2. REVIEW OF LITERATURE

Most of the data collection techniques, like survey studies, field experiments, scientific research findings, etc., produce huge amount of information. Missing values are inevitable in such data collected. Moreover, data mining techniques, like clustering and classification, have been designed to analyze and discover knowledge from data that is complete, that is, it does not contain missing values. The presence of missing values degrades the performance of these data analysis techniques. Methods to improve the data mining process, hence, have been an area that has attracted many researchers. Generally, the analyst has two options to create a dataset containing no missing values.

(i) to delete or ignore these faulty records with missing values
(ii) fill the missing value with estimated values

Deleting or ignoring rows with missing values have been proved to be inefficient and therefore methods that predict the missing values have gained more attention. Some of these methods that are related to the present research work are discussed in this chapter.

As the present research work focus on classification of data with missing data, the first section starts with a study on the works related to general classification, followed by various studies that have focused on missing values.

2.1 GENERAL CLASSIFICATION SYSTEMS

Classification is an important data mining problem. Many classification models have been proposed in the literature. While classification is a well-studied problem, only recently there has been focus on algorithms that can handle large databases. The intuition is that by classifying larger datasets, the accuracy of the classification model can be improved ([33], [35], [36]). Most algorithms in the machine learning and
statistics community are main memory algorithms, even though today’s databases are in general much larger than main memory. Several classification models have been proposed over the years, examples include, neural networks [104], statistical models like linear/quadratic discriminants [79], decision trees [22] and genetic models [67]. This section presents studies that are based on these classifiers.

2.1.1. Decision trees

Decision trees classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified and each branch represents a value that the node can assume. Instances are classified starting at the root node and are sorted based on their feature values. The feature that best divides the training data would be the root node of the tree.

There are numerous methods for finding the feature that best divides the training data but a majority of studies have concluded that there is no single best method [119]. The authors presented a detailed study on the working of decision tree classification and discussed its usefulness in the field of machine learning. Breslow and Aha [23] presents a survey of the various methods based on decision trees that improve classification performance.

Because of the huge computations and memory requirements decision trees normally perform pruning operation before classification. The pre-pruning method apart from reducing size can also be used to tackle the problem of overfitting. A comparative study of well-known pruning methods is presented by [54].

The most well-known algorithm in the literature for building decision trees is the C4.5 [138]. One of the latest studies that compare decision trees and other learning
algorithms has been done by [162]. The study shows that C4.5 has a very good combination of error rate and speed. C4.5 assumes that the training data fits in memory, thus, [64] proposed Rainforest, a framework for developing fast and scalable algorithms to construct decision trees that gracefully adapt to the amount of main memory available. Baik and Bala [10] presented preliminary work on an agent-based approach for the distributed learning of decision trees.

2.1.2. Neural networks

The usage of neural networks to classify data into several classes is another area that has gained widespread attention. An overview of techniques using Artificial Neural Networks is presented by [190] and [194]. The applicability of neural networks for classification is studied by [30]. Kon and Plaskota [92] also studied the minimum amount of neurons and the number of instances necessary to program a given task into feed-forward neural networks.

The most well-known and widely used learning algorithm to estimate the values of the weights is the Back Propagation (BP) algorithm. Feed-forward neural network is another classifier that is usually trained by the original back propagation algorithm or by some variant. This has the drawback of being slow but is efficient in terms of classification accuracy. One of the approaches to speed up the training rate is to estimate optimal initial weights [185]. Genetic algorithms have also been used to train the weights of neural networks [155] and to find the architecture of neural networks [186].

There are also Bayesian methods in existence which attempt to train neural networks. Vivarelli and Williams [174] compare two Bayesian methods for training neural networks. A number of other techniques have emerged recently which attempt to
improve ANNs training algorithms by changing the architecture of the networks as training proceeds. These techniques include pruning useless nodes or weights [32], and constructive algorithms, where extra nodes are added as required [128]. A Radial Basis Function (RBF) neural network has been also widely applied in many science and engineering fields [141].

2.1.3 Statistical learning algorithms

Statistical approaches are characterized by having an explicit underlying probability model, which provides a probability that an instance belongs in each class, rather than simply a classification, as with neural networks. Under this category of classification algorithms, one can find Bayesian networks and instance-based methods.

A comprehensive book on Bayesian networks is Jensen’s [80].

- **Bayesian networks**

A Bayesian Network (BN) is a graphical model for probability relationships among a set of variables (features). The task of learning a Bayesian network can be divided into two subtasks: initially, the learning of the structure of the network and then the determination of its parameters. Probabilistic parameters are encoded into a set of tables, one for each variable, in the form of local conditional distributions of a variable given its parents. Given the independences encoded into the network, the joint distribution can be reconstructed by simply multiplying these tables. Within the general framework of inducing Bayesian networks, there are two scenarios: known structure and unknown structure. In the first scenario, the structure of the network is given (e.g. by an expert) and assumed to be correct. Once the network structure is fixed, learning the parameters in the Conditional Probability Tables (CPT) is usually solved by estimating a locally exponential number of parameters from the data provided [80].
Each node in the network has an associated CPT that describes the conditional probability distribution of that node given the different values of its parents.

In spite of the remarkable power of Bayesian Networks, they have an inherent limitation, that is, the computation difficulty. Several researchers have shown experimentally that the selection of a single good hypothesis using greedy search often yields accurate predictions ([74], [43]).

A BN structure can be also found by learning the conditional independence relationships among the features of a dataset. Using a few statistical tests (such as the Chi-squared and mutual information test), one can find the conditional independence relationships among the features and use these relationships as constraints to construct a BN. These algorithms are called CI-based algorithms or constraint-based algorithms. Cowell [46] has shown that for any structure search procedure based on CI tests, an equivalent procedure based on maximizing a score can be specified. A comparison of scoring-based methods and CI-based methods is presented by [74].

Madden [108] compared the performance of a number of Bayesian Network Classifiers. His experiments demonstrated that very similar classification performance can be achieved by classifiers constructed using the different approaches described above.

A problem of BN classifiers is that they are not suitable for datasets with many features [42]. The reason for this is that trying to construct a very large network is simply not feasible in terms of time and space. A final problem is that before the induction, the numerical features need to be discretized in most cases.
• **Instance-based learning**

Instance-based learning algorithms are lazy-learning algorithms [117], as they delay the induction or generalization process until classification is performed. Lazy-learning algorithms require less computation time during the training phase than eager-learning algorithms (such as decision trees, neural and Bayes nets) but more computation time during the classification process. One of the most straightforward instance-based learning algorithms is the nearest neighbour algorithm. Aha [4] and De Mantaras and Armengol [48] presented a review of instance-based learning classifiers.

k-Nearest Neighbour (kNN) is based on the principle that the instances within a dataset will generally exist in close proximity to other instances that have similar properties. The kNN locates the k nearest instances to the query instance and determines its class by identifying the single most frequent class label. For more accurate results, several algorithms use weighting schemes that alter the distance measurements and voting influence of each instance.

A survey of weighting schemes is given by [180]. The same authors investigated the behavior of the kNN in the presence of noisy instances. The experiments showed that the performance of kNN was not sensitive to the exact choice of k when k was large. They found that for small values of k, the kNN algorithm was more robust than the single nearest neighbour algorithm (1NN) for the majority of large datasets tested. However, the performance of the kNN was inferior to that achieved by the 1NN on small datasets (<100 instances).

Okamoto and Yugami [126] represented the expected classification accuracy of kNN as a function of domain characteristics including the number of training instances, the number of relevant and irrelevant attributes, the probability of each attribute, the
noise rate for each type of noise, and k. They also explored the behavioral implications of the analyses by presenting the effects of domain characteristics on the expected accuracy of kNN and on the optimal value of k for artificial domains.

The time to classify the query instance is closely related to the number of stored instances and the number of features that are used to describe each instance. Thus, in order to reduce the number of stored instances, instance-filtering algorithms have been proposed. Brighton and Mellish [24] found that their ICF algorithm and RT3 algorithm [182] achieved the highest degree of instance set reduction as well as the retention of classification accuracy: they are close to achieving unintrusive storage reduction. One other choice in designing a training set reduction algorithm is to modify the instances using a new representation such as prototypes [145].

Breiman [20] reported that the stability of nearest neighbor classifiers distinguishes them from decision trees and some kinds of neural networks. A learning method is termed “unstable” if small changes in the training-test set split can result in large changes in the resulting classifier. The major disadvantage of instance-based classifiers is their large computational time for classification. A key issue in many applications is to determine which of the available input features should be used in modeling via feature selection [188], because it could improve the classification accuracy and scale down the required classification time. Furthermore, choosing a more suitable distance metric for the specific dataset can improve the accuracy of instance-based classifiers.

2.2. MISSING DATA HANDLING METHODS

Inspite of the various classification algorithms present, when presented with incomplete datasets, as mentioned previously, the accuracy of the classifier decreases.
In these situations, special methods have to be developed to fill the missing values efficiently so as to maintain the accuracy of the classification process. Some of these methods are discussed in this section.

2.2.1. Imputation-based studies

The issue of missing values (or missing data) has been studied extensively in the statistical and machine learning literature. Various imputation methods have also been tried, such as imputation by the most common value [44], clustering [38], and other learning models [21]. In C4.5 ([138], [139]), a different approach is used in which a test example with missing values is distributed into branches probabilistically. A comparison of various imputation methods has also been published [99].

Methods which do not use imputation have also been proposed. These methods have the advantage that they use a cost-sensitive learning method. Turney [170] presents an excellent survey on different types of costs in cost-sensitive learning, among which misclassification costs and test costs are singled out as most important. Much work has been done in recent years on non-uniform misclassification costs (alone), such as [49], [53] and [83]. Some previous work, such as [125] and [159], considers the test cost alone without incorporating misclassification cost, which is obviously an oversight. A few researchers ([34], [68], [169], [196]) consider both misclassification and test costs, but their methods are less computationally efficient as our approach is based on decision trees.

Ling et al. [103] propose a decision-tree learning algorithm that uses minimum total cost of tests and misclassifications as the attribute split criterion. Basically, given a set of training examples, the total cost without further splitting and the total cost after splitting on an attribute can be calculated and the difference of the two is called cost
reduction. The attribute with the maximum, positive cost reduction is chosen for growing the tree. All examples with missing values of an attribute stay at the internal node of that attribute. The method produces decision trees with minimal total cost of tests and misclassifications on the training data.

2.2.2. Multiple Imputation

The idea of multiple imputations, first proposed by [142], is to impute more than one value for the missing item. The advantage of multiple imputation is that it represents the uncertainty about which value to impute. This is as opposed to imputing the mean response which does not incorporate the degree of uncertainty about which value to impute. Therefore, the analyses that treats imputed values just like observed values generally underestimate the variability [143]. Multiple imputation can be implemented for either longitudinal measurements or a single response.

The general strategy for multiple imputation is to replace each missing value with two or more values from an appropriate distribution for the missing values. This produces two or more complete data sets. Repeated draws are made from the posterior predictive distribution of the missing values. As [143] point out, in practice, implicit models can be used in place of explicit models. Lavori et al. [100] discuss a propensity-based imputation where one models the probability of remaining in the study given a vector of observed covariates.

2.2.3. Association rule based Imputation

The term association rule was first introduced in the Apriori algorithm developed by Agrawal and group in the context of market basket analysis [3]. Another work of using association rule induction utilized Apriori method to find duplicate
relations in a representative biological dataset as introduced by [89]. Another research in the area of using association rules was done by [157] who considered the problem of discovering association rules in the presence of constraints that are Boolean expressions over the presence or absence of items.

Marcus et al. [110] introduced ordinal association rules to uncover relationships, numerical ordering or equality between attributes that commonly occur in the dataset which help in identifying attributes that do not conform to the discovered ordering. Later, [31] introduced relational association rules which are an extension of the ordinal association rules to be able to capture various kinds of relationships between record attributes. Nayak and Cook [121] described an association rule mining algorithm called approximate association rules (~AR), which is an enhancement of the Apriori algorithm. It allows data that approximately matches the pattern to contribute toward the overall support of the pattern which is useful in processing missing values.

The work by [75] compares between several algorithms dealing with association rules. Several researchers worked on the issue of resolving missing values in datasets ([140], [29], [183], [127]).

Calders et al. [29] introduced new definitions for the terms support and confidence based on the absence of missing values in database for the attributes of the itemsets. In addition, a new notation called representative was introduced to restrict the influence of itemsets that are not observed thoroughly by confidence and support. The XMiner algorithm was proposed using the new measures. An evaluation function and a completion procedure for finding missing values were presented by [183]. The proposed evaluation function is calculated according to the support, confidence, and the antecedent of the association rules.
2.3. IMPUTATION METHODS BASED ON MACHINE LEARNING

This section reviews the various techniques based on machine learning algorithms.

2.3.1. Imputation based on Artificial Neural Network Imputation

The imputation approaches based on Artificial Neural Networks (ANNs) generally consist of training an ANN model to estimate values using the complete instances, e.g., a Multi-Layer Perceptron (MLP) is used as a regression model in order to estimate missing values ([152], [124]). After a complete edited set is obtained, i.e., complete and imputed instances, a classifier system must be trained using this edited set to solve the classification task. ANNimpute can be an useful tool for reconstructing missing values. However, its main disadvantage is that when missing items appear in several attributes, several MLP models have to be designed, one per missing variables combination. Other MLP methods have been proposed. In 1999, Yoon and Lee [187] proposed the TEST (Training-EStimation-Training) algorithm as a way of using MLP to impute missing data. It consist of three steps. Firstly, an MLP is trained with all complete patterns. Secondly, use the network parameters (weights) to estimate the missing data in all incomplete patterns by means of backpropagation in the inputs. Thirdly, the network is trained again using the whole data set, i.e, complete and imputed patterns.

However, this procedure cannot estimate missing values in the test set. Recently, an useful neural network approach based on Multi-Task Learning (MTL) has been proposed (61], [62]). This procedure (MTLimpute) combines missing data imputation and pattern classification in one network. MTLimpute utilizes the incomplete features as extra tasks, and learns them in parallel with the main
classification task. Outputs that learn incomplete features are used to estimate missing values during learning process. Imputed values are those that contribute to improve the classification accuracy, because the learning of imputation tasks is oriented by the learning of the main classification task. By doing this, missing data imputation is oriented to solve the classification task. Although not strictly an ANN, Support Vector Machines (SVM) have been also used for missing data imputation, where predicted values generated by the SVM model used as imputed ones [109].

2.3.2. Imputation based on Recurrent Neural Network

A Recurrent Neural Network (RNN) is an architecture with feedback connections from its units. Bengio and Gingras [14] propose RNN with feedback into the input units for estimating missing data. Firstly, missing values are initialized with mean imputation, and these values are updated using the feedback connections, while the network is trained to learn the classification task. Missing values are modified as a function of missing input in the past iteration and the weighted sum of a set of recurrent links from the other units (hidden and missing) to the missing unit with an unit delay. Parveen [129] has extended RNNimpute using true values of incomplete features as extra targets during training.

2.3.3. Imputation based on Auto-Associative Neural Network Imputation

An Auto-Associative Neural Network (AANN), also known as autoencoders, is a set of neurons that are completely connected in the sense that each neuron receives input from, and sends output to, all the other neurons. Some works tackle the missing data imputation by means of this kind of networks ([120], [111], [113], [122]). In [120] and [111], an AANN learns from complete cases, in order to duplicate all of the inputs as outputs. After that, when unknown values are detected, the weights are not updated.
Instead, missing values are replaced by the network outputs. Marwala and Chakraverty [113] and [122] propose a method that combines the use of AANN with Genetic Algorithms (GA) to approximate missing data. The GA is used to estimate the missing values by optimizing a cost function between the input vector and the estimated outputs by the AANN.

2.3.4. Imputation based on K-Nearest Neighbour Algorithm

The K-nearest neighbour method select K patterns from the complete cases such that they minimize some similarity measure. The nearest, most similar, neighbours are found by minimising a distance function [13]. Once the K nearest neighbours has been found, a replacement value to substitute for the missing attribute value must be estimated. How the replacement value is calculated depends on the type of data; the mode can be used for discrete data and the mean for continuous data. An improved alternative is to weight the contribution of each neighbour according to their distance to the incomplete pattern whose values will be imputed, giving greater contribution to close neighbours [167]. An advantage over mean/mode imputation and simple hot deck method (in fact, KNNimpute with K = 1) is that the replacement values are only influenced by the most similar cases rather than by all cases or the most similar one, respectively. The main drawback of this approach is that whenever KNNimpute looks for the most similar patterns, the algorithm looks for through all training data set (in the complete data portion), which implies a high computational cost.

2.3.5. Imputation based on Self-Organizing Map

The basic Self-Organizing Map (SOM) consists of a twodimensional array of nodes which defines a mapping from the input data space onto a latent space, using weight vectors that connects each node of the SOM with input data (these weights have
the same dimensionality than the input patterns) [91]. Samad and Harp [144] implement SOM approaches to handle missing values by changing how the input data is processed. In particular, when an observation with missing features is given as input to the map, the missing variables are simply ignored when distances between observation and nodes are computed. This principle is also applied both for selecting the image-node and for updating weights.

Fessant and Midenet [56] extend this procedure to missing data imputation. Firstly, when an incomplete pattern is presented to the SOM, its image-node is chosen by ignoring the distances in the missing variables; secondly, an activation group composed of image-node’s neighbours is selected; and finally, each imputed value is computed based on the weights of the activation group’s nodes in the missing dimensions.

Following this idea, [134] implements missing data imputation in a Tree Structured SOM (TS-SOM), which is made of several SOMs arranged to a tree structure. The major advantages of this approach over the basic SOM are its faster convergence, and also, its computational benefit when the number of instances is large.

Finally, most of the imputation methods can be used following the Multiple imputation (MI) approach ([105], [150], [7]). MI is a general paradigm for the analysis of incomplete data. Each missing item is replaced by m simulated values, producing m simulated versions of the complete data. Each version is analyzed by standard complete-data methods, and the results are combined using simple rules to produce inferential statements that incorporate missing data uncertainty.
2.4. MAXIMUM LIKELIHOOD BASED APPROACHES

The methods described in this section make assumptions about the joint distribution of all variables in the model, and the model parameters are estimated using a Maximum Likelihood (ML) approach. The Expectation-Maximization (EM) algorithm is an efficient iterative procedure to compute the ML estimate in the presence of missing or hidden data.

In ML estimation, the main motive is to estimate the model parameter(s) for which the observed data are the most likely. Besides, mixture models provide a general semi-parametric model for arbitrary densities ([51], [19]), where the density functions are modeled as a linear combination of component densities. In particular, real valued data can be modeled as mixture of Gaussians (Gaussian Mixture Model, GMM); and for discrete valued data, it can be modeled as a mixture of Bernoulli densities or by a mixture of multinomial densities.

EM associates a given incomplete-data problem with a simpler complete-data problem, and iteratively finds the maximum likelihood estimates of the missing data. In a typical situation, EM converges monotonically to a fixed point in the state space, usually a local maximum. Each iteration of the EM algorithm consists of two processes: The E-step, and the Mstep. The Expectation or E-step computes the log likelihood of the data, and the Maximization or M-step finds the parameters that maximize this likelihood [66].

In the case of GMMs, the state space consists of all the possible assignments to the means, covariance matrices, and prior probabilities of the Gaussian distributions. Missing features can be treated naturally in this framework. In the expectation, or E-step, unknown values are estimated given the observed data and current estimate of the
model parameters. This is achieved using the conditional expectation. In the M-step, the likelihood function is maximized under the assumption that the missing data are known. The estimate of the missing data from the E-step is used instead of the actual missing data.

In a classification problem, the mixture modeling framework can model the class label as a multinomial variable, i.e., the mixture model estimate the joint probability that an input vector has a determined attributes and belongs to a determined class. Once the model is obtained, the most likely label for a particular input pattern may be obtained computing the class-posterior probabilities using the Bayes’ theorem ([51], [19], [66]).

Furthermore, efficient neural network approaches for handling missing data have been developed to incorporate the input data distribution modelling into the network ([5], [165], [166], [181]. Ahmad and Tresp [5] discuss Bayesian techniques for extracting class probabilities given partial data. Considering that the classifier outputs are good estimates of the class-posterior probabilities given the input vector, and splitting up into a vector of complete features and a vector of unknown features, they demonstrate that the optimal solution involves integrating over the missing dimensions weighted by the local probability densities.

Moreover, the same authors, closed-form approximations to the class probabilities are obtained using GMMs, and they extend to an MLP classifier calculating the integrals with Monte Carlo techniques. Instead of doing a numerical approximation of these integrals, ([165], [166]) propose an efficient solution using GMMs and Parzen windows to estimate the conditional probability densities that appear in the integral.
Williams et al. [181] have proposed a classification scheme with missing values based on logistic regression. In this approach, the conditional density functions are estimated with a GMM, whose parameters are obtained using both EM and Variational Bayesian EM (VB-EM). According to the authors, this method outperforms the GMM trained with EM, mean and multiple imputation, but its largest drawback is the restriction to a linear classifier.

2.5. NON-IMPUTATION METHODS

 Almost all of the studies discussed use machine learning procedures for replacing missing values by an imputed plausible value. This section reviews some studies which avoid explicit imputation methods to deal with missing data.

2.5.1. Ensemble Approaches

 Neural network ensemble models have also been used for classification of incomplete data ([152], [96], [82]). Sharpe and Solly [152] propose a procedure named Network Reduction. In this method, a set of MLPs is created, and each MLP performs classification on each different possible combination of complete features, in order to cover the complete range of attributes with missing values. The main drawback of this method is that it requires a huge number of neurons when multiple combinations of incomplete attributes are presented.

 Krause and Polikar [96] develop an ensemble of classifiers trained with random subsets of features for missing feature problem. As each classifier component of the ensemble is generated according to a weighted vector for features, it can not guarantee that all the instances have been trained on the ensemble. Therefore, useful information
maybe ignored. Juszczak and Duin [82] propose to form an ensemble of one-class classifiers trained on each feature. Thus when any input values are missing for a data point to be labeled, the ensemble can still make a reasonable decision based on the remaining classifiers.

2.5.2. Decision Trees Algorithms

In this kind of methods, we stand out two well-know approaches: ID3 and C4.5. These procedures can handle missing values in any attribute for both training and test sets ([138], [178]). ID3 is a basic top-down decision tree algorithm that handles an unknown attribute by generating an additional edge for the unknown. Thus “unknown” has been taken as a new possible value for each attribute and treated in the same way as other values.

C4.5 is an extension of ID3 proposed by [138]. It uses a probabilistic approach to handle missing values in the training and test data set. In this approach, missing data handling is different in training and testing stage. During training, each value for an attribute is assigned to a weight ([138], [178]). If the value an attribute is known, then the weight is established equal to one, otherwise, the weight of any other value for that attribute is the relative frequency of that attribute. On the testing phase, if a test case is incomplete, it explores all available branches (below the current node) and decides the class label by the most probabilistic value.

2.5.3. Fuzzy Approaches

Several fuzzy procedures have been developed in order to handle missing data ([78], [60], [16]). Ishibuchi and Moriola [78] propose an MLP classifier where unknown values are represented by interval inputs. For example, if the input space of a
particular classification problem is the d-dimensional unit cube \([0, 1]^d\), each unknown feature value is represented by the interval input \([0, 1]\) that includes all the possible values of that attribute.

When the attribute value is completely known, it is also represented by intervals (e.g., 0.3 is represented by \([0.3, 0.3]\)). This network is trained by means of back-propagation algorithm for fuzzy input vectors [78]. In [60] a General Fuzzy Min-Max (GFMM) neural network using hyperbox fuzzy sets was developed. A hyperbox defines a region of the d-dimensional pattern space, by its min-point and its max-point, and all patterns contained within the hyperbox have full class membership.

Learning in the GFMM neural network for classification consists of creating and adjusting hyperboxes in pattern space [60]. This procedure handles missing values in a similar way than the interval inputs, i.e., the missing attribute is modeled by a real valued interval spanning the whole range of values. Other solution is a fuzzy rule based classifier that consists of rules set for each possible category, where each rule can be decomposed into individual one-dimensional membership functions corresponding to the fuzzy sets. When an incomplete pattern has to be classified, the rules are obtained using only one-dimensional membership function of the known attributes, which is computationally efficient.

2.5.4. Support Vector Machines

In recent years, some works have extended the standard formulation of Support Vector Machines (SVMs) for dealing with missing values and uncertainty in the input data ([17], [130], [18]). Bhattacharyya et al. [17] propose a mathematical programming method to deal with uncertainty in the observations of a classification problem. They extend the standard SVM classifier to handle missing values replacing the linear
classification constraints by a probabilistic one. It is done modeling the missing variables as random variables by means of a Gaussian mixture [17].

Moreover, the model parameters (mean and covariance matrices) are estimated by means of the EM algorithm. Pelckmans et al. [130] propose a modified risk function taking into account the uncertainty of predicted outputs when missing values are involved. It is done by incorporating a probabilistic model for the missing data. This method generalizes the approach of mean imputation in the linear case, and the proposed kernel machine reduces to the standard SVM when no input values are missing [130]. Bi and Zhang [18] develop a novel formulation of support vector classification, which allows uncertainty in input data assuming that inputs are subject to an additive noise that follows certain distribution.

2.6. ASSOCIATIVE RULE MINING AND IMPUTATION

Associative rule mining in incomplete datasets is another technique that is considered in the present research work. The various works connected to this topic is reviewed in this section.

2.6.1. Associative rule mining

Ali et al. [6] Use association rule mining to do a partial classification in the context of very large numbers of class attributes, when most attribute values are missing, or the class distribution is highly skewed and the user is interested in understanding the low frequency classes.

Arotaritel [8] has discussed a fuzzy data mining algorithm in order to extrapolate the unknown values. Association rules can be used to remedy a particular
attribute with missing values in a given database [84] or all attributes with missing data [153].

In this processing, the number of discovered rules called strong rules can be so large that browsing the rules set and finding interesting rules from it can be quite difficult for the user [88]. So some interesting measure methods or models have been proposed ([160], [102]). Association rule mining based methods have been discussed for various application domain of missing data, such as that in ([81], [65], [93]). The successful use of association rule mining motivates us to study missing data imputation techniques continuously.

2.6.2. Automatic Threshold Estimation

In the past few years, there has been work on challenging the canonical support-confidence framework for associations mining. These efforts can be categorized into two paradigms: extending the constant support constraint and/or seeking substitutes for confidence measures.

The uniform support constraint was first argued by [71] while generalizing the association model into multiple-level associations on account of item hierarchy. The authors extended the uniform support constraint to a form of level-by-level, decreasing assignment. That is, items at the same level receive the same minimum support, and higher level items have larger support constraint. This level-wise support specification accounts for their progressive mining approach: an a priori-like algorithm is performed progressively from the top level to the bottom, and stops at the very level when no frequent itemset is generated.
Another form of association rules mining with non-uniform minimum supports was proposed by [106]. Their method allows the users to specify different minimum supports for different items, and the support constraint of an itemset is defined as the lowest minimum item support among the items in the itemset. The motivation is that the supports of items are non-uniform by nature, and high profit items (e.g. TV) usually occur less frequently than low value items (e.g. toothpaste). The multi-supported model was then extended to generalized associations with taxonomy information by [168]. The problem remains dangling.

Wang et al. [177] proposed a bin-oriented, non-uniform support constraint. Items are grouped into disjoint sets, called bins, and items within the same bin are regarded as non-distinguished with respect to the specification of minimum support. In particular, each support constraint specifies a set of bins $B_1, B_2, \ldots, B_s$ of the form $SC_i(B_1, B_2, \ldots, B_s) \geq \theta_i$, where $s \geq 0$ and $\theta_i$ is a minimum support. The end-user still needs to determine the appropriate bins and $\theta_i$ prior to the mining.

In general, the support of an itemset decreases when its length increases. The uniform support constraint may thus hinder the discovery of frequent long itemsets. In some applications, however, it may be interesting to discover associations between long itemsets. To solve this problem, Seno and Karypis [151] used a support constraint that decreases with the length of the itemset, which helps to find long itemsets without generating lots of spurious short itemsets.

There is also work on mining high confidence associations without support constraints [101]. The proposed method, however, is restricted to discovering all top rules ($\text{conf} = 100\%$) with the consequence being given.
As pointed out by many researchers ([25], [26], [2]), the primary deficiency of confidence-based associations is their poor predictive ability, i.e. the confidence measure is unable to capture the real implication. For illustration, consider the example transaction database (D) in Table 2.1.

**TABLE 2.1**

TRANSACTION DATABASE

<table>
<thead>
<tr>
<th>TID</th>
<th>Items purchased</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>PC, Printer, PDA</td>
</tr>
<tr>
<td>12</td>
<td>Printer, Notebook</td>
</tr>
<tr>
<td>13</td>
<td>Printer, Scanner</td>
</tr>
<tr>
<td>14</td>
<td>PC, Printer, Notebook</td>
</tr>
<tr>
<td>15</td>
<td>PC, Scanner</td>
</tr>
<tr>
<td>16</td>
<td>Printer, Scanner</td>
</tr>
<tr>
<td>17</td>
<td>PC, Scanner</td>
</tr>
<tr>
<td>18</td>
<td>PC, Printer, Scanner, PDA</td>
</tr>
<tr>
<td>19</td>
<td>PC, Printer, Scanner</td>
</tr>
</tbody>
</table>

For a minimum support of 30% and minimum confidence of 60%, the following association rule is discovered:

Scanner $\Rightarrow$ Printer (sup = 44.4%, conf = 66.7%)

One may conclude that this rule is interesting because of its high support and high confidence. However, note that the support of Printer is 77.8%. This means that a customer who is known to purchase Scanner is less likely (by 11%) to buy Printer than a customer about whom there is no information. The rule is misleading as it does not conform to what the direct association means: when people buy Scanner, they are also
likely to buy Printer. Instead, buying Scanner and purchasing Printer are negatively associated.

To remedy the above deficiency, two alternative measures have been proposed. They are lift [15] (also known as interest) and conviction. For an association rule \( A \Rightarrow B \), the lift is defined as

\[
lift(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{\sup(A \cup B)}{\sup(A)\sup(B)} = \frac{\text{conf}(A \Rightarrow B)}{\sup(B)}
\] (2.1)

Lift measures the deviation of the rule from independence. The farther the value is from 1, the higher the dependence will be. Lift values above 1 indicate positive dependence, while those below 1 express negative dependence. The conviction is

\[
\text{conv}(A \Rightarrow B) = \frac{P(A)P(\neg B)}{P(A \cup \neg B)} = \frac{1-\sup(B)}{1-\text{conf}(A \Rightarrow B)}
\] (2.2)

which measures the implication strength of the rule from statistical independence. The conviction value of a rule is between 0 and \( \infty \). A larger than 1 value indicates that it is greater than the expected presence.

Conviction appears to be preferable to lift in capturing the natural semantics of directed associations because it is directed, i.e. \( \text{conv}(A \Rightarrow B) \neq \text{conv}(B \Rightarrow A) \), whereas lift is not. Furthermore, conviction has better discrimination power. To see this, let us consider a rule \( A \Rightarrow B \). Assume \( \sup(A) = 10\% \), \( \sup(B) = 90\\% \) and \( \sup(A \cup B) = 10\% \). The lift(\( A \Rightarrow B \)) is calculated as \( 0.1/(0.1 \times 0.9) = 1.11 \) which is only slightly above 1. But this rule has the highest possible conviction value of \( \infty \), which conform to its 100% confidence.
There is also work on investigating alternatives to the association model for attribute set mining. Brin et al. [25] first proposed the correlation framework, aiming at mining strongly correlated attribute sets. They adopted the well-known chi-squared test from classical statistics to measure the correlation. However, this measure, though statistically precise, is prohibitively expensive in constructing the contingency table for each itemset. Another criterion for measuring correlation called collective strength (cs) was thus proposed by [2], which is defined as follows:

$$cs(A) = \frac{1 - v(A)}{1 - E[v(A)]} \cdot \frac{E[v(A)]}{v(A)}$$  \hspace{1cm} (2.3)

where $v(A)$ denotes the violation rate of an itemset $A$, i.e. the fraction of transactions which contain a proper non-null subset of $A$, and $E[v(A)]$ the expected number of violations of itemset $A$. However, although in theory the correlation framework would discover strongly correlated items without the support constraint, in practice the support threshold is still of essence. Without a support threshold, the computation cost will be prohibitively expensive and many unqualified itemsets will be generated. As such, users still confront the problem of appropriate support specification.

In the work proposed by [45], instead of searching for high confidence associations, they focus on identifying similar itemsets (column pairs) without any support threshold. To this end, they have introduced a new measure, called similarity, whose symmetric property enables the elimination of support constraint. This method has the disadvantage that it is not feasible for applications that adhere to the traditional asymmetric confidence measure.

The primary problems of a support-confidence framework are poor predictive ability and uniform support constraint. Although substantial work has provided various
methods to alleviate these problems, such as adding the lift or conviction measure and using non-uniform support constraint, there is still no guideline for users in the support specification. The user has to either, at the cost of computational efficiency, set the support constraint low enough so as not to lose any interesting rules, or risk missing new insight patterns.

2.7. CONCLUSION

Many applications in the real world consider missing value as a bottleneck problem, which has to be resolved before mining of knowledge. From the literature survey conducted and as [158] noted the “range of potential approaches to the problem of missing data is very broad, from ignoring the problem altogether to sophisticated mathematical techniques for predicting what data would have appeared in the missing cells”. Further, as pointed out by [28], “No one missing data technique is optimal for every missing data situation”.

Further it is also understood that the correct choice of a missing data treatment is a hard and complex task, i.e., a method can work well in some problems, and in contrast, its results are not good in other applications. All these make research in missing value an on-going process, with the common goal of finding an efficient algorithm that can accurately replace the missing values so as to improve the performance of the data mining techniques.

In the present research work focus on proposing methods that treat missing values and analyze its effect on classification accuracy. The methodology and research design used is given in the next chapter, Methodology.