersion. *Review of Economics and Statistics*, 0 (2016).

[177] WITTEN, I. H., FRANK, E., HALL, M. A., AND PAL, C. J. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[178] WOOD, A. M., WHITE, I. R., AND THOMPSON, S. G. Are missing outcome data adequately handled? a review of published randomized controlled trials in major medical journals. *Clinical trials 1* (2004), 368–376.

[179] WU, X., KUMAR, V., QUINLAN, J. R., GHOSH, J., YANG, Q., MOTODA, H., MCLACHLAN, G. J., NG, A., LIU, B., PHILIP, S. Y.,

ET AL. Top 10 algorithms in data mining. *Knowledge and information systems 14*, 1 (2008), 1–37.

[180] XUE, B., ZHANG, M., AND BROWNE, W. N. Single feature ranking and binary particle swarm optimisation based feature subset ranking for feature selection. In *Proceedings of the Thirty-fifth Australasian Computer Science Conference-Volume 122* (2012), pp. 27–36.

[181] XUE, B., ZHANG, M., AND BROWNE, W. N. Particle swarm optimization for feature selection in classification: A multi-objective approach. *Cybernetics, IEEE Transactions on 43* (2013), 1656–1671.

[182] XUE, B., ZHANG, M., AND BROWNE, W. N. A comprehensive comparison on evolutionary feature selection approaches to classification. *International Journal of Computational Intelligence and Applications 14* (2015), 1550008.

[183] XUE, B., ZHANG, M., BROWNE, W. N., AND YAO, X. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation 20* (2016), 606–626.

[184] YAN, Y.-T., ZHANG, Y.-P., AND ZHANG, Y.-W. Multi-granulation ensemble classification for incomplete data. In *International Conference on Rough Sets and Knowledge Technology* (2014), Springer, pp. 343–351.

[185] YAN, Y.-T., ZHANG, Y.-P., ZHANG, Y.-W., AND DU, X.-Q. A selective neural network ensemble classification for incomplete data. *International Journal of Machine Learning and Cybernetics* (2016), 1–12.

[186] YU, Q., MICHE, Y., EIROLA, E., VAN HEESWIJK, M., SÉVERIN, E., AND LENDASSE, A. Regularized extreme learning machine for regression with missing data. *Neurocomputing 102* (2013), 45–51.

[187] ZHANG, C., QIN, Y., ZHU, X., ZHANG, J., AND ZHANG, S. Clustering-based missing value imputation for data preprocessing. In *Industrial Informatics, 2006 IEEE International Conference on* (2006), pp. 1081–1086.

[188] ZHANG, L., JACK, L. B., AND NANDI, A. K. Fault detection using genetic programming. *Mechanical Systems and Signal Processing 19* (2005), 271–289.

[189] ZHANG, M., GAO, X., AND LOU, W. A new crossover operator in genetic programming for object classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37* (2007), 1332–1343.

[190] ZHANG, M., AND SMART, W. Multiclass object classification using genetic programming. In *Applications of Evolutionary Computing*. 2004, pp. 369–378.

[191] ZHANG, M., AND SMART, W. Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recognition Letters 27* (2006), 1266–1274.

[192] ZHANG, M., AND WONG, P. Genetic programming for medical classification: a program simplification approach. *Genetic Programming and Evolvable Machines 9* (2008), 229–255.

[193] ZHOU, Z.-H. Ensemble learning. *Encyclopedia of biometrics* (2015), 411–416.