A Transfer-based additive LS-SVM classifier for handling missing data

Guanjin Wang, Jie Lu, *Fellow, IEEE,* Kup-Sze Choi, Guangquan Zhang

***Abstract*—The performance of a classifier might greatly deteriorate due to missing data. Many different techniques to handle this problem have been developed. In this work, we solve the problem of missing data using a novel transfer learning perspective and show that when additive LS-SVM is adopted, model transfer learning can be used to enhance classification performance on incomplete training datasets. A novel transfer-based additive LS-SVM classifier is accordingly proposed. This method also simultaneously determines the influence of classification errors caused by each incomplete sample using a fast leave-one-out cross validation strategy, as an alternative way to clean the training data to further improve data quality. The proposed method has been applied to seven public datasets. The experimental results indicate that the proposed method achieves at least comparable, if not better, performance than case deletion, mean imputation, and k-nearest neighbor imputation methods, followed by the standard LS-SVM and SVM classifiers. Moreover, a case study on a community healthcare dataset using the proposed method is presented in detail, which particularly highlights the contributions and benefits of the proposed method to this real world application.**

***Index Terms*—missing data, transfer learning, classification, data cleaning, support vector machine**

I. Introduction

CLassification in artificial intelligence categorizes unknown data into predefined classes through learning. A supervised classifier discovers patterns in data with class labels (training data) and then uses them to classify new data without class labels (testing data). The rapid growth of classification techniques has seen them successfully applied in various fields such as computer science, engineering, finance, biology, nursing, and so on. Relevant applications include remote sensing, housing investment, cancer diagnosis, and to estimate quality of life. However, data missing is a common issue, and is attributed to various causes. For example, participants might skip questions in surveys or drop out of experiments. Patients may not qualify for certain medical tests, or operators may take incorrect measurements during data acquisition. Any inappropriate treatment of missing data might consequently deteriorate classification performance and, as such, the ability to appropriately handle missing data in classification problems has always been an essential demand.

There are many methods in the literature for dealing with the classification of missing data. For example, Thirukumaran et al. [1] explored the imputation technique for the missing data. Razzaghi et al. [2] discussed a multilevel learning paradigm of the cost-sensitive SVM on health care missing data. Lorenzi et al. [3] designed a specific kernel combination in a support vector regression, which demonstrates that only few support vectors are needed to reconstruct a missing area. Zhang et al. [4] adopted least squares support vector machines to handle missing traffic flow data. Most of these methods apply classifiers after the missing data have been preprocessed, such as imputation. However, one category goes beyond the traditional methods, and uses machine learning solutions which work directly with the missing data instead of hypothetically predicting missing values. Research work on this topic is rapidly growing and many machine learning solutions have achieved satisfactory performance. Nevertheless, so far there are no reports of using transfer learning as part of an approach. Additionally, most machine learning methods focus on improving general performance on missing data, but little attention has been given to how to detect or remove corrupt and/or meaningless incomplete samples from the dataset for a simultaneous and unbiased estimation guarantee quickly. If this can be achieved, data consistency and quality can be improved.

Missing data can occur in scenarios where every sample in the dataset has one or more missing values, or where a portion of samples in the dataset have missing values but others are complete. In this work, we focus on the latter situation which is very common in real world. Medical data is one such example. We propose a novel additive least squares support vector machine (LS-SVM) classifier for directly handling missing data in both the training and testing datasets from a transfer learning perspective. It is assumed that the LS-SVM framework [5] is adopted in both the source and target domains, where the source domain represents the complete sample, while the target domain represents the complete and incomplete samples with the missing values in the whole training dataset. The proposed method aims to leverage the learned model-based knowledge from the source domain onto the target domain by finding a correlation between the weight parameters of the source and target domains within the LS-SVM framework. This work

The work was supported by the Australian Research Council (ARC) under Discovery Grant DP170101632 and the Research Grants Council of Hong Kong (PolyU 152040/16E). G. Wang is supported by the Y.C. Yu Scholarship for Centre for Smart Health.

Guanjin Wang is with Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia, and also with Centre for Smart Health, School of Nursing, the Hong Kong Polytechnic University, Hong Kong (e-mail: Guanjin.Wang@student.uts.edu.au).

Jie Lu and Guangquan Zhang are with Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia (e-mail: Jie.Lu, Guangquan.Zhang@uts.edu.au).

Kup-Sze Choi is with Centre for Smart Health, School of Nursing, the Hong Kong Polytechnic University, Hong Kong (e-mail: thomasks.choi@polyu.edu.hk).

makes the following contributions:

(1) A novel transfer-based additive LS-SVM classifier is proposed for classification with missing data, by minimizing disagreement between the source and target classifiers using weight parameter consensus regularization terms.

(2) The proposed classifier provides distinct information for data cleaning to guarantee data quality. By evaluating the influence of classification errors caused by each incomplete sample during the model's construction, incomplete samples with high error influence can be discovered and immediately removed.

(3) The proposed classifier uses a fast leave-one-out cross validation strategy to determine how much parameter knowledge should be learnt from the source classifier and the influence level of the classification errors caused by each incomplete sample in the target domain unbiasedly and quickly.

The proposed classifier is applied to UCI public datasets with various combinations of missing data rates and columns, and its accuracy is compared to traditional missing data treatment methods, including case deletion, mean imputation and KNN imputation. Moreover, a case study on a real community healthcare dataset is also presented. The experimental results demonstrate the effectiveness of the proposed method.

The remainder of this paper is organized as follows. Related work is introduced in Section II. In Section III the novel transfer-based additive LS-SVM classifier is proposed. The fast leave-one-out cross validation strategy for the parameters is developed in Section IV. Section V presents the experimental results on the UCI public datasets, and Section VI reports the case study on the real world community health dataset. Finally, the conclusions and future work are given in VII.

II. Related Work

A. Classification with missing data

Pattern classification involves two parts: handling missing data and pattern classification. Generally, we can summarize methods in literature into four categories [6].

Methods in the first category simply remove incomplete samples, and use complete samples for classifier construction [7]. However, deleting samples may cause loss of information and introduce bias into the analysis, particularly when the missing data are not entirely randomly distributed [7], [8].

Methods in the second category impute missing values and construct classifiers using the recovered dataset. The statistical imputation methods used include mean imputation [5], regression imputation [7] and so on. Mean imputation is the simplest: a missing value is filled by the average value of the same feature. In regression imputation, the feature with missing values is estimated by a regression model constructed using non-missing features. The former method does not consider the correlations between missing and non-missing features [9] while the latter method only follows a single regression curve limited by the inherent variation in the data [7]. Imputation can also use machine learning techniques such as k-nearest neighbour (KNN). In this method, the k-nearest neighbours are selected from complete samples to estimate the missing values. However, the performance of KNN imputation is dependent on parameter settings, such as the value of k, the distance function, and the weighting function, and no theoretical approaches can directly determine them. In addition, the search for the nearest neighbours, i.e. the most similar samples, within the portion of complete data is computationally expensive.

Methods in the third category estimate the data distributions of the complete and incomplete data portions in the dataset , and make use of them for pattern classification. In this approach, an expectation maximization (EM) algorithm is commonly used to estimate the data distribution, and Bayesian decision theory is applied for classification [10]. However, the methods in this category have massive computational costs. Calculating standard errors for the estimates [11]and Monte Carlo implementation of the EM algorithm (MCEM) to model joint distribution of the covariates [12] is complicated and limits the applicability of these methods.

Methods in the fourth category handle missing data and construct the classifier at the same time. An increasing number of studies in this category have attempted to improve the generalization ability, and many have demonstrated satisfactory results [6]. In recent years, some works have concentrated on SVM for handling missing data [13]–[17]. Pelckmans et al. [13] presented an idea to integrate the uncertainty caused by missing values into an appropriate risk function, and an extension of this work was based on a formulation of an SVM and LS-SVM classifier. In [14], SVM was incorporated into a Gaussian process to handle missing data. In this approach, how to estimate missing values is equivalent to finding efficient optimization methods such as the EM algorithm. Chechik et al. [15] proposed a maxmargin learning framework using a geometrically-inspired objective function to directly classify incomplete data with lower computational costs. Bi and Zhang [16] were also inspired by the probability modelling approach, and proposed a new SVM classification formulation, which handles missing data with an intuitive geometric interpretation. In [17], a standard SVM classifier was extended for missing data classification using probabilistic classification constraints instead of linear ones. Our proposed method belongs to this category, and is extended to an additive LS-SVM classifier, from a transfer learning perspective, to solve the classification of missing data.

B. Transfer learning

Since our work is based on transfer learning, we first provide a brief review of this field. Three important aspects should be considered:

1) What to transfer: considers which part of the knowledge can be transferred across domains and tasks. In different scenarios, leveraged knowledge can be categorized by instances, feature representations, or model parameters [18].

Instance transfers The main idea in this category is that, even though not all of the data in the source domain can be reused directly, a certain portion can be re-weighted for use in the target domain. In [19], Jiang et al. proposed a general instance weighting framework to solve classification problems across domains in natural language processing. Huang et.al.

[20] presented a non-parametric method which directly generates resampling weights without distribution estimations in scenarios where the training and testing datasets are drawn from different distributions.

Feature representation transfers In this category, the knowledge transferred between domains is encoded into a shared representative knowledge structure, and model construction in the target domain is guided by the new feature space. In [21], Jebara computed a common feature selection or kernel selection configuration for multiple SVM constructed in different domains. In [22], Argyriou et al. proposed a framework to learn the common structure of multi-tasks through regularization with spectral functions of matrices. Another very simple and easy approach was proposed in [23] for classification across different domains by augmenting the feature space. Raina et al. presented 'self-taught learning' for the target domain by constructing higher-level features on unsupervised classification tasks.

Model/parameter transfers This category can be further divided into two subcategories, ensemble learning and domain adaption, ensemble learning combines several classifiers from different domains to achieve one ensemble classifier, while domain adaption assumes that the source and target domains share some parameters or prior distributions and is, nowadays, regarded as a very promising approach. In [24], the proposed method integrated both the global and local information regarding the domains to transform the domain adaptation problem into a bi-object optimization problem via the kernel-based method. Uzair et al. [25] proposed a blind domain adaptation method which does not require samples from target domain for training, via unsupervised learning with a global nonlinear extreme learning machine (ELM) model from the source domain data. Yan et al. [26] proposed maximum independence domain adaptation (MIDA) and semi-supervised MIDA to solve discrete and continuous distributional changes in the feature space.

2) How to transfer: After determining which knowledge to transfer, a corresponding transfer learning model needs to be built. Numerous techniques in computational intelligence have been applied to this area, including neural network transfer learning, Bayes transfer learning and fuzzy transfer learning [27]. Liu et al. [28] applied a neural network to initialize the weights of labelled data in the source domain. Each instance in the source domain is placed into a neural network trained by limited labelled target data to determine its contribution level based on errors. In [29], a novel aggregation method was defined for transfer learning that estimates and weights the average confidence probability of the source task on its similarity to the target task. Zuo et al. [30], [31] proposed a transfer learning method by using deep learning to extract hierarchical feature spaces, such that the knowledge in various feature spaces with different levels of abstraction from source domain can be explored and transferred to the target domain. In [32], Behbood et al. developed a fuzzy refinement domain adaptation method for long-term bank failure prediction by using similarity/dissimilarity concepts to modify the label values of samples in the target domain.

3) When to transfer: concerns circumstances in which knowledge transfer can or cannot be done. For example, in some scenarios, if the source and target domains are not related, brute force transfers may not work or might even damage the performance of learning in the target domain. This is also known as 'negative transfer' [33], [34]. An ideal transfer learning method would benefit from related domains or tasks but would avoid negative transfers. In this work, since we only focus on handling datasets that contain both complete and incomplete samples; the source domain is only comprised of complete samples; and the target domain is the entire dataset, the distributions of these two domains obviously remain similar. The ideal classifier built on the target domain should stay as similar as possible to the classifier built on the source domain. In other words, a transfer learning methodology can be used for missing data classification, especially in circumstances where it is easy to construct a classifier in the source domain such that model transfer learning can be applied.

Overall, this work expands on the three aspects discussed above. We propose a model-based transfer learning method that learns from the constructed model in the source domain (*what to transfer*), then leverages that model knowledge onto the target model (*how to transfer*). The similarity between the two domains is guaranteed, and the correlation between these two models is automatically evaluated (*when to transfer*).

III. A MODEL TRANSFER-BASED ADDITIVE LS-SVM CLASSIFIER

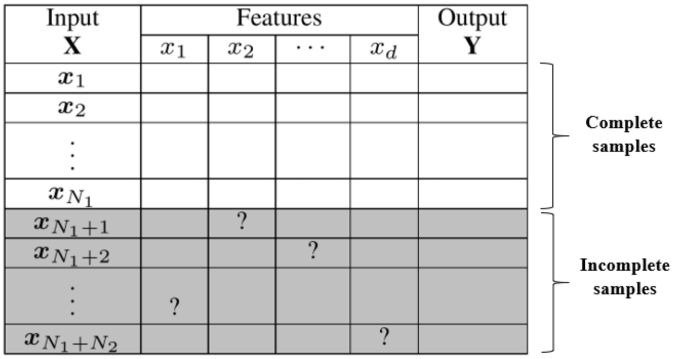
A. Dataset Representation

In this work, the dataset is denoted as $\mathbf{S}$ with N samples in total. The input set is denoted as X, with the corresponding output set as $\boldsymbol{Y}$, where $\mathbf{S} = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$, $\boldsymbol{x}_l = (x_1^l, x_2^l, ..., x_d^l) \in \boldsymbol{X} \subset \mathbf{R}^d$ and $y_l \in \boldsymbol{Y} = \{+1, -1\}$. The input set $\boldsymbol{X}$ is associated with two classes labeled +1 and -1, stored in the output set $\boldsymbol{Y}$, and each sample $\boldsymbol{x}_i$ contains d features.

The dataset $\mathbf{S}$ consists of two data portions of data ($N = N_1 + N_2$), N_1 includes the complete data samples $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{N_1})$ and N_2 includes the incomplete data samples ($\boldsymbol{x}_{N_1+1}$,$\boldsymbol{x}_{N_1+2}$,...,$\boldsymbol{x}_{N_1+N_2}$). We want to find a decision function $f : \boldsymbol{X} \rightarrow \boldsymbol{Y}$, that finds the matching y for any new incoming sample y for any new incoming sample $\boldsymbol{x}$. Fig. 1 describes the dataset $\mathbf{S}$ where missing values are denoted by the symbol ?.

B. Framework of proposed classifier

The framework of the proposed transfer-based additive LS-SVM is illustrated in Fig. 2. The source domain contains complete data N_1, and the target domain contains both N_1 and incomplete data N_2. We first construct an additive LS-SVM for the source domain and then a transfer-based additive LS-SVM classifier is constructed for classification in the target domain containing the missing data.



Input $\mathbf{X}$	Features x_1	x_2	$\cdots$	x_d	Output $\mathbf{Y}$
$\boldsymbol{x}_1$					
$\boldsymbol{x}_2$					
$\vdots$					
$\boldsymbol{x}_{N_1}$					
$\boldsymbol{x}_{N_1+1}$		?			
$\boldsymbol{x}_{N_1+2}$			?		
$\vdots$	?				
$\boldsymbol{x}_{N_1+N_2}$				?	

Complete samples

Incomplete samples

Fig. 1: Dataset representation

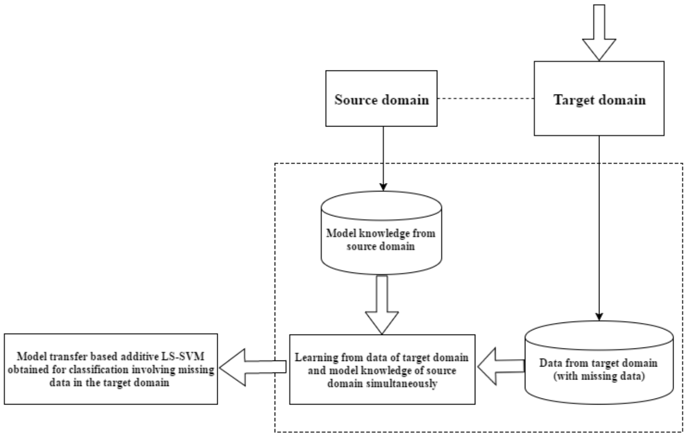

Source domain
Target domain
Model knowledge from source domain
Data from target domain (with missing data)
Learning from data of target domain and model knowledge of source domain simultaneously
Model transfer based additive LS-SVM obtained for classification involving missing data in the target domain

Fig. 2: Framework of the transfer-based additive LS-SVM

C. Adaptive Regularization

In order to find the function $\mathcal{H}$ in the hypothesis space which approximates the unknown decision in the hypothesis space, which approximates the unknown decision function f, the described learning process can be formalized as an optimization problem, which minimizes the structural risk:

$$\eta\Omega(f) + R_{emp}(f(\boldsymbol{x_l}), y_l) \tag{1}$$

where $\eta > 0$ is a regularization parameter which balances good generalization performance with the smoothness or simplicity enforced by a small $\Omega(f)$. The empirical risk $R_{emp}(f)$ can be those using squared loss or the ϵ-insensitive loss. To maximize the margin of classification in the feature space using the regularization term $\frac{1}{2}\|\boldsymbol{w}\|^2$, we get

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + R_{emp}(f(\boldsymbol{x_l}), y_l) \tag{2}$$

In our framework, the distribution P_s in the source domain and the distribution P_t in the target domain are related, and the model on each domain shares similarity to some extent. Thus, the model knowledge learned from the source domain can be leveraged to help the learning process in the target domain. For example, we can first find the optimal $\boldsymbol{w}_s$ by minimizing Eq. (2) in the source domain. When we encounter a new target domain, we can construct a model in which $\boldsymbol{w}_t$ gets as close as possible to the known $\boldsymbol{w}_s$. Through editing the regularization term, the learning classification task becomes

$$\frac{1}{2}\|\boldsymbol{w}_t - \boldsymbol{w}_s\|^2 + R_{emp}(f(\boldsymbol{x_l}), y_l) \tag{3}$$

where $f(\boldsymbol{x}_l)$ on the target domain is parameterized in terms of $\boldsymbol{w}_t$.

In addition, to evaluate the similarity between $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$ (*when to transfer*) in the optimization problem above, we can further edit the regularization term into $\|\boldsymbol{w}_t - \lambda\boldsymbol{w}_s\|$ by adding the weighting factor λ.

D. Transfer-based additive LS-SVM classifier

To construct our proposed transfer-based additive LS-SVM classifier for missing data, we use $\lambda\boldsymbol{w}_s$ as reference in the regularization term in Eq. (3), and the square loss $R_{emp}(f(\boldsymbol{x}_l), y_l) = (f(\boldsymbol{x}_l) - y_l)^2$. Moreover, the upper bound of the classification error caused by each incomplete sample with missing values in the input space of the target domain is denoted as c_l. In this case, λ and c_l are treated as the learning parameters. They are selected by the fast leave-one-out cross validation strategy which will be discussed later. The objective function based on LS-SVM framework is minimized to

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}(\boldsymbol{w} - \lambda\boldsymbol{w}_s)^2 + \frac{C}{2}\sum_{l=1}^{N}(\xi_l - c_l)^2 \\ \text{s.t.} \quad & y_l = \sum_{j=1}^{d} w_j \phi(x_j^l) I_j^l + b + \xi_l \\ & l = 1, 2, ..., N_1 + N_2 (= N) \end{aligned} \tag{4}$$

where

$$I_j^l = \begin{cases} 1 & \text{if feature j of the } l\text{-th sample has value} \\ 0 & \text{if feature j of the } l\text{-th sample has no value} \end{cases}$$

Since $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{N_1})$ is a group of the complete data, I_j^l $(l = 1, 2, ..., N_1)$ is set to 1, and c_l $(l = 1, 2, ..., N_1)$ is set to 0 accordingly. Also, $\tilde{\phi}(\boldsymbol{x}_l) = (\phi(x_1^l), \phi(x_2^l), ..., \phi(x_j^l), ..., \phi(x_d^l))$ and it is a feature mapping such that the kernel K below can be adopted in Eq. (4).

$$K(\boldsymbol{x}_l, \boldsymbol{x}_k) = \tilde{\phi}(\boldsymbol{x}_l)^T \tilde{\phi}(\boldsymbol{x}_k) = \sum_{j=1}^{d} k(x_j^l, x_j^k) \tag{5}$$

where

$$k(x_j^l, x_j^k) = \begin{cases} \tilde{k}(x_j^l, x_j^k) & \text{both } x_j^l \text{ and } x_j^k \text{ are not missing} \\ 0 & \text{otherwise} \end{cases}$$

$\tilde{k}(x_j^l, x_j^k)$ is a kernel function. In this study, a Gaussian function is used as the kernel, i.e., $\tilde{k}(x_j^l, x_j^k) = e^{\frac{-(x_j^l - x_j^k)^2}{\sigma^2}}$, where σ is the kernel width. It is obvious that $K(\boldsymbol{x}_l, \boldsymbol{x}_k)$ in Eq. (5) is an additive Gaussian kernel [35].

The Lagrangian L of Eq. (4) is given by

$$L = \tfrac{1}{2}(\boldsymbol{w} - \lambda\boldsymbol{w}_s)^2 + \tfrac{C}{2}\textstyle\sum_{l=1}^{N}(\xi_l - c_l)^2 + \sum_{l=1}^{N}\alpha_l(y_l - \sum_{j=1}^{d} w_j \phi(x_j^l) I_j^l - b - \xi_l) \tag{6}$$

where $\boldsymbol{\alpha} \in \mathbf{R}^N$ is the vector of all Lagrangian multipliers. With respect to $\boldsymbol{w}$, ξ_i, b, α_i, the optimality condition can be calculated by

$$\tfrac{\partial L}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \lambda \boldsymbol{w}_s + \textstyle\sum_{l=1}^{N} \alpha_l (I_1^l \phi(x_1^l), I_2^l \phi(x_2^l), ..., I_d^l \phi(x_d^l)) \quad (7)$$

$$\frac{\partial L}{\partial \xi_l} = 0 \quad \Rightarrow \xi_l = \alpha_l / C + c_l \quad (8)$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \sum_{i=1}^{N} \alpha_l = 0 \quad (9)$$

$$\frac{\partial L}{\partial \alpha_l} = 0 \quad \Rightarrow y_l = \sum_{j=1}^{d} w_j \phi(x_j^l) I_j^l + b + \xi_l \quad (10)$$

Combining Eq. (7), Eq. (8) with Eq. (10), we get

$$\textstyle\sum_{l=1}^{N} \sum_{j=1}^{d} \alpha_l I_j^k I_j^l \phi(x_j^k) \phi(x_j^l) + b + \boldsymbol{\alpha}_l / C = y_l - \lambda \sum_{j=1}^{d} w_j I_j^l \phi(x_j^l) - c_l \quad (11)$$

Based on a kernel trick in Eq. (5), we can replace $\tilde{\phi}(\boldsymbol{x}_l)^T \tilde{\phi}(\boldsymbol{x}_k)$ by $K(\boldsymbol{x}_l, \boldsymbol{x}_k)$. We can further write the linear equation of Eq. (11) in matrix form:

$$\begin{bmatrix} \mathbf{K} + \frac{1}{C}\boldsymbol{\Lambda} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} - \boldsymbol{\gamma} \\ 0 \end{bmatrix} \quad (12)$$

where $\boldsymbol{\Lambda}$ is a matrix in which each diagonal entry is one and all other entries are zero, $\boldsymbol{y}$ is the real label vector of all the samples in the training dataset and

$$\boldsymbol{\gamma} = \begin{pmatrix} \lambda \sum_{j=1}^{d} w_j^s I_j^1 \phi(x_j^1) \\ \lambda \sum_{j=1}^{d} w_j^s I_j^{N_1} \phi(x_j^{N_1}) \\ \lambda \sum_{j=1}^{d} w_j^s I_j^{N_1+1} \phi(x_j^{N_1+1}) + c_{N_1+1} \\ \lambda \sum_{j=1}^{d} w_j^s I_j^{N_1+2} \phi(x_j^{N_1+2}) + c_{N_1+2} \\ \vdots \\ \lambda \sum_{j=1}^{d} w_j^s I_j^{N} \phi(x_j^{N}) + c_N \end{pmatrix}$$
$$= \lambda \begin{pmatrix} \sum_{j=1}^{d} w_j^s I_j^1 \phi(x_j^1) \\ \vdots \\ \sum_{j=1}^{d} w_j^s I_j^{N_1} \phi(x_j^{N_1}) \\ \sum_{j=1}^{d} w_j^s I_j^{N_1+1} \phi(x_j^{N_1+1}) \\ \sum_{j=1}^{d} w_j^s I_j^{N_1+2} \phi(x_j^{N_1+2}) \\ \vdots \\ \sum_{j=1}^{d} w_j^s I_j^{N} \phi(x_j^{N}) \end{pmatrix} + c_{N_1+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + c_{N_1+2} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + c_N \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \quad (13)$$

Since $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{N_1})$ is a group of the complete data, c_l $(l = 1, \cdots, N_1)$ should be $\mathbf{0}$. Thus, we do not represent them in the above formula. Our goal is to evaluate c_l $(l = N_1 + 1, \cdots, N)$ and λ using the proposed fast leave-one-out cross validation.

We can rewrite Eq. (12) into

$$\begin{bmatrix} \mathbf{K} + \frac{1}{C}\boldsymbol{\Lambda} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} - \lambda \boldsymbol{I}_1 - c_{N_1+1} \boldsymbol{I}_2 - c_{N_1+2} \boldsymbol{I}_3 - \cdots - c_N \boldsymbol{I}_{N_2+1} \\ 0 \end{bmatrix}$$
$$= \begin{bmatrix} \boldsymbol{y} - \sum_{l=1}^{N_2+1} \beta_l \boldsymbol{I}_l \\ 0 \end{bmatrix} \quad (14)$$

where $\boldsymbol{\beta} = (\lambda, c_{N_1+1}, c_{N_1+2}, \cdots, c_N)$, and

$$\boldsymbol{I}_1 = \begin{pmatrix} \sum_{j=1}^{d} w_j^s I_j^1 \phi(x_j^1) \\ \sum_{j=1}^{d} w_j^s I_j^{N_1} \phi(x_j^{N_1}) \\ \sum_{j=1}^{d} w_j^s I_j^{N_1+1} \phi(x_j^{N_1+1}) \\ \sum_{j=1}^{d} w_j^s I_j^{N_1+2} \phi(x_j^{N_1+2}) \\ \vdots \\ \sum_{j=1}^{d} w_j^s I_j^{N} \phi(x_j^{N}) \end{pmatrix}, \ \boldsymbol{I}_2 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ \boldsymbol{I}_3 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ -1 \\ \vdots \\ 0 \end{pmatrix}, \cdots, \ \boldsymbol{I}_{N_2+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ -1 \end{pmatrix}$$

Finally, we use $\mathbf{H}$ to represent the first matrix on the left hand of Eq. (14), the model parameters can be calculated simply using a matrix inversion:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \mathbf{Q} \begin{bmatrix} \boldsymbol{y} - \sum_{l=1}^{N_2+1} \beta_l \boldsymbol{I}_l \\ 0 \end{bmatrix} \quad (15)$$

where $\mathbf{Q} = \mathbf{H}^{-1}$. Also, if we can get β_l for all the samples with missing values, $\boldsymbol{\alpha}$ and b can be calculated accordingly from Eq. (15), and hereby $\boldsymbol{w}$ and b from Eq. (7) and Eq. (10) respectively. Therefore, for a new input sample $\boldsymbol{x}_t$, we can obtain the predicted label y_t using the decision function $y_t = \boldsymbol{w}^T \tilde{\phi}(\boldsymbol{x}_t) + b$.

We can also extend the proposed additive LS-SVM classifier explained above for multi-classification tasks. This one-against-all strategy is used to find the multiple decision functions that separate one class from the remaining classes. In the end, the predicted label of the new input data sample $\boldsymbol{x}_t$ is determined by $\max\limits_{k=1,...,M} y_k(\boldsymbol{x}_t)$, where M denotes the number of the classes.

E. Decision Function

After determining the value of $\boldsymbol{\alpha}$ in Eq. (12) with the selected value of $\boldsymbol{\beta}$, the optimal solution becomes

$$\boldsymbol{w} = \lambda \boldsymbol{w}_s + \sum_{l=1}^{N} \alpha_l (I_1^l \phi(x_1^l), I_2^l \phi(x_2^l), ..., I_d^l \phi(x_d^l)) \quad (16)$$

Therefore, the decision function for the new sample $\boldsymbol{x}_t$ is

$$\begin{aligned} f(\boldsymbol{x}_t) &= \sum_{j=1}^{d} \left(\lambda w_{sj} + \sum_{l=1}^{N} \alpha_l I_j^l \phi(x_j^l) \right) \phi(x_j^t) + b \\ &= \sum_{j=1}^{d} \left(\lambda w_{sj} \phi(x_j^t) + \sum_{l=1}^{N} \alpha_l I_j^l k(x_j^l, x_j^t) \right) + b \end{aligned} \quad (17)$$

IV. Fast leave-one-out cross validation for parameters

A. Fast leave-one-out cross validation for parameters

From the last section, it is clear that the classification performance of the proposed transfer-based additive LS-SVM classifier relies on determining the parameter $\boldsymbol{\beta}$. Traditionally, a cross-validation method is used as an unbiased estimator to determine the parameters in the model, but this is computationally expensive and time-consuming. In this work, we propose a fast version of the leave-one-out cross-validation method to find the optimal value of $\boldsymbol{\beta}$ in Eq. (15), and it is this approach that is discussed in this section.

We decompose $\mathbf{H}$ into its block representation and isolate the first row and first column, i.e.,

$$\mathbf{H} = \begin{bmatrix} \mathbf{K} + \frac{1}{C}\boldsymbol{\Lambda} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} = \begin{bmatrix} h_{11} & \mathbf{h}_1^T \\ \mathbf{h}_1 & \mathbf{H}_{(-1)} \end{bmatrix} \quad (18)$$

$\alpha_{(-i)}$ and $b_{(-i)}$ represents the model parameters in the i-th iteration of the leave-one-out cross validation procedure. In the first iteration, where the first training sample is excluded, we have

$$\begin{bmatrix}\boldsymbol{\alpha}_{(-1)}\\ b_{(-1)}\end{bmatrix} = \mathbf{Q}_{(-1)}\begin{bmatrix}\boldsymbol{y}_{(-1)} - \sum_{l=1}^{N_2+1}\beta_l \boldsymbol{I}_{l(-1)}\\ 0\end{bmatrix} \tag{19}$$

where $\mathbf{Q}_{(-1)} = \mathbf{H}_{(-1)}^{-1}$ and $\boldsymbol{y}_{(-1)} = (y_2, y_3, ..., y_N, 0)^T$. We denote the predicted label on the i-th sample excluded from the training dataset by $\tilde{y}_i$, and the predicted label for the first training sample becomes

$$\begin{aligned}\tilde{y}_1 &= \mathbf{h}_1^T\begin{bmatrix}\boldsymbol{\alpha}_{(-1)}\\ b_{(-1)}\end{bmatrix} + \sum_{l=1}^{N_2+1}\beta_l I_{l1}\\ &= \mathbf{h}_1^T\mathbf{Q}_{(-1)}\left(\boldsymbol{y}_{(-1)} - \sum_{l=1}^{N_2+1}\beta_l \boldsymbol{I}_{l(-1)}\right) + \sum_{l=1}^{N_2+1}\beta_l I_{l1}\end{aligned} \tag{20}$$

where I_{l1} represents the first element of $\boldsymbol{I}_l$. Considering the last N equations in Eq. (12), we get $\begin{bmatrix}\mathbf{h}_1 \; \mathrm{H}_{(-1)}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}^T \; b\end{bmatrix}^T = \left(\boldsymbol{y}_{(-1)} - \sum_{l=1}^{N_2+1}\beta_l \boldsymbol{I}_{l(-1)}\right)$, and

$$\begin{aligned}\tilde{y}_1 &= \mathbf{h}_1^T\mathbf{Q}_{(-1)}\begin{bmatrix}\mathbf{h}_1 \, \mathbf{H}_{(-1)}\end{bmatrix}[\alpha_1, \cdots, \alpha_N, b]^T + \sum_{l=1}^{N_2+1}\beta_l I_{l1}\\ &= \mathbf{h}_1^T\mathbf{Q}_{(-1)}\mathbf{h}_1\alpha_1 + \mathbf{h}_1^T[\alpha_2, \cdots, \alpha_N, b]^T + \sum_{l=1}^{N_2+1}\beta_l I_{l1}\end{aligned} \tag{21}$$

From Eq. (12), the first equation of the system is $y_1 - \sum_{l=1}^{N_2+1}\beta_l I_{l1} = h_{11}\alpha_1 + \mathbf{h}_1^T[\alpha_2, \alpha_3, \cdots, \alpha_N, b]^T$, and hence $\tilde{y}_1 = y_1 - \alpha_1(h_{11} - \mathbf{h}_1^T\mathbf{Q}_{(-1)}\mathbf{h}_1)$. Finally, by using $\mathbf{Q} = \mathbf{H}^{-1}$ and the block matrix inversion lemma we can obtain

$$\mathbf{Q} = \begin{bmatrix} v^{-1} & -v^{-1}\mathbf{h}_1\mathbf{Q}_{-1}\\ \mathbf{Q}_{(-1)} + v^{-1}\mathbf{Q}_{(-1)}\mathbf{h}_1^T\mathbf{h}_1\mathbf{Q}_{(-1)} & -v^{-1}\mathbf{Q}_{(-1)}\mathbf{h}_1^T\end{bmatrix} \tag{22}$$

where $v = h_{11} - \mathbf{h}_1^T\mathbf{Q}_{(-1)}\mathbf{h}_1$. Since the system of linear equations in Eq. (12) is insensitive to permutations of the ordering of the equations, then

$$\tilde{y}_i = y_i - \alpha_i/\mathbf{Q}_{ii} \tag{23}$$

By defining $\left[\boldsymbol{\alpha}^{'T}, b^{'}\right]^T = \mathbf{Q}\left[\boldsymbol{y}^T, 0\right]$, $\left[\boldsymbol{\alpha}^{''T}, b^{''}\right] = \mathbf{Q}\left[\boldsymbol{I}_l^T, 0\right]$, and $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{'} - \sum_{l=1}^{N_2+1}\beta_l\boldsymbol{\alpha}_l^{''}$, then we can get

$$\tilde{y}_i = y_i - \frac{\alpha_i^{'}}{\mathbf{Q}_{ii}} + \frac{\sum_{l=1}^{N_2+1}\beta_l\alpha_{li}^{''}}{\mathbf{Q}_{ii}} \tag{24}$$

It can be seen from (24) that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ has a linear relationship, which means that after determining $\boldsymbol{\beta}$, the learning model can be obtained as well. The optimal $\boldsymbol{\beta}$ is supposed to keep the same sign of $\tilde{y}_i$ and y_i for all samples in the training dataset. However, it might bring many local minima issues due to non-convex formulation. Thus, in the end we adopt the following loss function, which is similar to the hinge loss:

$$(\tilde{y}_i, y_i) = |1 - \tilde{y}_i y_i|_+ = \left| y_i\frac{\alpha_i^{'} - \sum_{l=1}^{N_2+1}\beta_l\alpha_{li}^{''}}{\mathbf{Q}_{ii}}\right|_+ \tag{25}$$

where $|x|_+ = \max\{0, x\}$. This is a convex upper bound to the leave-one-out misclassification loss, and it prefers solutions in which $\tilde{y}_i$ has an absolute value equal to or bigger than 1 and the same sign as y_i. Finally, the objective function is

$$\begin{aligned}&\sum_{i=i}^{N} l(\tilde{y}_i, y_i)\\ \text{s.t} \quad &\|\boldsymbol{\beta}\|_2 \le D\end{aligned} \tag{26}$$

where D is a constant and $\|\cdot\|_2$ is the L_2 norm (Euclidean) in the constraint. A regularization based on this can induce numerical stability. This optimization process can be implemented by a projected sub-gradient descent algorithm and the pseudocode is given in Algorithm 1, in which $\boldsymbol{\beta}_1$ denotes λ in Eq. 4 and so $\boldsymbol{\beta}_1$ should be positive.

Algorithm 1: Projected Sub-gradient Descent Algorithm

```
Input: $\boldsymbol{\alpha}^{'}$, $\boldsymbol{\alpha}_k^{''}$ and $\boldsymbol{I}$
Initialize: $\boldsymbol{\beta} \leftarrow \mathbf{0}$ and $t \leftarrow 1$
Repeat
$\tilde{y}_i = y_i - \frac{\alpha_i'}{\boldsymbol{Q}_{ii}} + \frac{\sum_{l=1}^{N_2+1}\beta_l\alpha_{li}^{''}}{\boldsymbol{Q}_{ii}}$, $i = 1, 2, ..., N$
$d_i \leftarrow \mathbf{1}\{\tilde{y}_i y_i > 0\}$, $i = 1, 2, ..., N$
$\beta_l \leftarrow \beta_l - \frac{1}{\sqrt{t}}\sum_{i=1}^{N} d_i y_i \frac{\alpha_{li}^{''}}{Q_{ii}}$, $l = 1, 2, ..., N_2 + 1$
If $\|\boldsymbol{\beta}\|_2 > D$ then $\boldsymbol{\beta} \leftarrow \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|_2}D$
End if
$\boldsymbol{\beta}_1 \leftarrow max(\boldsymbol{\beta}_1, 0)$,$l = 1, 2, ..., d$
$t \leftarrow t + 1$
Until convergence
Output: $\boldsymbol{\beta}$
```

B. Computational complexity

One highlight in the proposed transfer-based additive SVM classifier is its fast computational ability. Its computational cost contains three parts, which can be represented as $O(N_1^3 + N^3 + N(N_2+1))$. The first part includes the model knowledge obtained using LS-SVM on the source domain N_1. Therefore, the complexity of this part is $O(N_1^3)$, which is the complexity of LS-SVM. The second part includes the calculation of the matrix $\mathbf{Q}$ by the inverse related to the training dataset on the target domain, and so the corresponding computational complexity becomes $O(N^3)$. The third part includes the computational complexity of each iteration in the Algorithm 1 to optimize Eq. (26), which can be represented as $O((N_2+1)N)$.

Let us consider the traditional cross-validation strategy. If a standard LS-SVM is adopted and T ($\geq$ 3) grid values for each parameter are simply considered, the whole time complexity would become $O(N_1^3 + (N^3 * N)^{T(N_2+1)}) = O(N_1^3 + N^{4T(N_2+1)})$ which is much more computationally expensive, and even impractical, than $O(N_1^3 + N^3 + N(N_2+1))$ occupied by the proposed fast cross-validation strategy.

C. Interpretation of parameter c_l and data cleaning

The obtained parameter c_l $(l = N_1 + 1, N_1 + 2, \cdots, N1 + N2)$ of each sample with missing value(s) tells us the relative influence level of the classification error caused by those data samples, which accordingly helps us to clean the training set data.

If $|c_l|$ or $\max\limits_{k=1,...,M}|c_l^k|$ of the l-th incomplete sample is greater than a given small positive threshold, the influence on the classification error from this incomplete sample is serious and should be cleaned from the dataset. Inversely, if $|c_l|$ or

TABLE I: Dataset descriptions

Dataset	Number of samples	Features	Class	Class(%)
Surgery	470	17	F T	85.11 14.89
Diabetic	1151	19	0 1	46.92 53.08
Pima	769	8	0 1	65.02 34.98
Bupa	345	6	1 2	42.03 57.97
Breast	699	9	2 4	65.52 34.48
Titanic	887	6	0 1	61.44 38.56
German	1000	24	1 2	70.00 30.00

$\min\limits_{k=1,\ldots,M} |c_l^k|$ of the l-th incomplete sample is less than a given small positive threshold, the influence of the classification error from this incomplete sample is tolerable and can remain in the training dataset.

V. Experimental results

A. Datasets

In the experiments, seven public datasets (*Surgery, Diabetic, Pima, Bupa, Breast, Titanic* and *German*) were adopted. The original breast dataset has missing values, which were removed during data processing in order to fully control the missing data in our experiments. The rest of datasets are complete with no missing data. Table I summarizes the datasets adopted in this work.

B. Experimental Design

The main purpose of the experiments conducted in this work is to evaluate the performance of the proposed transfer-based additive LS-SVM classifier for missing data, compared to traditional missing data classification methods, denoted as follows:

(A) **Case deletion** all samples with missing values were removed.

(B) **Mean imputation** missing values for a certain feature were replaced with the mean of values of complete samples for that feature.

(C) **KNN imputation** missing values were replaced with the weighted mean of the k nearest-neighbour columns.

Using the proposed method, missing data was assembled by constructing a classifier. Using the comparative methods (A), (B) and (C), missing data were first manipulated, and then both standard LS-SVM and SVM classifiers were used on the processed data for model construction. To make the comparison fair, we adopted the additive Gaussian kernel on both proposed and comparative methods. We first calculated the standard deviation of each feature in the dataset and then took their average value as $\overline{\sigma}$. Accordingly, we established a trade-off parameter C and a Gaussian kernel parameter σ by searching $C \in \{1, 10, 50, 100, 1000, 10000\}$ and $\sigma \in \{\overline{\sigma}/16, \overline{\sigma}/8, \overline{\sigma}/4, \overline{\sigma}/2, \overline{\sigma}, 2\overline{\sigma}, 4\overline{\sigma}, 8\overline{\sigma}, 16\overline{\sigma}\}$. Additionally, we obtained $\boldsymbol{w}_s$ from the source domain for the proposed model transfer method in advance. Finding an optimal value for the neighbouring parameter k for the method (C) was a major issue. The missing values were filled using estimated values from their 1, 3, 7, 9 and 10 nearest neighbours. Due to the space limitations, we only show results from the 3 and 10 nearest neighbours, identified as KNN3 and KNN10 respectively in this work. All the experiments were implemented using 64-bit MATLAB on a computer with an Intel Core i5-6300 2.40 GHz CPU and 8.00GB RAM.

Missing data were artificially inserted in different features with different proportions into the public datasets. We first selected the first, second and third most relevant feature(s) using wrapper and filter techniques, then modified their values to unknown. Doing this allowed us to consider that less relevant or non relevant features might not contribute to classifier construction or even compromise the experimental analysis. We also inserted various proportions of missing data in the datasets (10%, 20%, 30%, 40%, 50%, 60%) such that we could analyse the corresponding performance of the classifiers.

C. Classification performances

The 10-fold cross validation strategy was used in the experiments for performance evaluation, to ensure that every sample from the dataset had a chance to be used in the training and testing sets. Here, the dataset was randomly divided into ten subsets. The model was built using nine subsets and tested on the remaining one. This process was repeated 10 times, and the mean and standard deviation of accuracy in the 10-fold cross validation procedure was calculated.

Tables III-IX display the numerical experimental results of the proposed and comparative methods on seven public datasets in terms of accuracy. Figure 5 use line graphs to further demonstrate the change tendencies of performances with different missing data rates. In order to detect significant differences among the performances of the proposed and comparative methods, we also carried out the Friedman ranking test followed by Holm post-hoc test [36], [37] for multiple comparisons on seven datasets. The Friedman ranking test was used to evaluate whether there was a statistically significant difference among all the methods. If the p-value is smaller than 0.05, the null hypothesis is rejected and there is significant difference. The Holm post-hoc test was used to further verify if there was a statistical difference between the best Friedman ranking method and each of the rest, and the hypothesis of equivalence of the methods is rejected if $p < \alpha/i$. Tables XII and XIII list the corresponding statistical results about Friedman ranking test and Holm post-hoc test, respectively. According to these results, we make the following observations:

(1) In most cases, our proposed classifier achieved better classification performances than those using other comparative methods. This indicates that our proposed classifier, by leveraging the knowledge learned from the model on the source domain to the target domain, has the ability to perform classification with missing data and achieve advantageous performances compared with the traditional missing data treatments followed by LS-SVM or SVM.

(2) In very few cases, with a specific combination of the missing data rate and missing feature(s), the performance results of our proposed method were lower than those using the case deletion method. For example, in Table VII, when there were 40% missing data in the *Breast* dataset, (case deletion + SVM) achieved marginally higher accuracies than the other methods. The similar situation occurs in Table IV, when there were 20% missing data in the *Diabetic* dataset. This might be due to the reason that those randomly selected missing data coincidently had the noise and thus data removal enhanced the classification performance, particularly of the SVM which suffers from the noise sensitivity problem. Also, there are few cases in Tables III and VI that the proposed method was beaten by (KNN3+SVM) and (KNN10+SVM). We noticed that these usually occurred when the missing data rate was comparatively higher ($\geq 30\%$), which may greatly fluctuate the classification performance. In Tables XII and XIII, there are significant differences between the proposed method and all the comparative methods except (case deletion + SVM) ($0.171857 > 0.05$) in terms of accuracy, we must notice that the proposed classifier also has the advantage on data cleaning via the fast leave-one-out cross validation strategy, which case deletion and all other imputation methods cannot achieve. Further details are discussed in the *MIHC* case study.

VI. A CASE STUDY

A. Data collection

A nurse-led mobile integrative health centre (MIHC) [38] in Hong Kong provides free health screening services for elderly people. They house a local database and server to provide a computer service and store data for the clinic.

In August 2013, a dataset was collected which contains the records of 444 patients, each made up of 33 features. Because of the nature of both the tests performed and the patients' themselves, some information is missing. For instance, certain tests proved too physically or cognitively taxing for some elderly patients; sometimes language barriers prevent the nurses from communicating clearly with the patients, etc. The dataset contains demographic, socioeconomic, social relationship, and social participation data. Additionally, information on the patients' health history, such as smoking and drinking habits, chronic illnesses, and data from a series of health assessments with descriptions is also included, as shown in TABLE II.

B. Data processing

The range of values recorded under the World Health Organization questionnaire on quality of life: short form Hong Kong version (WHOQOL-BREF(HK)) framework [39], [40] lacked extreme values for an overall quality of life score on a 1 to 5 scale. Therefore, some data pre-processing was required. To avoid unintended bias in the training set, these values were re-mapped to a scale of 3, where "1" indicates poor, "2" indicates neutral, and "3" indicates good quality of life.

C. The challenge

Using the 33 features inherent in the *MIHC* dataset, we intended to construct a classifier to predict the quality of life of elderly patients using the same scale mentioned above - poor, neutral, and good. However, in this dataset 14 of the 33 features, and 159 of the 444 patient records, contain missing values, which presents problems for constructing a prediction model.

D. The solution and analysis

In this case study, the proposed transfer-based additive LS-SVM classifier was applied to the dataset to predict quality of life and the results are compared to the same methods described in the Section V-B. Table X and Fig. 3 demonstrate that the proposed classifier provided the best classification performance with the accuracy 0.7258 among all the methods. The running time of the proposed method which had the fast leave-one-out cross validation was 4.03 seconds. Thus, in this practical application, the proposed transfer-based additive LS-SVM classifier outperforms both conventional methods and the standard LS-SVM classifier for missing data classification.

Additionally, as discussed in IV-C, the influence of each incomplete sample in the training dataset can be determined by $|c_l|$ (binary classification) or $\max\limits_{k=1,2,3} |c_l^k|$ (the multi-class classification) obtained during the classification process. We performed data cleaning on the *MIHC* dataset and observed the corresponding classification results on the cleaned dataset. Fig. 4 shows $\max\limits_{k=1,2,3} |c_l^k|$ of each incomplete sample in the *MIHC* training dataset. We can observe that the $\max\limits_{k=1,2,3} |c_l^k|$ ranged from 0 to 1.8499. In fact, the $\max\limits_{k=1,2,3} |c_l^k|$ of all the samples were below 1 except one (1.8499), which indicated that this incomplete sample had a comparatively big influence on the classification error and must be removed. Based on the range of these values, the threshold was set to 0.6, 0.65, 0.80, 1.00. The l-th incomplete sample whose $\max\limits_{k=1,2,3} |c_l^k|$ was bigger than the chosen threshold was then removed and the corresponding performance results were displayed in Table. XI. We observe that the performance after data cleaning was maintained or improved, to a certain extent, by given different thresholds. The best classification accuracy achieved was 0.7327 when the incomplete samples with $\max\limits_{k=1,2,3} |c_l^k|$ greater than 0.8 were removed. This result shows that the proposed method has the ability to clean unnecessary incomplete samples in the dataset based on the influence of classification errors in practical applications.

E. Contribution

Due to the complex nature of the way the MIHC acquires data, any automated predictive algorithm that could decrease the workload for staff nurses would be valuable. More importantly, given the patient type, it is highly likely that future datasets from the MIHC will contain missing data and any over-collection of data will only increase the likelihood of missing features. As previously mentioned, a common reason for missing features is a loss of patience by the patient or an inability to communicate. Using the proposed method, it is possible to perform classification directly on the missing data. Moreover, unnecessary samples are automatically removed by

TABLE II: Health related assessments and questionnaire on MIHC

Title	Description
Bio-measurements	Major vital signs of the patients were measured, e.g. body temperature, pulse rate, oxygen saturation (SpO2), blood pressure and waist-hip ratio (WHR).
Berg Balance Scale (BBS)	Patient balance ability measured using a metric established using 14 tests. These tests include having the patient stand up from a sitting position and other more taxing balance tests e.g. standing on one foot. From this, a score between 0 to 4, with 4 being the highest, is assigned to each completed test. This gives an overall rating ranging from 0 to 56 for the patients overall balance ability.
Timed Up and Go Test (TUG)	A physical test to measure basic functional mobility, this test is well known and has good reliability. This test required that the patient would, starting from a sitting position, rise from a chair and walk for 3 meters, then turn around and return to the original sitting position. Patients were required to repeat the task three times to establish a best time.
Visual Analogue Scale (VAS) for pain	A scale used to measure the extent of pain to which a client felt localized at the most painful part of the body. Using a 10 cm vertical line, patients indicate visually somewhere from the lower end to upper end of this line the extent of their experienced pain: ranging from no pain to unbearable pain from top to bottom of the line respectively.
The 30-second Chair Stand Test (30-s CST)	Designed to test lower body strength and endurance, specifically when undertaking demanding tasks found in daily life. Intended to measure the patients ability to perform daily tasks such as climbing stairs, getting up from chairs or out of the bath tub. The test required that the patient repeatedly rise from a chair to a fully standing position and then sit down again. The number of times that the patient was able to perform this task within a 30 second window was recorded.
Body Composition Analysis	Used to determine patient levels of fitness and obesity using body mass index (BMI), skeletal muscle mass, body fat mass and body fat percentage (BFP).
Handgrip Strength	Using a dynamometer the grip strength of both the dominant and non-dominant hand were measured taking the average over three trials. This test required that the patient squeeze the device with maximum effort from a standing position. A total of six trials, three for each hand, was performed and the average strength for both were recorded.
Quality of Life (QOL)	The QOL of the clients was measured using the WHOQOL-BREF(HK).

determining which samples have the least impact on the overall accuracy of the classification when they are missing from the dataset. Beyond assisting with data cleaning, determining a classifier that effectively handles the corrupt or missing data samples, may vastly improve the overall performance and effectiveness of the MIHC itself. If patients and practitioners are less concerned about fully complying with the rigours of the tests, it is likely that stress levels and therefore test times will decrease. As a result, interpersonal relationships improve, leading to increased participation and better overall accuracy, and the cycle perpetuates.

VII. Conclusions and further study

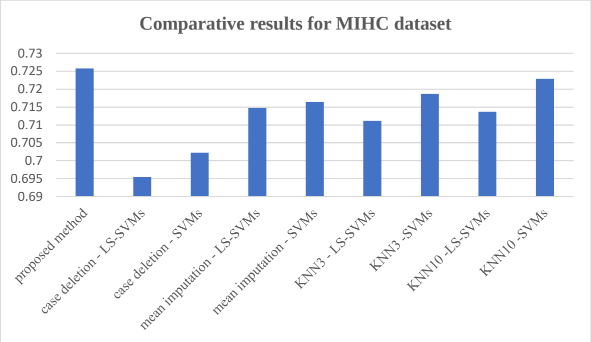


Fig. 3: Comparative results for the *MIHC* dataset

Missing data is an inevitable problem in many machine learning processes. In this paper, we present a novel additive LS-SVM classifier to handle missing data from a transfer learning perspective. The proposed classifier has the ability to learn model weights from the source domain of a dataset - the complete portion of the dataset - and transfer them to the target domain - the entire dataset with missing data. The classifier also simultaneously determines the influence of the classification error caused by each incomplete sample in the training dataset through a fast leave-one-out cross validation strategy. This provides an alternative approach to data cleaning to guarantee data quality. We compare our proposed classifier to both traditional missing data treatments and the LS-SVM classifier on seven public datasets. Experimental results confirm the effectiveness of the proposed method for classification problems for a wide range of missing data rates and columns in both training and testing datasets. Moreover, a case study on a community health dataset is presented in detail, which particularly highlights the benefits and contributions of the proposed method to this real world application.

Even though the proposed method shows promising performance, this study has some limitations which hold promise for future work. For example, missing values were filled randomly in this study. In future work, we can analyse the behaviour of methods when missing values are not randomly distributed, or randomly assigned, with different distributions. Using a case study was advantageous, as it allows for situations with a dataset with naturally occurring missing values, but it would be worth investigating the nature of this distribution for further indications as to how to later assign missing values into complete public datasets.

TABLE III: Performance results for the *Surgery* dataset

Missing rates	Missing feature - 1								
	proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	**0.8582±0.0123**	0.8402±0.0356	0.8466±0.0270	0.8440±0.0123	0.8482±0.0146	0.8511±0.0208	0.8489±0.0174	0.8475±0.0250	0.8511±0.0177
20%	**0.8573±0.0217**	0.8378±0.0262	0.8374±0.0275	0.8434±0.0178	0.8426±0.0279	0.8434±0.0227	0.8433±0.0273	0.8463±0.0178	0.8539±0.0207
30%	0.8534±0.0291	0.8267±0.0298	0.8387±0.0343	0.8424±0.0139	0.8411±0.0174	0.8427±0.0244	**0.8539±0.0201**	0.8428±0.0071	0.8411±0.0253
40%	**0.8485±0.0108**	0.8282±0.0350	0.8373±0.0316	0.8427±0.0113	0.8455±0.0274	0.8440±0.0165	0.8376±0.0355	0.8400±0.0256	0.8439±0.0198
50%	0.8465±0.0261	0.8330±0.0351	0.8386±0.0390	0.8345±0.0225	0.8433±0.0218	0.8369±0.0188	0.8389±0.0272	0.8417±0.0108	**0.8467±0.0250**
60%	**0.8463±0.0178**	0.8388±0.0365	0.8377±0.0389	0.8364±0.0108	0.8355±0.0208	0.8407±0.0148	0.8404±0.0239	0.8418±0.0183	0.8454±0.0081
	Missing features - 1, 10								
10%	**0.8581±0.0191**	0.8365±0.0267	0.8457±0.0324	0.8434±0.0108	0.8440±0.0198	0.8463±0.0148	0.8553±0.0146	0.8322±0.0249	0.8546±0.0244
20%	**0.8582±0.0188**	0.8514±0.0353	0.8566±0.0250	0.8471±0.0259	0.8511±0.0278	0.8440±0.0225	0.8461±0.0307	0.8374±0.0227	0.8504±0.0223
30%	**0.8487±0.0148**	0.8364±0.0222	0.8374±0.0355	0.8418±0.0205	0.8454±0.0281	0.8369±0.0309	0.8447±0.0244	0.8274±0.0228	0.8482±0.0280
40%	**0.8463±0.0178**	0.8365±0.0311	0.8293±0.0427	0.8392±0.0235	0.8440±0.0229	0.8369±0.0188	0.8411±0.0215	0.8317±0.0148	0.8461±0.0232
50%	**0.8440±0.0142**	0.8352±0.0326	0.8349±0.0414	0.8363±0.0205	0.8404±0.0227	0.8360±0.0071	0.8405±0.0230	0.8298±0.0213	0.8433±0.0178
60%	**0.8436±0.0041**	0.8332±0.0377	0.8326±0.0345	0.8358±0.0269	0.8324±0.0227	0.8329±0.0219	0.8433±0.0165	0.8334±0.0164	0.8424±0.0370
	Missing features -1, 10, 11								
10%	**0.8576±0.0148**	0.8433±0.0249	0.8460±0.0272	0.8406±0.0217	0.8440±0.0227	0.8369±0.0142	0.8454±0.0200	0.8440±0.0188	0.8475±0.0246
20%	**0.8572±0.0256**	0.8538±0.0306	0.8496±0.0262	0.8347±0.0108	0.8482±0.0328	0.8274±0.0148	0.8418±0.0223	0.8369±0.0156	0.8433±0.0251
30%	**0.8481±0.0164**	0.8255±0.0364	0.8267±0.0318	0.8298±0.0071	0.8468±0.0161	0.8340±0.0195	0.8433±0.0253	0.8311±0.0275	0.8426±0.0402
40%	**0.8445±0.0188**	0.8299±0.0317	0.8400±0.0359	0.8180±0.0205	0.8389±0.0253	0.8337±0.0164	0.8440±0.0253	0.8394±0.0209	0.8400±0.0252
50%	**0.8500±0.0082**	0.8365±0.0370	0.8366±0.0345	0.8252±0.0123	0.8369±0.0284	0.8302±0.0157	0.8411±0.0255	0.8363±0.0148	0.8496±0.0271
60%	0.8419±0.0206	0.8329±0.0358	0.8312±0.0369	0.8234±0.0228	0.8354±0.0279	0.8323±0.0228	0.8355±0.0161	0.8382±0.0175	**0.8428±0.0227**

TABLE IV: Performance results for the *Diabetic* dataset

Missing rates	Missing feature - 2								
	proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	**0.7346±0.0061**	0.7325±0.0148	0.7330±0.0168	0.7119±0.0109	0.7188±0.0138	0.7133±0.0207	0.7194±0.0192	0.7235±0.0208	0.7254±0.0245
20%	**0.7298±0.0200**	0.7250±0.0314	0.7278±0.0287	0.7091±0.0234	0.7100±0.0226	0.7158±0.0174	0.7246±0.0228	0.7225±0.0145	0.7289±0.0148
30%	0.7268±0.0204	0.7194±0.0316	**0.7273±0.0335**	0.7074±0.0188	0.7048±0.0272	0.7093±0.0076	0.7159±0.0094	0.7126±0.0173	0.7179±0.0184
40%	**0.7241±0.0191**	0.7206±0.0293	0.7221±0.0199	0.7033±0.0017	0.7039±0.0304	0.6950±0.0152	0.7052±0.0288	0.7125±0.0252	0.7130±0.0121
50%	**0.7182±0.0229**	0.7102±0.0432	0.7165±0.0270	0.7000±0.0179	0.7025±0.0281	0.7008±0.0155	0.7110±0.0188	0.7062±0.0178	0.7142±0.0182
60%	**0.7170±0.0282**	0.7029±0.0302	0.7035±0.0298	0.6917±0.0093	0.6936±0.0225	0.7009±0.0261	0.7048±0.0236	0.7058±0.0188	0.7049±0.0229
	Missing features - 2, 3								
10%	**0.7338±0.0116**	0.7317±0.0189	0.7305±0.0219	0.7007±0.0106	0.7040±0.0116	0.7208±0.0150	0.7220±0.0211	0.7110±0.0161	0.7208±0.0255
20%	**0.7244±0.0207**	0.7237±0.0236	0.7216±0.0274	0.7033±0.0017	0.7083±0.0081	0.7119±0.0261	0.7127±0.0153	0.7106±0.0093	0.7142±0.0342
30%	0.7257±0.0020	0.7236±0.0283	**0.7281±0.0117**	0.7023±0.0017	0.7090±0.0251	0.7108±0.0226	0.7032±0.0171	0.7100±0.0184	0.7055±0.0192
40%	**0.7263±0.0225**	0.7216±0.0307	0.7218±0.0367	0.6715±0.0188	0.6969±0.0181	0.6985±0.0145	0.7055±0.0232	0.7007±0.0207	0.7032±0.0182
50%	**0.7232±0.0245**	0.7185±0.0208	0.7220±0.0321	0.6888±0.0271	0.6899±0.0231	0.6878±0.0224	0.6982±0.0234	0.6991±0.0149	0.6910±0.0186
60%	**0.7159±0.0261**	0.6964±0.0301	0.6949±0.0306	0.6869±0.0204	0.6879±0.0244	0.6875±0.0174	0.6827±0.0187	0.6840±0.0206	0.6884±0.0228
	Missing features - 2, 3, 9								
10%	0.7290±0.0184	0.7302±0.0194	**0.7330±0.0210**	0.7110±0.0104	0.7124±0.0156	0.7129±0.0209	0.7243±0.0290	0.7301±0.0185	0.7306±0.0229
20%	**0.7254±0.0204**	0.7234±0.0269	0.7112±0.0282	0.7105±0.0167	0.7090±0.0160	0.7119±0.0104	0.7150±0.0194	0.7113±0.0120	0.7208±0.0155
30%	**0.7211±0.0225**	0.7120±0.0284	0.7190±0.0256	0.6982±0.0213	0.6990±0.0139	0.7012±0.0060	0.7052±0.0170	0.7107±0.0253	0.7116±0.0271
40%	**0.7206±0.0225**	0.7058±0.0239	0.7179±0.0261	0.6900±0.0186	0.6935±0.0121	0.6965±0.0100	0.6997±0.0179	0.7004±0.0159	0.7000±0.0211
50%	**0.7199±0.0225**	0.7150±0.0419	0.7156±0.0248	0.6843±0.0177	0.6870±0.0246	0.6917±0.0217	0.6879±0.0236	0.7018±0.0058	0.7068±0.0276
60%	**0.7038±0.0016**	0.6891±0.0349	0.6906±0.0425	0.6840±0.0250	0.6824±0.0278	0.6850±0.0192	0.6827±0.0192	0.6802±0.0290	0.6856±0.0234

TABLE V: Performance results for the *Pima* dataset

Missing rates	Missing feature - 2								
	proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	**0.7706±0.0087**	0.7628±0.0202	0.7635±0.0236	0.7489±0.0115	0.7515±0.0178	0.7549±0.0043	0.7572±0.0202	0.7570±0.0066	0.7580±0.0253
20%	**0.7576±0.0217**	0.7557±0.0269	0.7351±0.0271	0.7388±0.0109	0.7342±0.0328	0.7401±0.0132	0.7411±.0233	0.7446±0.0130	0.7472±0.0298
30%	**0.7588±0.0263**	0.7517±0.0304	0.7401±0.0349	0.7330±0.0238	0.7203±0.0248	0.7334±0.0174	0.7424±0.0218	0.7172±0.0100	0.7307±0.0230
40%	**0.7505±0.0254**	0.7261±0.0424	0.7355±0.0203	0.7304±0.0180	0.7325±0.0236	0.7204±0.0175	0.7212±0.0287	0.7274±0.0107	0.7238±0.0324
50%	**0.7330±0.0136**	0.7271±0.0388	0.7267±0.0187	0.7215±0.0222	0.7134±0.0266	0.7158±0.0254	0.7104±0.0333	0.7206±0.0075	0.7225±0.0253
60%	**0.7317±0.0152**	0.7201±0.0404	0.7183±0.0504	0.7108±0.0250	0.7134±0.0171	0.7085±0.0025	0.7113±0.0302	0.7117±0.0109	07229±0.0221
	Missing features - 2, 6								
10%	**0.7763±0.0090**	0.7587±0.0201	0.7601±0.0246	0.7532±0.0197	0.7554±0.0172	0.7504±0.0205	0.7528±0.0307	0.7556±0.0212	0.7563±0.0211
20%	**0.7568±0.0225**	0.7508±0.0333	0.7427±0.0306	0.7345±0.0246	0.7433±0.0247	0.7317±0.0288	0.7346±0.0250	0.7409±0.0109	0.7416±0.0181
30%	0.7358±0.0139	**0.7512±0.0249**	0.7259±0.0249	0.7217±0.0215	0.7212±0.0259	0.7140±0.0156	0.7169±0.0146	0.7174±0.0152	0.7234±0.0228
40%	**0.7388±0.0132**	0.7254±0.0415	0.7239±0.0308	0.7201±0.0214	0.7199±0.0190	0.7039±0.0152	0.7056±0.0221	0.7003±0.0075	0.7013±0.0186
50%	**0.7272±0.0207**	0.7191±0.0567	0.7224±0.0526	0.7131±0.0238	0.7160±0.0216	0.6869±0.0200	0.6870±0.0198	0.6987±0.0163	0.6900±0.0391
60%	**0.7172±0.0100**	0.7166±0.0373	0.7169±0.0283	0.7100±0.0170	0.7122±0.0250	0.6797±0.0212	0.6732±0.0258	0.6849±0.0257	0.6857±0.0362
	Missing features -1, 2, 6								
10%	**0.7619±0.0189**	0.7562±0.0349	0.7553±0.0249	0.7448±0.0195	0.7459±0.0277	0.7480±0.0109	0.7403±0.0203	0.7509±0.0198	0.7515±0.0216
20%	**0.7518±0.0218**	0.7492±0.0254	0.7462±0.0269	0.7330±0.0200	0.7310±0.0332	0.7305±0.0307	0.7307±0.0277	0.7317±0.0090	0.7372±0.0194
30%	0.7359±0.0229	**0.7436±0.0293**	0.7243±0.0276	0.7165±0.0109	0.7188±0.0264	0.7042±0.0205	0.7147±0.0209	0.7159±0.0154	0.7117±0.0366
40%	**0.7201±0.0218**	0.7114±0.0326	0.7169±0.0463	0.6948±0.0109	0.7088±0.0208	0.7043±0.0198	0.7027±0.0322	0.7002±0.0149	0.6905±0.0184
50%	0.7148±0.0222	**0.7171±0.0469**	0.7141±0.0422	0.6984±0.0238	0.7030±0.0247	0.6812±0.0152	0.6827±0.0280	0.6827±0.0189	0.6861±0.0260
60%	**0.7128±0.0107**	0.7106±0.0359	0.7104±0.0258	0.7082±0.0259	0.7065±0.0343	0.6782±0.0090	0.6727±0.0272	0.6714±0.0090	0.6814±0.0243

TABLE VI: Performance results for the *Bupa* dataset

Missing rates	Missing feature - 5								
	proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	**0.7212±0.0333**	0.6876±0.0464	0.6903±0.0220	0.6810±0.0314	0.6888±0.0449	0.7040±0.0468	0.6837±0.0426	0.6865±0.0329	0.6865±0.0529
20%	**0.7147±0.0350**	0.6948±0.0403	0.6964±0.0275	0.6790±0.0114	0.6700±0.0351	0.6865±0.0389	0.6654±0.0506	0.6869±0.0427	0.6721±0.0312
30%	**0.6955±0.0347**	0.6923±0.0411	0.6932±0.0224	0.6771±0.0282	0.6772±0.0225	0.6881±0.0399	0.6779±0.0452	0.6831±0.0394	0.6856±0.0385
40%	**0.6945±0.0317**	0.6831±0.0383	0.6857±0.0305	0.6750±0.0361	0.6763±0.0284	0.6775±0.0369	0.6625±0.0426	0.6713±0.0434	0.6712±0.0518
50%	**0.6763±0.0242**	0.6581±0.0390	0.6654±0.0383	0.6575±0.0378	0.6538±0.0319	0.6665±0.0369	0.6683±0.0590	0.6735±0.0422	0.6692±0.0517
60%	0.6744±0.0296	0.6590±0.0476	0.6667±0.0376	0.6644±0.0402	0.6692±0.0344	0.6721±0.0403	**0.6788±0.0294**	0.6705±0.0167	0.6865±0.0382
	Missing features - 3, 5								
10%	**0.7051±0.0242**	0.6846±0.0430	0.6860±0.0360	0.6837±0.0426	0.6885±0.0260	0.6840±0.0374	0.6837±0.0309	0.6935±0.0356	0.6894±0.0531
20%	**0.7019±0.0192**	0.6928±0.0426	0.6912±0.0238	0.6856±0.0428	0.6692±0.0251	0.6598±0.0350	0.6596±0.0418	0.6810±0.0428	0.6673±0.0495
30%	**0.7008±0.0304**	0.6712±0.0467	0.6732±0.0209	0.6775±0.0411	0.6785±0.0161	0.6608±0.0386	0.6692±0.0467	0.6810±0.0421	0.6808±0.0477
40%	**0.6795±0.0294**	0.6720±0.0447	0.6762±0.0181	0.6588±0.0357	0.6558±0.0275	0.6308±0.0432	0.6471±0.0408	0.6515±0.0343	0.6269±0.0486
50%	**0.6787±0.0211**	0.6474±0.0111	0.6485±0.0359	0.6683±0.0357	0.6692±0.0439	0.6383±0.0399	0.6260±0.0383	0.6179±0.0409	0.6087±0.0484
60%	**0.6674±0.0344**	0.6429±0.0391	0.6419±0.0281	0.6606±0.0444	0.6619±0.0251	0.6337±0.0446	0.6375±0.0453	0.6275±0.0473	0.6154±0.0260
	Missing features -3, 5, 6								
10%	**0.6963±0.0284**	0.6874±0.0393	0.6886±0.0258	0.6895±0.0309	0.6923±0.0251	0.6917±0.0353	0.6856±0.0253	0.6927±0.0438	0.6837±0.0404
20%	**0.6951±0.0156**	0.6918±0.0170	0.6929±0.0252	0.6867±0.0462	0.6831±0.0331	0.6621±0.0323	0.6712±0.0384	0.6740±0.0311	0.6750±0.0506
30%	0.6763±0.0200	**0.6859±0.0419**	0.6713±0.0288	0.6740±0.0416	0.6731±0.0240	0.6565±0.0358	0.6653±0.0456	0.6742±0.0361	0.6654±0.0455
40%	**0.6699±0.0334**	0.6502±0.0330	0.6635±0.0379	0.6468±0.0452	0.6454±0.0225	0.6285±0.0358	0.6308±0.0398	0.6338±0.0358	0.6404±0.0228
50%	**0.6663±0.0338**	0.6478±0.0294	0.6400±0.0229	0.6611±0.0415	0.6446±0.0228	0.6163±0.0412	0.6173±0.0381	0.6169±0.0387	0.6154±0.0377
60%	**0.6699±0.0242**	0.6429±0.0238	0.6433±0.0219	0.6450±0.0332	0.6492±0.0233	0.6090±0.0056	0.6115±0.0645	0.5942±0.0367	0.6087±0.0301

TABLE VII: Performance results for the *Breast* dataset

Missing rates	Missing feature - 2								
	proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	**0.9756±0.0109**	0.9741±0.0117	0.9744±0.0104	0.9711±0.0077	0.9678±0.0056	0.9702±0.0103	0.9712±0.0087	0.9699±0.0101	0.9702±0.0084
20%	**0.9707±0.0049**	0.9678±0.0114	0.9637±0.0111	0.9699±0.0085	0.9620±0.0011	0.9696±0.0117	0.9634±0.0098	0.9674±0.0113	0.9654±0.0123
30%	**0.9703±0.0098**	0.9658±0.0106	0.9611±0.0167	0.9687±0.0104	0.9698±0.0094	0.9688±0.0102	0.9639±0.0126	0.9698±0.0101	0.9620±0.0085
40%	0.9707±0.0049	0.9740±0.0092	**0.9755±0.0093**	0.9701±0.0090	0.9727±0.0117	0.9706±0.0115	0.9605±00049	0.9700±0.0104	0.9707±0.0098
50%	**0.9711±0.0075**	0.9705±0.0139	0.9689±0.0081	0.9704±0.0114	0.9707±0.0114	0.9690±0.0094	0.9683±0.0098	0.9694±0.0118	0.9668±0.0136
60%	**0.9690±0.0076**	0.9563±0.0122	0.9634±0.0149	0.9673±0.0100	0.9668±0.0111	0.9683±0.0111	0.9678±0.0132	0.9687±0.0093	0.9680±0.0306
	Missing features - 2, 6								
10%	**0.9750±0.0056**	0.9732±0.0117	0.9719±0.0140	0.9687±0.0097	0.9680±0.0164	0.9686±0.0106	0.9688±0.0158	0.9686±0.0089	0.9659±0.0126
20%	**0.9724±0.0065**	0.9674±0.0104	0.9610±0.0111	0.9683±0.0093	0.9649±0.0106	0.9668±0.0108	0.9693±0.0069	0.9654±0.0085	0.9665pm0.0067
30%	**0.9701±0.0049**	0.9643±0.0117	0.9608±0.0103	0.9683±0.0100	0.9688±0.0089	0.9684±0.0092	0.9663±0.0101	0.9686±0.0088	0.9605±0.0074
40%	0.9691±0.0028	0.9650±0.0111	**0.9706±0.0136**	0.9671±0.0111	0.9629±0.0112	0.9650±0.0108	0.9644±0.0138	0.9644±0.0104	0.9571±0.0136
50%	**0.9707±0.0123**	0.9659±0.0106	0.9689±0.0221	0.9691±0.0100	0.9659±0.0069	0.9683±0.0106	0.9673±0.0110	0.9676±0.0104	0.9595±0.0103
60%	**0.9688±0.0146**	0.9556±0.0195	0.9585±0.0118	0.9681±0.0087	0.9639±0.0117	0.9682±0.0108	0.9639±0.0118	0.9673±0.0108	0.9683±0.0120
	Missing features -2, 6, 1								
10%	**0.9749±0.0098**	0.9728±0.0117	0.9708±0.0082	0.9676±0.0109	0.9688±0.0112	0.9694±0.0106	0.9663±0.0120	0.9647±0.0102	0.9693±0.0130
20%	**0.9695±0.0123**	0.9659±0.0126	0.9625±0.0033	0.9692±0.0097	0.9668±0.0080	0.9648±0.0096	0.9595±0.0103	0.9645±0.0129	0.9615±0.0099
30%	**0.9707±0.0176**	0.9653±0.0110	0.9610±0.0079	0.9681±0.0096	0.9608±0.0168	0.9674±0.0104	0.9654±0.0137	0.9650±0.0096	0.9639±0.0113
40%	0.9685±0.0102	0.9675±0.0129	**0.699±0.0068**	0.9675±0.0114	0.9629±0.0101	0.9625±0.0122	0.9649±0.0110	0.9637±0.0084	0.9634±0.0111
50%	**0.9691±0.0075**	0.9648±0.0139	0.9689±0.0081	0.9670±0.0088	0.9610±0.0150	0.9633±0.0107	0.9527±0.0100	0.9611±0.0118	0.9624±0.0153
60%	**0.9659±0.0049**	0.9366±0.0208	0.9585±0.0139	0.9652±0.0097	0.9651±0.0199	0.9639±0.0086	0.9615±0.0081	0.9615±0.0096	0.9610±0.0142

TABLE VIII: Performance results for the *Titanic* dataset

Missing rates	Missing feature - 2								
	proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	**0.8115±0.0232**	0.8067±0.0224	0.8025±0.0339	0.8052±0.0112	0.7925±0.0187	0.7953±0.0206	0.8007±0.0287	0.7905±0.0216	0.7869±0.0217
20%	**0.7878±0.0204**	0.7809±0.0071	0.7822±0.0188	0.7728±0.0238	0.7760±0.0236	0.7578±0.0132	0.7610±0.0229	0.7609±0.0151	0.7674±0.0191
30%	**0.7851±0.0142**	0.7594±0.0108	0.7754±0.0151	0.7708±0.0226	0.7715±0.0221	0.7503±0.0238	0.7536±0.0326	0.7415±0.0195	0.7427±0.0277
40%	**0.7828±0.0198**	0.7533±0.0072	0.7666±0.0215	0.7640±0.0262	0.7561±0.0239	0.7466±0.0108	0.7534±0.0198	0.7428±0.0142	0.7482±0.0257
50%	**0.7740±0.0192**	0.7469±0.0189	0.7654±0.0505	0.7409±0.0078	0.7416±0.0384	0.7219±0.0213	0.7221±0.0208	0.7303±0.0132	0.7307±0.0226
60%	**0.7461±0.0244**	0.7432±0.0270	0.7402±0.0357	0.7419±0.0205	0.7446±0.0144	0.7141±0.0150	0.7161±0.0225	0.7116±0.0120	0.7139±0.0201
	Missing features - 2, 6								
10%	**0.7940±0.0112**	0.7853±0.0192	0.7867±0.0254	0.7815±0.0078	0.7790±0.0211	0.7809±0.0163	0.7820±0.0198	0.7802±0.0206	0.7818±0.0153
20%	**0.7881±0.0276**	0.7800±0.0194	0.7850±0.0160	0.7628±0.0284	0.7753±0.0191	0.7540±0.0185	0.7638±0.0188	0.7627±0.0233	0.7667±0.0185
30%	**0.7740±0.0213**	0.7533±0.0218	0.7642±0.0246	0.7615±0.0244	0.7703±0.0174	0.7590±0.0057	0.7596±0.0217	0.7403±0.0099	0.7488±0.0308
40%	**0.7747±0.0206**	0.7530±0.0122	0.7669±0.0229	0.7618±0.0120	0.7326±0.0248	0.7458±0.0214	0.7506±0.0249	0.7415±0.0173	0.7498±0.0245
50%	**0.7703±0.0213**	0.7513±0.0205	0.7636±0.0303	0.7465±0.0228	0.7408±0.0192	0.7216±0.0255	0.7494±0.0239	0.7301±0.0132	0.7364±0.0183
60%	**0.7701±0.0156**	0.7450±0.0207	0.7606±0.0299	0.7405±0.0150	0.7423±0.0302	0.7128±0.0135	0.7464±0.0262	0.7139±0.0216	0.7479±0.0150
	Missing features -2, 6, 1								
10%	**0.7842±0.0228**	0.7817±0.0199	0.7826±0.0205	0.7665±0.0213	0.7693±0.0252	0.7790±0.0209	0.7798±0.0266	0.7753±0.0281	0.7809±0.0209
20%	**0.7828±0.0169**	0.7802±0.0187	0.7808±0.0270	0.7618±0.0132	0.7588±0.0279	0.7527±0.0086	0.7629±0.0189	0.7440±0.0099	0.7464±0.0266
30%	**0.7640±0.0163**	0.7528±0.0196	0.7605±0.0121	0.7528±0.0172	0.7538±0.0154	0.7466±0.0249	0.7610±0.0247	0.7409±0.0065	0.7385±0.0186
40%	**0.7541±0.0189**	0.7483±0.0232	0.7469±0.0266	0.7505±0.0120	0.7348±0.0114	0.7441±0.0173	0.7367±0.0209	0.7378±0.0182	0.7315±0.0213
50%	**0.7516±0.0212**	0.7442±0.0130	0.7450±0.0202	0.7319±0.0250	0.7363±0.0115	0.7206±0.0264	0.7330±0.0175	0.7214±0.0142	0.7206±0.0236
60%	**0.7592±0.0057**	0.7449±0.0196	0.7504±0.0302	0.7079±0.0192	0.6944±0.0210	0.7116±0.0163	0.6918±0.0285	0.7089±0.0234	0.6940±0.0292

TABLE IX: Performance results for the *German* dataset

Missing rates	Missing feature(s) - 1								
	Proposed method	case deletion		mean imputation		KNN3		KNN10	
		LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
10%	0.7667±0.0165	0.7556±0.0120	0.7600±0.0109	**0.7757±0.0171**	0.7670±0.0105	0.7567±0.0202	0.7547±0.0219	0.7573±0.0225	0.7640±0.0132
20%	**0.7644±0.0255**	0.7533±0.0231	0.7558±0.0172	0.7583±0.0300	0.7587±0.0166	0.7530±0.0229	0.7520±0.0204	0.7607±0.0168	0.7633±0.0233
30%	**0.7589±0.0158**	0.7393±0.0227	0.7361±0.0212	0.7473±0.0173	0.7480±0.0206	0.7400±0.0091	0.7420±0.0150	0.7420±0.0344	0.7427±0.0055
40%	**0.7533±0.0209**	0.7389±0.0226	0.7408±0.0120	0.7407±0.0159	0.7427±0.0210	0.7429±0.0209	0.7423±0.0077	0.7423±0.0088	0.7413±0.0250
50$	**0.7478±0.0287**	0.7367±0.0223	0.7385±0.0203	0.7296±0.0139	0.7350±0.0215	0.7411±0.0336	0.7393±0.0174	0.7267±0.0338	0.7307±0.0218
60%	**0.7322±0.0212**	0.7127±0.0201	0.7188±0.0212	0.7083±0.0171	0.7107±0.0112	0.7144±0.0393	0.7213±0.0186	0.7189±0.0550	0.7183±0.0256
	Missing features - 1, 2								
10%	0.7644±0.0115	0.7600±0.0190	0.7630±0.0128	**0.7670±0.0170**	0.7630±0.0175	0.7543±0.0221	0.7553±0.0279	0.7650±0.0203	0.7667±0.0122
20%	**0.7598±0.0126**	0.7550±0.0212	0.7583±0.0056	0.7530±0.0228	0.7533±0.0171	0.7460±0.0253	0.7477±0.0215	0.7530±0.0203	0.7573±0.0028
30%	**0.7588±0.0242**	0.7430±0.0150	0.7469±0.0122	0.7477±0.0236	0.7487±0.0201	0.7407±0.0251	0.7467±0.0213	0.7460±0.0225	0.7420±0.0084
40%	**0.7466±0.0084**	0.7322±0.0160	0.7409±0.0203	0.7313±0.0234	0.7320±0.0222	0.7344±0.0139	0.7347±0.0238	0.7448±0.0019	0.7390±0.0208
50%	**0.7344±0.0204**	0.7260±0.0335	0.7333±0.0206	0.7264±0.0184	0.7307±0.0106	0.7342±0.0096	0.7293±0.0202	0.7222±0.0168	0.7280±0.0281
60%	**0.7356±0.0184**	0.7067±0.0162	0.7100±0.0156	0.7022±0.0184	0.7140±0.0140	0.7133±0.0150	0.7137±0.0214	0.7149±0.0151	0.7138±0.187
	Missing features - 1, 2, 3								
10%	**0.7600±0.0209**	0.7497±0.0156	0.7570±0.0184	0.7587±0.0142	0.7580±0.0208	0.7497±0.0175	0.7560±0.0130	0.7557±0.0246	0.7547±0.0090
20%	**0.7589±0.0038**	0.7525±0.0197	0.7558±0.0088	0.7500±0.0174	0.7547±0.0207	0.7313±0.0159	0.7367±0.0103	0.7487±0.0224	0.7447±0.0107
30%	**0.7567±0.0233**	0.7366±0.0210	0.7389±0.0217	0.7443±0.0166	0.7410±0.0201	0.7353±0.0168	0.7253±0.0266	0.7428±0.0078	0.7430±0.0267
40%	**0.7411±0.0215**	0.7333±0.0120	0.7389±0.0116	0.7387±0.0164	0.7370±0.0222	0.7347±0.0058	0.7380±0.0201	0.7311±0.0334	0.7380±0.0180
50%	**0.7322±0.0139**	0.7207±0.0115	0.7293±0.0218	0.7267±0.0167	0.7273±0.0203	0.7267±0.0145	0.7240±0.0192	0.7219±0.0051	0.7260±0.0195
60%	**0.7244±0.0254**	0.7105±0.0205	0.7167±0.0129	0.7037±0.0122	0.7073±0.0205	0.7112±0.0069	0.7160±0.0158	0.7156±0.0051	0.7133±0.0211

TABLE X: Performance results for the *MIHC* dataset

Dataset	Missing features rate	Missing rate	Proposed method	Case deletion		Mean imputation		KNN3		KNN10	
				LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM	LS-SVM	SVM
MIHC	39.39%	35.81%	**0.7258±0.0289**	0.6954±0.0479	0.7023±0.031	0.7147±0.0330	0.7164±0.0271	0.7110±0.0229	0.7187±0.0211	0.7190±0.0305	0.7229±0.023

TABLE XI: Performance results after data cleaning for the *MIHC* dataset

Threshold	0.60	0.65	0.80	1.00
Performance	0.7265±0.0212	0.7288±0.0210	0.7327±0.0098	0.7300±0.0331

TABLE XII: Average rankings of the proposed and comparative methods on seven public datasets in terms of average accuracy (p-value=0.000704)

Methods	Ranking
Proposed method	1
case deletion + SVM	3
mean imputation + SVM	5
case deletion + LS-SVM	5
mean imputation + LS-SVM	5.4286
KNN10 + SVM	5.7143
KNN10 + LS-SVM	6.1429
KNN3 + SVM	6.7143
KNN3 + LS-SVM	7

TABLE XIII: Holm Post-Hoc comparison results for the proposed and comparative methods in terms of average accuracy with $\alpha = 0.05$

i	Methods	z-value	p-value	Holm=α/i
8	KNN3 + LS-SVM	4.09878	0.000042	0.00625
7	KNN3 + SVM	3.9036	0.000095	0.007143
6	KNN10 + LS-SVM	3.51324	0.000443	0.008333
5	KNN10 + SVM	3.22047	0.00128	0.01
4	mean imputation + LS-SVM	3.02529	0.002484	0.0125
3	case deletion + LS-SVM	2.73252	0.006285	0.016667
2	mean imputation + SVM	2.73252	0.006285	0.025
1	case deletion + SVM	1.36626	0.171857	0.05

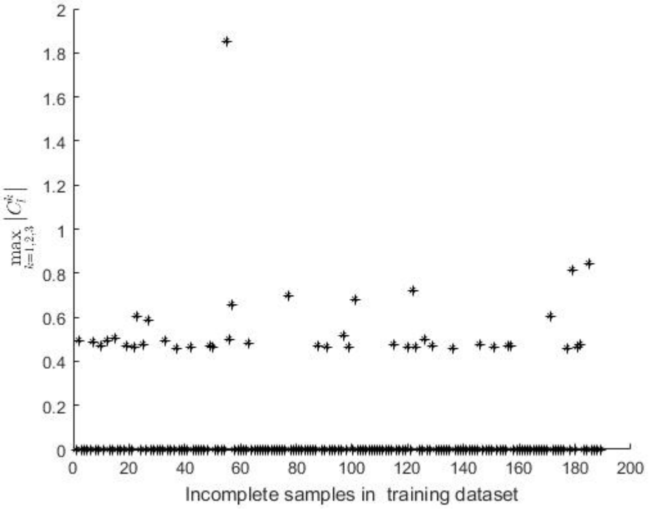


Fig. 4: Comparative results after data cleaning on the *MIHC* dataset

References

[1] S. Thirukumaran and A. Sumathi, "Improving accuracy rate of imputation of missing data using classifier methods," in 2016 10th International Conference on Intelligent Systems and Control (ISCO). IEEE, 2016, pp. 1–7.

[2] T. Razzaghi, O. Roderick, I. Safro, and N. Marko, "Fast imbalanced classification of healthcare data with missing values," in 2015 18th International Conference on Information Fusion (Fusion). IEEE, 2015, pp. 774–781.

[3] L. Lorenzi, G. Mercier, and F. Melgani, "Support vector regression with kernel combination for missing data reconstruction," IEEE Geoscience and Remote Sensing Letters, vol. 10, no. 2, pp. 367–371, 2013.

[4] Y. Zhang and Y. Liu, "Data imputation using least squares support vector machines in urban arterial streets," IEEE Signal Processing Letters, vol. 16, no. 5, pp. 414–417, 2009.

[5] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, Least Squares Support Vector Machines. Singapore: World Scientific, 2002.

[6] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," Neural Computing and Applications, vol. 19, no. 2, pp. 263–282, 2010.

[7] R. J. Little and D. B. Rubin, Statistical analysis with missing data. John Wiley & Sons, 2014.

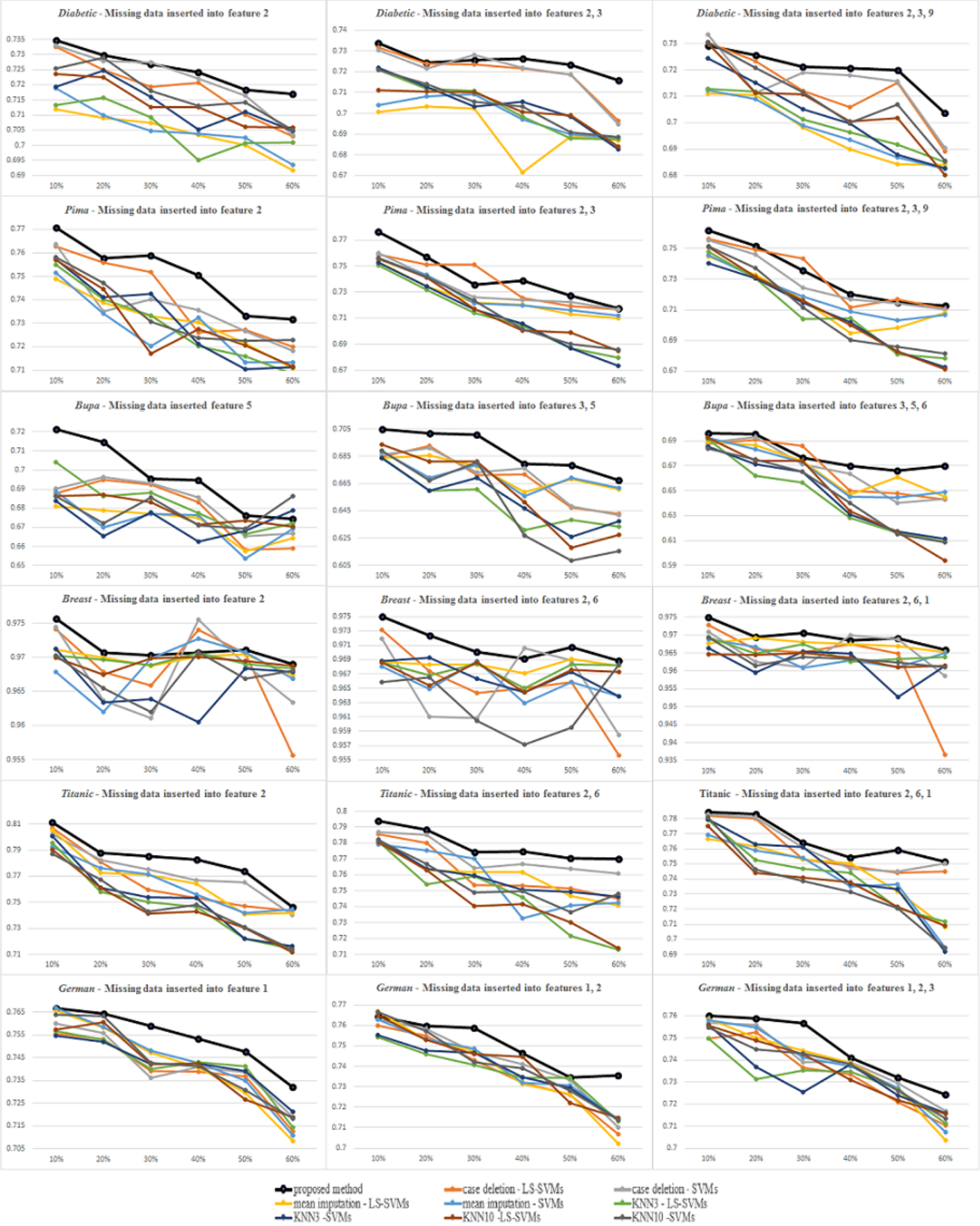


Fig. 5: Comparative results of proposed and comparative methods on seven public datasets

[8] G. E. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," Applied Artificial Intelligence, vol. 17, no. 5-6, pp. 519–533, 2003.

[9] A. R. T. Donders, G. J. van der Heijden, T. Stijnen, and K. G. Moons, "Review: a gentle introduction to imputation of missing values," Journal of Clinical Epidemiology, vol. 59, no. 10, pp. 1087–1091, 2006.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," Journal of the Royal Statistical Society. Series B (Methodological), pp. 1–38, 1977.

[11] N. J. Horton and K. P. Kleinman, "Much ado about nothing," The American Statistician, 2012.

[12] J. G. Ibrahim, M.-H. Chen, and S. R. Lipsitz, "Monte carlo em for missing covariates in parametric regression models," Biometrics, vol. 55, no. 2, pp. 591–596, 1999.

[13] K. Pelckmans, J. De Brabanter, J. A. Suykens, and B. De Moor, "Handling missing values in support vector machine classifiers," Neural Networks, vol. 18, no. 5, pp. 684–692, 2005.

[14] A. J. Smola, S. Vishwanathan, and T. Hofmann, "Kernel methods for missing variables," in Proceedings of 10th International Conference on Artifical Intelligence and Statistics (AISTATS), Barbados, 2005, p. 325.

[15] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, "Max-margin classification of incomplete data," in Advances in Neural Information Processing Systems, 2006, pp. 233–240.

[16] J. B. T. Zhang, "Support vector classification with input data uncertainty," Advances in Neural Information Processing Systems, vol. 17, pp. 161–169, 2005.

[17] P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola, "Second order cone programming approaches for handling missing and uncertain data,"

Journal of Machine Learning Research, vol. 7, no. Jul, pp. 1283–1314, 2006.

[18] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, 2010.

[19] J. Jiang and C. Zhai, “Instance weighting for domain adaptation in nlp,” in Proceedings of 45th Annual Meeting of the Association for Computational Linguistics (ACL), vol. 7, Prague, Czech Republic, June 2007, pp. 264–271.

[20] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, “Correcting sample selection bias by unlabeled data,” in Advances in Neural Information Processing Systems, 2006, pp. 601–608.

[21] T. Jebara, “Multi-task feature and kernel selection for svms,” in Proceedings of 21st International Conference on Machine Learning (ICML), Banff, Alberta, Canada, July 2004, p. 55.

[22] A. Argyriou, M. Pontil, Y. Ying, and C. A. Micchelli, “A spectral regularization framework for multi-task structure learning,” in Advances in Neural Information Processing Systems, 2007, pp. 25–32.

[23] H. Daumé III, “Frustratingly easy domain adaptation,” in Proceedings of 45th Annual Meeting of the Association for Computational Linguistics (ACL), June 2007, pp. 256–263.

[24] M. Jiang, W. Huang, Z. Huang, and G. G. Yen, “Integration of global and local metrics for domain adaptation learning via dimensionality reduction,” IEEE Transactions on Cybernetics, vol. 47, no. 1, pp. 38–51, 2017.

[25] M. Uzair and A. Mian, “Blind domain adaptation with augmented extreme learning machine features,” IEEE Transactions on Cybernetics, vol. 47, no. 3, pp. 651–660, 2017.

[26] K. Yan, L. Kou, and D. Zhang, “Learning domain-invariant subspace using domain features and independence maximization,” IEEE transactions on cybernetics, 2017.

[27] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, “Transfer learning using computational intelligence: a survey,” Knowledge-Based Systems, vol. 80, pp. 14–23, 2015.

[28] W. Liu, H. Zhang, and J. Li, “Inductive transfer through neural network error and dataset regrouping,” in IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), vol. 1, Shanghai, China, 2009, pp. 777–781.

[29] R. Luis, L. E. Sucar, and E. F. Morales, “Inductive transfer for learning bayesian networks,” Machine Learning, vol. 79, no. 1-2, pp. 227–255, 2010.

[30] H. Zuo, G. Zhang, V. Behbood, J. Lu, and X. Meng, “Transfer learning in hierarchical feature spaces,” in International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2015, pp. 183–188.

[31] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, “Granular fuzzy regression domain adaptation in takagi-sugeno fuzzy models,” IEEE Transactions on Fuzzy Systems, 2017.

[32] V. Behbood, J. Lu, and G. Zhang, “Fuzzy refinement domain adaptation for long term prediction in banking ecosystem,” IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1637–1646, 2014.

[33] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, “To transfer or not to transfer,” in Conference on Neural Information Processing Systems 2005 Workshop On Transfer Learning, vol. 898, 2005.

[34] C.-W. Seah, Y.-S. Ong, and I. W. Tsang, “Combating negative transfer from predictive distribution differences,” IEEE Transactions on Cybernetics, vol. 43, no. 4, pp. 1153–1165, 2013.

[35] D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen, “Additive gaussian processes,” in Advances in Neural Information Processing Systems, 2011, pp. 226–234.

[36] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” Journal of Machine Learning Research, vol. 7, no. Jan, pp. 1–30, 2006.

[37] S. Garcia and F. Herrera, “An extension on ’statistical comparisons of classifiers over multiple data sets’ for all pairwise comparisons,” Journal of Machine Learning Research, vol. 9, no. Dec, pp. 2677–2694, 2008.

[38] K.-S. Choi, R. K. Wai, and E. Y. Kwok, “Healthcare information system: a facilitator of primary care for underprivileged elderly via mobile clinic,” in International Conference on Smart Health. Springer, 2013, pp. 107–112.

[39] K. Leung, W. Wong, M. Tay, M. Chu, and S. Ng, “Development and validation of the interview version of the hong kong chinese whoqol-bref,” Quality of Life Research, vol. 14, no. 5, pp. 1413–1419, 2005.

[40] K. Leung, M. Tay, S. Cheng, and F. Lin, “Hong kong chinese version world health organization quality of life measure-abbreviated version,” WHOQOL-BREF (HK), 1997.

Guanjin Wang received both Bachelor and Master degrees in Information Technology and Systems from Monash University, Australia in 2012 and 2014 respectively. She is currently completing a joint Ph.D with the Faculty of Engineering and Information Technology at the University of Technology Sydney and School of Nursing at the Hong Kong Polytechnic University. Her research interest includes machine learning, computational intelligence and health informatics.

Jie Lu is a Distinguished Professor and associate dean of research with the Faculty of Engineering and Information Technology at the University of Technology Sydney. Her main research expertise is in fuzzy transfer learning, decision support systems, concept drift, and recommender systems. She has published 10 research books and 400 papers, and have won over 20 Australian Research Council (ARC) discovery grants and other research grants over $4 million. She serves as Editor-In-Chief for KBS (Elsevier) and IJCIS (Atlantis), and has delivered 20 keynote speeches in international conferences. She is a Fellow of IEEE and Fellw of IFSA.

Kup-Sze Choi is an Associate Professor with the School of Nursing in The Hong Kong Polytechnic University and Director of Centre for Smart Health. He received the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong. He has over 100 publications and awarded 4 General Research Fund (GRF) projects. His research focuses on healthcare innovations by leveraging virtual reality, machine learning and artificial intelligence.

Guangquan Zhang is an Associate Professor with the Faculty of Engineering and Information Technology at the University of Technology Sydney. His research interests include fuzzy machine learning, fuzzy optimization, and machine learning and data analytics. He has authored four monographs, five textbooks, and 350 papers including 160 refereed international journal papers. He has been awarded seven Australian Research Council (ARC) Discovery Project grants and many other research grants. He has served as a member of the editorial boards of several international journals, as a guest editor of eight special issues for IEEE transactions and other international journals, and co-chaired several international conferences and workshops in the area of fuzzy decision-making and knowledge engineering.