DESIGNING INTELLIGENT EXPERT SYSTEMS TO COPE WITH LIARS

Completed Research Paper

Yuanfeng Cai
College of Business
Iowa State University
3132 Gerdin Business Building
Ames, IA 50011-1350
yfcai@iastate.edu

Zhengrui Jiang
College of Business
Iowa State University
2340 Gerdin Business Building
Ames, IA 50011-1350
zjiang@iastate.edu

Vijay S. Mookerjee,
School of Management
University of Texas at Dallas
800 West Campbell Road
Richardson, Texas 75083-0688
vijaym@utdallas.edu

Abstract

To cope with the problem of input distortion by users of Web-based expert systems, we develop methods to distinguish liars from truth-tellers based on verifiable attributes, and redesign the expert systems to control the impact of input distortion. The four methods we propose are termed split tree, consolidated tree, value based split tree, and value based consolidated tree. They improve the performance of expert systems by improving accuracy or reduce misclassification cost. Numerical examples confirm that the most possible accurate recommendation is not always the most economical one. The recommendations based on minimizing misclassification costs are more moderate compared to that based on accuracy. In addition, the consolidated tree methods are more efficient than the split tree methods, since they do not always require the verification of attribute values.

Keywords: Expert systems, input distortion, noise handling, misclassification

1. Introduction

Expert systems are widely applied in diverse fields such as medical treatments, production management, and financial investing (Liao 2005). The mechanism behind expert systems is that they replicate experts' knowledge in a specialized domain to establish their decision rules. Based on information provided by users, expert systems can arrive at a particular recommendation to solve problems. Similar to consultants, some expert systems can provide suggestions as well as explanations (Turban and Aronson 2001). Organization can take advantage of expert systems to reduce the cost of human experts or make better decisions (Duan et al. 2005).

Broadly speaking, there are two types of expert systems: inductive expert systems and deductive expert systems. Inductive expert systems are built using induction algorithms, which develop decisions rules based on pre-classified training datasets (Mookerjee and Dos Santos 1993). Deductive expert systems, on the other hand, are built on deductive algorithms, which derive rules based on existing knowledge base and additional evidence through deductive reasoning (Zhang and Wu 2010). Instead of learning decision rules from training data, decision rules for a deductive expert system could be directly provided by human experts. In this study, our focus is on deductive expert systems.

Internet technologies have provided new opportunities for the deployment and wider application of expert systems (Power 2000). Through Web-based interfaces, users can conveniently access an expert system from different locations and recommendations can be easily delivered to them. However, along with the greater conveniences, Web-based interfaces also bring some challenges, which we discuss next.

1.1. Input Distortion

Input distortion occurs when Internet users are not willing to disclose their true personal information online when they are required to. Hoffman et al. (1999) find that 95% of the users are reluctant to provide information requested by websites. One reason behinds this behavior is the lack of fundamental trust between consumers and businesses on the Web today (Merzger 2004). Users' concern about their privacy and information security prevents them from revealing true information. Therefore, users may falsify input data to protect themselves. Another important factor that contributes to this lying behavior is that self-interested customers may deliberately seek improper benefits by providing incorrect data. For example, during a credit card application, users who are not confident about their financial background may manipulate their partial information in order to get approval. This type of lying behavior is further exacerbated by the fact that without face-to-face interaction with users, lying is more difficult and costly to detect.

Regardless of the causes of lying, firms can incur significant costs as a result of incorrect input data. In the scenario of a credit card application, granting the card to high-risk customers can result in financial losses. On the other hand, incorrectly denying deserving customers can lead to loss of potential revenue and impair firm's reputation. One intuitive method to solve the falsified credit application is to impose penalty. Worsham (2010) has reported that prison sentences range from a period of months to years and fines upwards of $200,000 or more may be charged for falsifying information in a credit application. However, punishments are often costly to enforce, thus may have limited effect on the prevention of input distortion. Another possible method is to utilize incentive mechanisms to discourage users from lying. Incentive mechanisms are easy to carry out but their goals are hardly realized as long as the users perceive the benefit of lying is greater than the incentive.

Since input distortion is practically impossible to completely eliminate, one may suggest that all user inputs be manually verified to ascertain their accuracy. However, manually verifying users inputs for a frequently used expert systems, such as one used for consumer credit screening, will be costly, thus offsetting the benefits and the conveniences of expert systems. Moreover, manually verification process is typically time-consuming and resource intensive. Therefore, manual verification is not a feasible approach to deal with user input distortion for most expert systems. In this study, we focus on automatic approaches to address users' lying behavior when using expert systems.

1.2. Literature Review

Human's lying behavior has long been studied by researchers. One research stream deals with deception detection. Previous studies suggest that it is possible to use verbal and nonverbal cues to detect deception (Buller and Burgoon 1996, George et al. 2004). In addition, researchers have proposed methods to detect deception via linguistic cues (Zhou et al. 2003 ; Zhou et al. 2008 ; Zhou and Zhang 2008). However, the techniques for deception detection cannot be directly applied to address users' lying behavior in our problem context. For example, during an online credit card application, consumer may only input numeric and simple text information (e.g., name and address). Without face-to-face contact, it is not possible to capture non-verbal or verbal cues that are critically important for deception detection. Similarly, without rich text information, the linguistic based on methods also cannot be applied. Furthermore, existing studies on deception detection do not address input distortion for expert systems.

Another stream of research deals with noise handling for inductive expert systems. For inductive expert systems, noisy data can affect the derived decision rules and subsequently the recommendations. One way of dealing with noisy data is enhancing the quality of training data. Various solutions have been proposed for this purpose, such as class noise identification (Zhu et al. 2003; Brodley and Friedl 1999), erroneous attribute value location (Zhu et al. 2004), and missing attribute value imputation (Fellegi & Holt 1976, Rubin 2004). One problem associated with this type of methods is that important information may be lost during this elimination process (Wu & Zhu 2008). Another important method is decision tree pruning, which could improve the tree performance under noisy data (Quinlan 1986). Mookerjee et al. (1995) apply the pruning technique during the tree construction phase instead of after it. Boylu et al. (2010) adapt support vector machines (SVM) to generate classification; their method takes into consideration users' possible strategic behavior such as distorting data. To evaluate their relative performance, Zhou et al. (2004) empirically compare various noise handling techniques and find that only neural networks exhibited consistent performance. Although these solutions improve the performance of inductive experts systems, they all require that a similar error pattern exist in training and testing data, which is not applicable to deductive expert systems.

For deductive expert systems, decision rules are typically provided by domain experts instead of induced from training data, hence we expect that noisy input data affects only the recommendations but not the decision rules used to build system. When formulating decision rules, experts typically assume that the input data at the time of consulting will be accurate, i.e., no lying will occur. However, as discussed earlier, this is often not the case. To cope with input distortion, Jiang et al. (2005) propose two novel approaches to improve the accuracy of recommendations. The first approach, termed knowledge modification (or KM), generates a new decision tree based on experts' decision rules as well as users' lying patterns. At the time of consulting, users' input data would be directly fed into the modified decisions tree. The second approach, termed input modification, still uses the decision tree built from decision rules provided by experts, but modifies a user's input data at the time of consulting. Jiang et al. show that both the knowledge modification method and the input modification method lead to a significantly improved accuracy than the traditional method that ignores input distortion, with the knowledge modification method outperforming the input modification methods for almost all problem scenarios.

Although the method proposed by Jiang et al. substantially increases the accuracy of recommendations under input noises, the method have two limitations. First, the KM method does not attempt to differentiate liars from truth-tellers at the time of consulting. Instead, all user provided inputs are directly fed into the same KM tree in the same way. Second, the KM method does not consider misclassification costs. In the real-world, misclassification costs are often asymmetric. For instance, classifying a non-worthy customer as a worthy one could potentially be much more costly than classifying a worthy customer as a non-worthy one. The KM method maximizes the expected accuracy of recommendations while completely ignores such misclassification costs.

1.3. Contributions

To the best of our knowledge, no previous study differentiates liars from truth-tellers and considers misclassification costs when dealing with input noises for deductive expert systems. Our study fills this void and makes two important contributions. Our first major contribution is that we differentiate between liars and truth-tellers in all methods proposed in this study. By comparing the user-provided value and

the verified true value for an attribute, we first calculate the probability that a user is a liar, and the user may be treated differently based on the calculated probability. The first method we propose is termed *split tree method* (ST). If a customer's probability of liar is above a threshold, she is treated as a liar and her inputs are fed into a *Liar tree*, which is built based the pattern of input distortion by liars. Otherwise, she is treated as a truth-teller and the recommendation is generated using the *True tree* (TT) built directly from decision rules provided by experts while ignoring all input distortions. Using the split tree method, one attribute value needs to be verified for every user. The second method is termed as *consolidated tree method* (CT). By taking into consideration all input scenarios under both the True tree and the Liar tree, this method re-computes a new consolidated tree that is more efficient at the time of consulting. For each possible input vector, the consolidated tree always selects the recommendation with the highest probability of being accurate. With the consolidated tree method, a user's true attribute values may not be needed for certain input vectors, hence it is more efficient than the split tree methods. We expect that the two methods will lead to better accuracy than the original KM method proposed by Jiang et al (2005).

As the second major contribution, we propose approaches to minimize the total misclassification cost. Existing approaches focus on improve accuracy of the recommendations. However, a tradeoff may exist between accuracy and misclassification costs. Therefore, we first propose two value-based methods. The first one is extended based on ST so it is termed *value-based split tree method* (VST) while the second one is modified based on CT and is named as *value-based consolidated tree method* (VCT). The primary difference between the value-based methods and the accuracy-based methods is that the first two methods generate recommendations that maximize accuracy, while the last two provide recommendations that minimize the expected misclassification cost. The two value-based methods can be considered generalizations of the accuracy-based methods, and are particular useful when misclassification costs are asymmetric.

The rest of the paper is organized as follows. In Section 2, we propose two accuracy-based methods. In Section 3, the two value-based methods are developed. Section 4 summarizes findings and Section 5 discusses implications of this research.

2. Accuracy-Based methods

Deductive expert systems make decisions based on decision rules that are typically provided by human experts. For better efficiency at the time of consulting, such decision rules need to be transformed into a decision tree, which is then used to generate recommendations based on inputs provided by users. To differentiate it from decision trees that are generated from other methods, we call the decision tree built on decision rules provided by experts as the *True Tree* (*TT*).

When the decision rules are formulated, experts implicitly assume that all input provided by users are accurate. For instance, an expert may classify a customer with medium income and full-time employment as low-risk. An implicit assumption behind this decision rule is that the customer indeed has a medium income and a full-time job. However, as discussed earlier, users may lie when providing inputs to the expert systems. In the same example, if a customer who claims to have a medium income and a full-time job is actually unemployed with no significant income, then classifying the customer as a low-risk one could potentially lead to financial losses. Facing such lying behavior, a *KM tree*, built based on the knowledge modification method proposed by Jiang et al. (2005), can replace the True tree to serve as the "expert" at the time of consulting. The KM method, however, does not differentiate liars from true-tellers, and hence leaves room for improvement. In this section, we propose two accuracy-based methods to address this challenge: the *Split Tree* (ST) method and the *Consolidated Tree* (CT) method. Both the ST and CT methods try to estimate the probability that a particular customer is a liar or not, and may provide different recommendations based on the calculated probability.

To illustrate the typical problem context for deductive expert systems and the different noise handling methods, we first present a credit risk assessment example that is similar to the one used by Jiang et al. (2005).

2.1. A Credit Risk Assessment Example

In order to decide whether to provide credit to potential customers, firms need to first assess their credit-worthiness. Human experts could provide a set of decision rules, as shown in Table 1, that can be used in such an assessment. Based on rules represented in this table, a customer can be classified into three risk levels, i.e., low risk (LR), medium risk (MR) and high risk (HR), based on the attribute values for four attributes: *Income* (high, medium, low), *Bachelor's Degree* (yes, no), *Employment* (yes, no), and *Bankruptcy* (yes, no). The dash entry ("-") in the table means that the value for that attribute "does not matter" for a given decision rule. For instance, rule R0 classifies a customer as low-risk as long as the customer's income is high, regardless of whether the customer has a degree, a bankruptcy history, or an employment. Since the decision rules shown in Table 1 are provided based on the assumption that all attribute values are correct, they represent the *True Table*. Based on heuristic algorithms, the True table can be translated into a True Tree, as shown in Figure 1.

Table 1. Decision Table for the Credit-Granting System Example

Attributes / Rules	Income	Bachelor's Degree	Employment	Bankruptcy	Classification
R0	H	-	-	-	**LR**
R1	M	Y	Y	-	**LR**
R2	M	-	N	-	**MR**
R3	M	N	Y	-	**MR**
R4	L	-	-	Y	**HR**
R5	L	-	-	N	**MR**

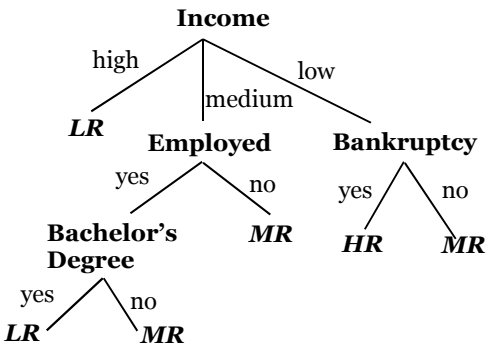

Figure 1. True Tree for the Credit Risk Assessment Example

2.2. Judging A User is Liar or Not

In order to estimate the probability that a particular user is a liar or true-teller, we need to compare the value provided by the user and the corresponding true value for at least one attribute. The attribute used for verification purposes should be relatively easy to validate. We call such an attribute *Verifiable Attribute* (or VA). In the credit rating example, Bankruptcy could be used as an VA since it can be relatively quickly and inexpensively obtained from a customer's credit report. Given both the observed and true value of an VA, denoted by VA^O and VA^T respectively, the conditional probability that the user is a liar can be derived based on the Bayesian theorem:

$$P(\text{liar} \mid VA^O, VA^T) = \frac{P(VA^O, VA^T \mid \text{liar}) \cdot P(\text{liar})}{P(VA^O, VA^T \mid \text{liar}) \cdot P(\text{liar}) + P(VA^O, VA^T \mid \text{truth - teller}) \cdot P(\text{truth - teller})}, \quad (1)$$

where

$$P(VA^O, VA^T \mid \text{liar}) = P(VA^O \mid VA^T, \text{liar}) \cdot P(VA^T, \text{liar}), \text{ and}$$
$$P(VA^O, VA^T \mid \text{truth-teller}) = P(VA^O \mid VA^T, \text{truth-teller}) \cdot P(VA^T, \text{truth-teller}).$$

Given (1), we have

$$P(\text{truth-teller} \mid VA^O, VA^T) = 1 - P(\text{liar} \mid VA^O, VA^T). \quad (2)$$

The conditional and marginal probabilities shown in (1) can be obtained from historical data or through sampling. During the sampling process, a certain number of users are selected and their true attribute values verified. A user who lied about at least one attribute is classified as a liar; those who did not lie about any attribute are classified as truth-tellers. Based on this classification, we can estimate the distribution of liars and truth-tellers in the population, as the one shown in Figure 2. In addition, we can estimate the *distortion matrixes* for liars and truth-tellers for each attribute, as shown in Figure 3. In a distortion matrix, the rows represent the true values while the columns represent the observed values. The numbers represent the conditional probability of every observed attribute value for each true value. For instance, the number 0.25 in the liar's distortion matrix for *Income* implies that among those whose true income is low, 25% are likely to claim that their income is high. From sampling, we can also estimate the marginal distribution of true attribute values for both liars and truth-tellers, as shown in Figure 4. Here, the marginal distributions need to be separately estimated for liars and truth tellers. Note that these distortion matrices and marginal distributions are similar to those estimated by Jiang et al. (2005). However, Jiang et al. do not estimate them separately for liars and truth tellers. Instead, distortion matrices and marginal distributions are estimated based on all users included the sample.

Liar (L)	0.4
Truth-Teller (T)	0.6

Figure 2. Distribution of Liar and Truth-Teller

Income (I)

	High	Medium	Low
High	0.85	0.1	0.05
Medium	0.6	0.35	0.05
Low	0.25	0.35	0.4

Bachelor's Degree (D)

	Yes	No
Yes	0.9	0.1
No	0.7	0.3

Employed (E)

	Yes	No
Yes	0.85	0.15
No	0.90	0.10

Bankruptcy (B)

	Yes	No
Yes	0.11	0.89
No	0.01	0.99

Figure 3a. Liars' Distortion Matrices for Each Attribute

Income (I)

	High	Medium	Low
High	1.0	0.0	0.0
Medium	0.0	1.0	0.0
Low	0.0	0.0	1.0

Bachelor's Degree (D)

	Yes	No
Yes	1.0	0.0
No	0.0	1.0

Employed (E)

	Yes	No
Yes	1.0	0.0
No	0.0	1.0

Bankruptcy (B)

	Yes	No
Yes	1.0	0.0
No	0.0	1.0

Figure 3b. Truth-Tellers' Distortion Matrices for Each Attribute

Income (I)

High	0.5
Medium	0.3
Low	0.2

Bachelor's Degree (D)

Yes	0.55
No	0.45

Employed (E)

Yes	0.65
No	0.35

Bankruptcy (B)

Yes	0.45
No	0.55

Figure 4a. Liars' Marginal Distributions for Each Attribute

***Income* (*I*)**	
High	0.67
Medium	0.3
Low	0.03

***Bachelor's Degree* (*D*)**	
Yes	0.7
No	0.3

***Employed* (*E*)**	
Yes	0.73
No	0.27

***Bankruptcy* (*B*)**	
Yes	0.03
No	0.97

Figure 4b. Truth-Tellers' Marginal Distributions for Each Attribute

Given the distributions and conditional probabilities included in the distortion matrices, we can estimate the probability that a user is a liar given the true and observed values of an attribute. To illustrate, consider an observed vector (I^O = "H", D^O = "N", E^O = "N", B^O = "N"), representing the observed values of *Income*, *Bachelor's Degree*, *Employment*, and *Bankruptcy*, and a verified true VA (Bankruptcy) value B^T = "N". We next show how to use the numbers shown in Figures 2-4 to estimate the conditional probability that this customer is a liar. Based on (1), we first obtain

$$P(\mathrm{B}^{\mathrm{O}}=\text{"N"}, \mathrm{B}^{\mathrm{T}}=\text{"N"}|\, liar) = P(\mathrm{B}^{O}=\text{"N"}|\, \mathrm{B}^{\mathrm{T}}=\text{"N"}, liar)\cdot P(\mathrm{B}^{\mathrm{T}}=\text{"N"}, liar) = 0.99*0.55, \text{ and}$$

$$P(\mathrm{B}^{\mathrm{O}}=\text{"N"}, \mathrm{B}^{\mathrm{T}}=\text{"N"}|\, truth-teller) = P(\mathrm{B}^{O}=\text{"N"}|\, \mathrm{B}^{\mathrm{T}} = \text{"N"}, truth\text{-}teller)\cdot P(\mathrm{B}^{\mathrm{T}}=\text{"N"}, truth\text{-}teller) = 1.0*0.97.$$

Then,

$$\mathrm{P}(\mathrm{liar}|\, \mathrm{B}^{\mathrm{O}}=\text{"N"}, \mathrm{B}^{\mathrm{T}}=\text{"N"}) = \frac{\mathrm{P}(\mathrm{B}^{\mathrm{O}}=\text{"N"}, \mathrm{B}^{\mathrm{T}}=\text{"N"}|\, \mathrm{liar})\cdot \mathrm{P}(\mathrm{liar})}{\mathrm{P}(\mathrm{B}^{\mathrm{O}}=\text{"N"}, \mathrm{B}^{\mathrm{T}}=\text{"N"}|\, \mathrm{liar})*\mathrm{P}(\mathrm{liar}) + \mathrm{P}(\mathrm{B}^{\mathrm{O}}=\text{"N"}, \mathrm{B}^{\mathrm{T}}=\text{"N"}|\, \mathrm{truth\text{-}teller})*\mathrm{P}(\mathrm{truth\text{-}teller})} = \frac{0.99*0.55*0.4}{0.99*0.55*0.4+1.0*0.97*0.6} = 0.272.$$

The result means that even if the observed and true values of Bankruptcy are both "N," the user still has a 27.2% chance of being a liar. This has an important implication. Even though the observed value is the same as the true one, it does not guarantee that the customer is a truth-teller. On the other hand, a user is a liar with certainty if the observed value is not the same as the truth value. This can be verified based on (1). If B^O is not equal to B^T, P (B^O | B^T, truth-teller) =0. Thus, P (liar| $B^O \neq B^T$) =1, which means it is a definite liar.

2.3. Split Tree Method (ST)

Once the probability that a given user is a liar is estimated, we can decide whether to use the True Tree or a tree specifically built for liars, named *Liar Tree*, to generate recommendations. As shown in Figure 5, if the Threshold is set at 0.5, then the True tree should be consulted if the probability that a user being a liar is 0.272, as calculated in the previous numerical example. On the other hand, the Liar Tree is consulted if the user's probability of being a liar is higher than 0.5. We call this method a Split Tree (or ST) method. Under the ST method, the True Tree is directly constructed from the expert-provided decision rules; the Liar Tree can be built based on the same steps for building a KM tree (Jiang et al. 2005), with the exception that Liar's distortion matrices and marginal distributions, instead of those for all users verified during sampling, are used. We next briefly described the steps used to generate the Liar Tree.

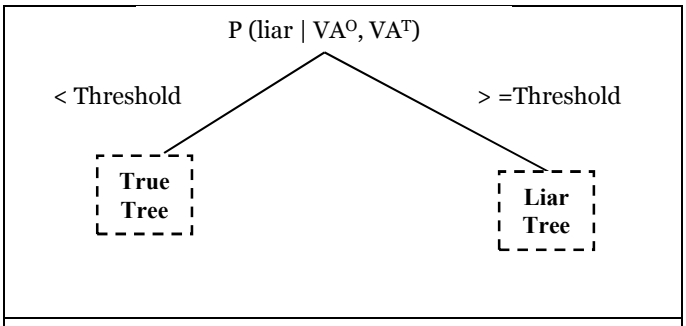

Figure 5. Split Tree Logic

To build the Liar Tree, we first need to calculate the *Liar's decision table* (or *LT table*), which includes recommendations for all possible observed vectors. For instance, for the credit risk assessment example, there are 3×2×2×2 = 24 rules in the LT table. Similarly, if there are 10 binary variables, the fully enumerated LT table will include 2^{10} rules. The recommendation for each possible observed vector is computed using the following steps:

Step 1: Given each observed vector, calculate the probabilities associated with every possible true vector.

Denote the observed and a true vector by ***Observed*** and ***True***, respectively. Based on Bayesian theorem, conditional probability is

$$P(\boldsymbol{True} \mid \boldsymbol{Observed}) = \frac{P(\boldsymbol{Observed} \mid \boldsymbol{True}) \cdot P(\boldsymbol{True})}{P(\boldsymbol{Observed})}. \quad (3)$$

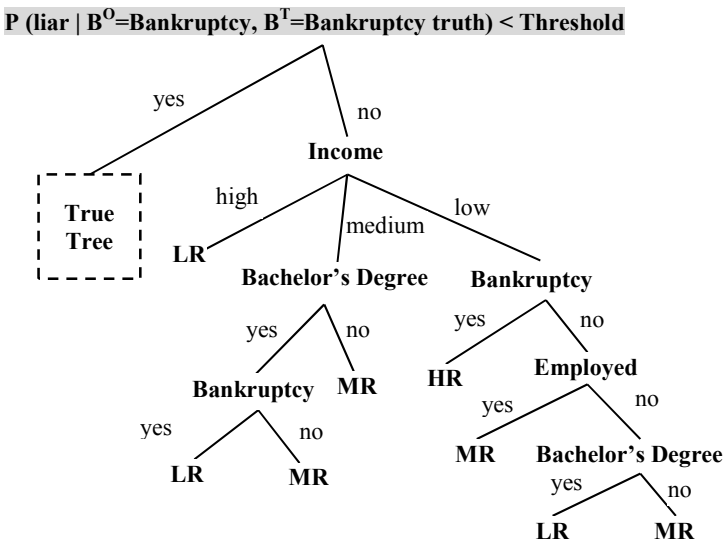

Figure 6. Complete Split Tree

The prior probability of the $P(\boldsymbol{True})$ vector can again be estimated from sampling data. If the number of possible true vectors is large, we can assume

$$P(\boldsymbol{True}) = \prod_i P(True_i),$$

where $True_i$ represents the value for attribute i in the true vector. Similar to the well-known naïve Bayes method, two other assumptions are adopted:

$P(\boldsymbol{Observed}|\boldsymbol{True}) = \prod_i P(Observed_i|\boldsymbol{True})$ for all i.

$P(Observed_i|\boldsymbol{True}) = P(Observed_i|True_i)$ for all i.

Then,

$$P(\boldsymbol{Observed} \mid \boldsymbol{True}) = \prod_i P(Observed_i \mid True_i) . \quad (4)$$

Finally, $P(\boldsymbol{Observed})$ can be calculated based on the law of total probability, i.e.,

$$P(\boldsymbol{Observed}) = \sum_r P(\boldsymbol{Observed} \mid \boldsymbol{True}^r) P(\boldsymbol{True}^r) , \quad (5)$$

where r is the index over all possible true vectors.

Step 2: Find the LT recommendation.

Use True tree to obtain the recommendation for every true vector. Add the conditional probabilities associated with all true vectors that have the same classification. The classification with the highest total probability will be selected as the LT recommendation for the observed vector ***Observed***, because this classification is most likely to be the accurate one for a user with the observed vector.

Step 3: Generate the LT table.

Repeating Step 1 and Step 2 for all possible observed vectors will generate the fully enumerated LT table.

Step 4: Create the condensed LT table

From the full table, it is easier to find that there are some redundant inputs entries which will generate the same results. The table could be condensed by removing redundant input entries.

Step 5: Build the liar tree from the condensed LT table.

The same heuristic used to build the True tree can again be used to construct the Liar Tree.

For the credit risk assessment example, based on data shown in Figures 2-4, we obtain the Liar tree based on the described steps. The complete Split is shown in Figure 6. For better clarity, the True tree structure is not included in the figure.

2.4. Consolidated Tree Method (CT)

With the ST method, the VA is always first verified before deciding which tree branch to traverse. With some careful analysis, we find that under certain situations, the true value of the VA does not matter. Motivated by this finding, we develop an alternative method to deal with users' lying behavior. With this alternative method, the true value of the VA is simply treated as a "separate" variable in the decision table. Therefore, each vector in the expanded decision includes the observed values for all variables as well as the true value of the VA. The best classification for this vector is one with the highest probability given the available information. The expanded decision table (or the CT Table) is then condensed and transformed into a single tree. We call this method Consolidated Tree method (or CT method) since there is no separate tree branches for liars and truth-tellers under this method. We next discuss how the consolidated tree can be constructed.

For each vector in the CT table, repeat Steps 1-4 to obtain its CT recommendation:

Step 1. Find the probability that the user with the given vector is a liar.
Since the given vector includes both the observed value and the real value of the VA, the probability that the user is a liar or a truth teller can calculated based on formulas (1) and (2). We denote these probabilities as P (liar | VA^O, VA^T) and P(truth-teller | VA^O, VA^T), respectively.

Step 2. Calculate LT path probability associated with each possible recommendation.
Since there is one consolidated tree, we need to consider the probability that a user is liar as well as the probability that she is a truth-teller. If the user is a liar, then similar to Step 1 in building the Liar tree, we

calculate the conditional probabilities associated with all possible true vectors and then sum up the conditional probabilities of all true vectors that have the same recommendation. Using the credit risk assessment example, we denote the total conditional probabilities associated with low-risk, medium risk, and high-risk by P(LR), P(MR), and P(HR) respectively. Since these probabilities are relevant only if she is a liar, we need to multiply each of them by P(liar| VA^O, VA^T) and obtain P(liar | VA^O, VA^T) P(LR) , P(liar | VA^O, VA^T) P(MR), and P(liar | VA^O, VA^T) P(HR). Each of these probabilities is referred to *LT path probability* associated with a recommendation.

Step 3. If P(truth-teller | VA^O, VA^T) > 0, feed the observed values into the true tree, and obtain the *True recommendation*. Since there is no uncertainty in the truth tree path, we set the *TT path probability* associated with the True recommendation as P(truth-teller | VA^O, VA^T) and that associated with all other recommendations as zero. Use the True Tree, get the recommendation with the TT path probability P(truth-teller | VA^O, VA^T)

Step 4. Obtain the CT recommendation for the given vector.

Add the LT path probabilities and the TT path probabilities for the same recommendations. The recommendation with the highest sum is selected as the CT recommendation for the given vector. For instance, if the True recommendation is MR for the credit risk assessment example, then we need to compare P(liar | VA^O, VA^T) P(MR) + P(truth-teller | VA^O, VA^T) with P(liar | VA^O, VA^T) P(LR) and P(liar | VA^O, VA^T) P(HR). The recommendation with the highest probability is selected as the CT recommendation.

Once the CT recommendations are obtained for all vectors in the CT table, we then condense it and transformed it into a True tree.

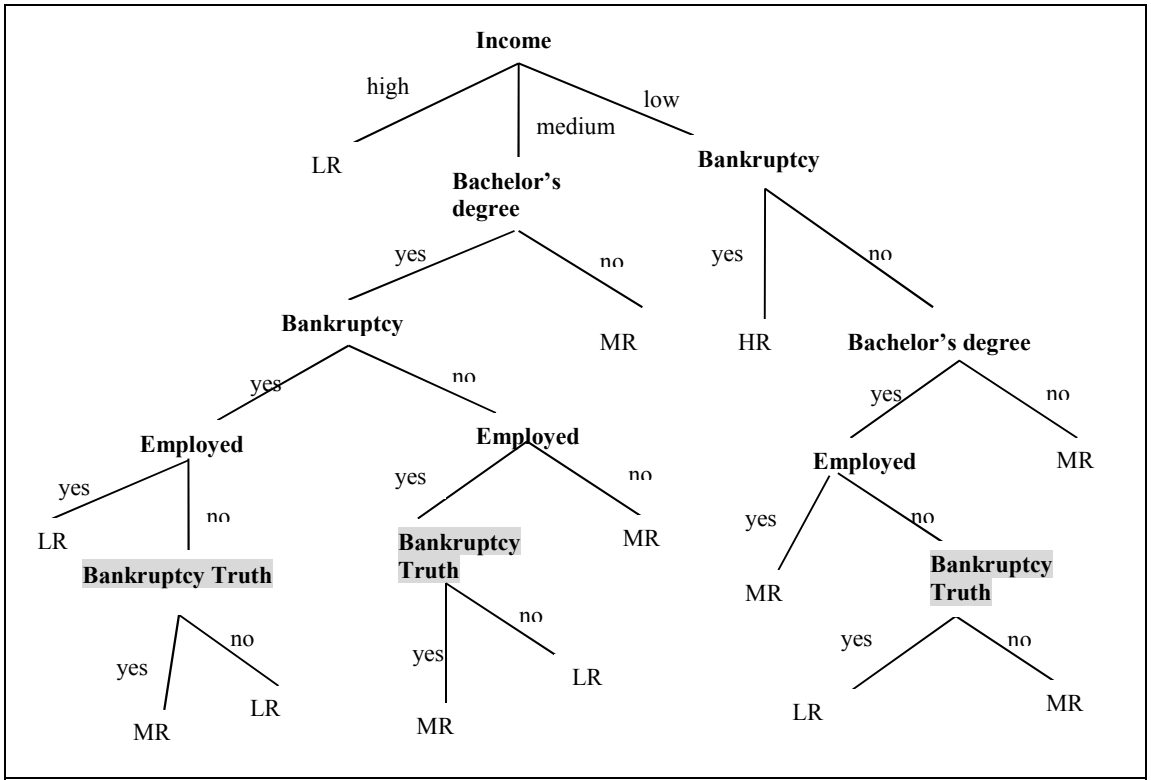

Figure 7 Consolidated tree

Following the CT method, we construct the consolidated tree for the credit assessment example, as shown in Figure 7. By Comparing the Split tree shown in Figure 6 and the Consolidated shown in Figure 7, we find that these two methods do not always leads to the same recommendations. Furthermore, under the

Consolidated Tree, a users' true value for Bankrupt do not always need to be verified in the majority of the branches. This represents a significant saving since verifying the true values of a VA may take time and cost money. Therefore, the consolidated tree method appears to be a more efficient method. We also expect the consolidated method to lead to better accuracy since when the tree is built, we maximize the accuracy of recommendation by taking into consideration the all possible paths and their associated probabilities.

3. Value-based Method

The ST and CT methods proposed in Section 3 attempts to maximize the accuracy of recommendations. An implicitly assumption made in the two methods is that the misclassification cost remain the same for all misclassification scenarios. In the real-world, this may not the case. For instance, incorrectly classifying a low-risk customer as a high risk one may lead to the denial of loan and hence lose opportunity to earn interests from the customer. On the other hand, misclassifying a high risk customer as a low-risk may lead to default. The second scenario can potentially be much more costly than the first scenario to a firm. In this section, we extend the two methods developed in the previous section to the corresponding value-based methods: *value-based split tree method* (VST) and *value-based consolidated tree method* (VCT). The main difference between the accuracy-based methods and value-based methods is that the former attempts to minimize the accuracy of recommendations, while the later makes recommendations that lead to the lowest misclassification costs.

In order to use the value-based methods can be used, we must first obtain the *misclassification cost matrix*. IN a misclassification cost matrix, the rows represent the true class, and the columns represent the recommended class. Figure 8 shows a hypothetical matrix for the credit risk assessment example. In this matrix, the entry "20" means that the cost of misclassifying a high-risk customer as low-risk one is 20.

	LR	MR	HR
LR	0	45	100
MR	10	0	50
HR	20	28	0

Figure 8 misclassification costs matrix

3.1. Value-Based Split Tree Method (VST)

Similar to the ST method, the VST method produces a split tree structure including a True tree branch and a Liar tree branch, and the branch traversed at the time of consulting depends on whether the probability that a user is a liar is above or below the given threshold or not. The difference lies in how the LT recommendation is generated when the Liar tree is constructed. In the ST method, the LT recommendation for a given observed vector is the one with the highest probability of being accurate. The VST method, on the other hand, will select the recommendation that leads to the lowest cost. Figure 9 shows a Value-Based Split Tree for the credit risk assessment example.

3.2. Value-Based Consolidated Tree Method (VCT)

Similar to the basic idea of VST method, for each given vector, the VCT method selects the recommendation that minimizes the total expected misclassification cost instead of maximizing its accuracy. The output of VCT process will be value-based consolidated tree. Figure 10 illustrates such a tree for the credit rating example.

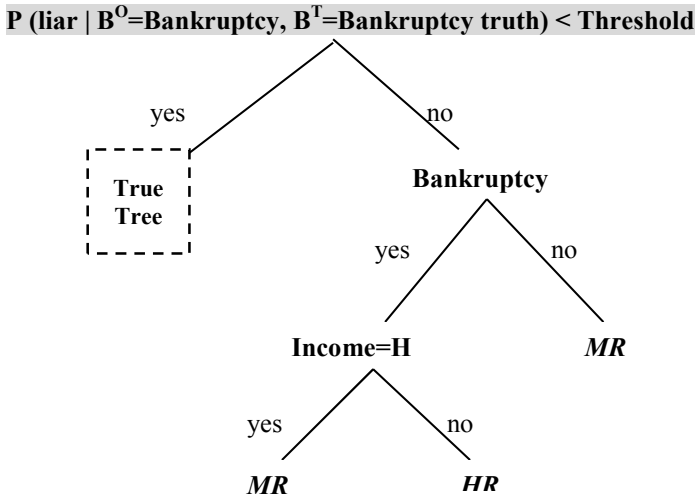

Figure 9 Value-Based Split Tree

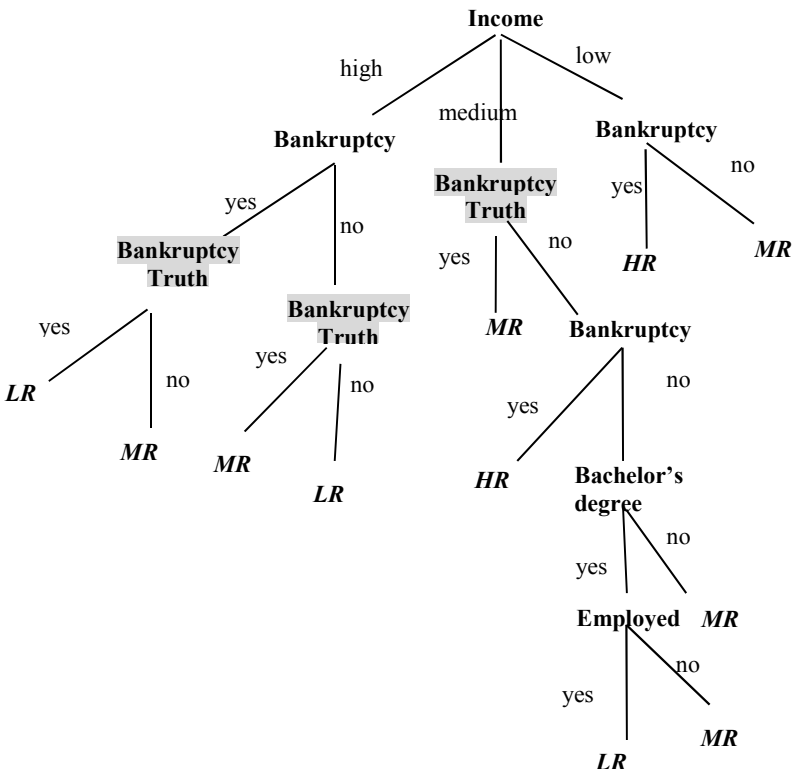

Figure 10 Value-Based Consolidated Tree

3.3. Comparison

The accuracy of recommendation has long been considered the most important criteria for evaluating expert systems. This study brings up the tradeoff between accuracy and economic value, which is reflected by misclassification costs. As shown the difference between Figure 6 and 9, Figure 7 and 10, the most accurate recommendation is not always the most economical one. Therefore, higher accuracy does not guarantee less loss. The recommendation based on minimizing misclassification costs is more moderate compared to that based on accuracy. More risky recommendation is generated when it intends to prevent loss. Furthermore, the two consolidated methods are typically more efficient than the two split methods. The split methods always require the verification of the true value of the VA, whereas the consolidated methods may only need to verify VA in certain cases.

4. Experimental Results

We conduct experiments to compare the performances of the different methods. We simulate 10,000 users in each run. For each simulated user, we randomly generate her true attribute values. The correct recommendation for this user is determined by the True Tree. Whether this user is a liar or not is randomly determined based on the assumed percentage of liars in the population. If the user is selected to be a liar, the observed attribute values are simulated based on the assumed distortion matrixes. From the 10,000 users, we select a small sample to obtain the estimated marginal distributions and distortion matrixes. The estimated parameters values are then used to build the Split Tree and Consolidated Tree. As a benchmark method, the KM tree (Jiang et al. 2005) is also constructed. In the last step, we use the observed attribute values for each of the 10,000 users to obtain the ST, CT, and KM recommendations, which are compared to the correct recommendation to determine their accuracy.

We repeat the experiment for 100 different simulated True Trees. For each True Tree, nine different distortion matrixes and eleven different liar percentage are used. Specifically, we change the liar percentage from 0 to 1 in steps of 0.1. In addition, we vary the distortion level of one variable from 0.1 to 0.9 in steps of 0.1, and fixed the distortion level for all other variables at 0.1, and record the changes in performances. The final experiment results include 9,900 rows, and each row contains the accuracy for every method. At the aggregate level, the results show that the CT method achieves the highest average accuracy of 73.79%, ST places second at 72.89%, and KM third at 71.36%. We also conduct T-tests to evaluate the significance of the performance difference between every two methods. The results show the performance ordering of CT > ST > KM is statistically significant.

We next examine the influence of liar percentage and distortion level on the performance of each method. Figures 11 and 12 display the impact of liar percentage on the performance of the various methods under two distortion levels (0.5 and 0.7 respectively). We find that CT always performs better than ST, KM, and True Tree. When the level of liar percentage is below 0.2, both CT and ST have the same accuracy. And when the liar percentage is as high as 1, both CT and ST also have the same accuracy since they provide the same recommendation as Liar Tree. The surprising result is that the True Tree can outperform the KM tree when liar percentage is not high. The advantage of True Tree over KM is more significant when distortion level is low. This is primarily because when both liar percentage and distortion level are low, most consumers are truth-tellers. But KM implicitly assumes that all users have the same tendency to lie about the required variables.

Figures 13 and 14 show the impact of input distortion on the performance of the various methods under two liar percentages (0.5 and 0.7 respectively). We again find that CT performs the best among all methods. Specifically, we can see that when distortion level is either as low as 0.1 or as high as 0.9, the performance difference between CT and ST narrows. But different from the results in Figures 11 and 12, True Tree has a general advantage to KM in any distortion level when liar percentage is even as high as 70%, which indicates that if there is a clear separation of liars and truth-tellers in the user population, the KM method is not ideal. Further, in Figures 11 and 12, the performance of True tree has a linear relationship with the liar percentage. But in Figures 13 and 14, the accuracy of True tree does not decrease so drastically as the distortion level increases. This indicates that liar percentage may have a higher influence on the performance difference compared to distortion level. The performance differences between methods are confirmed by t-tests with a p-value at 0.001.

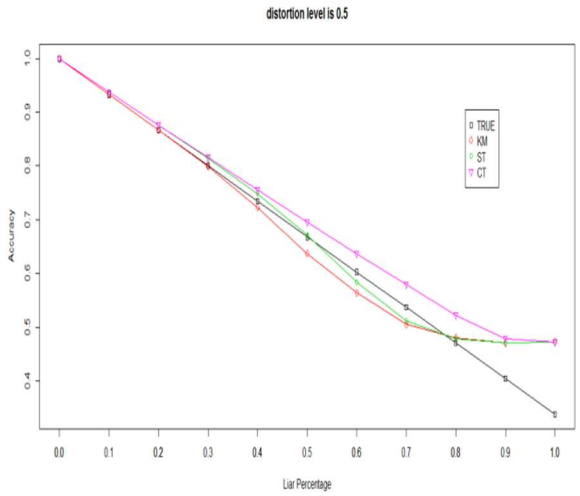

Figure 11 Impact of Liar Percentage on Accuracy (Distortion Level is 0.5)

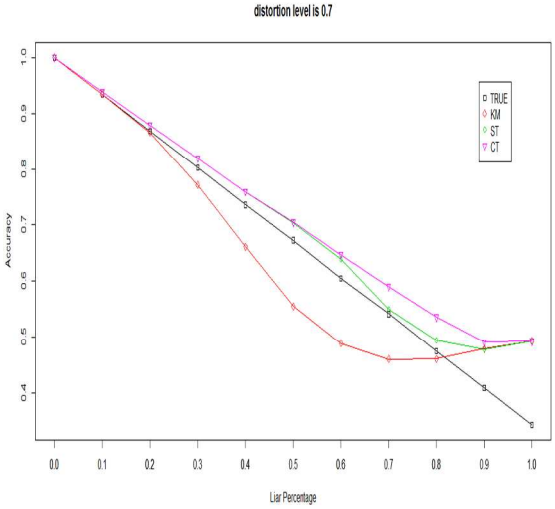

Figure 12 Impact of Liar Percentage on Accuracy (Distortion Level is 0.7)

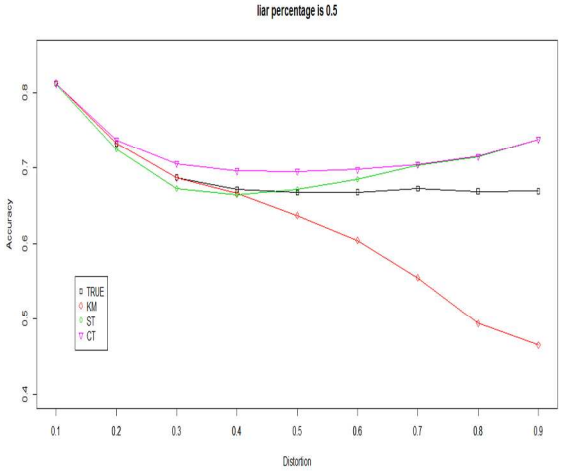

Figure 13 Impact of Distortion on Accuracy (Liar Percentage is 0.5)

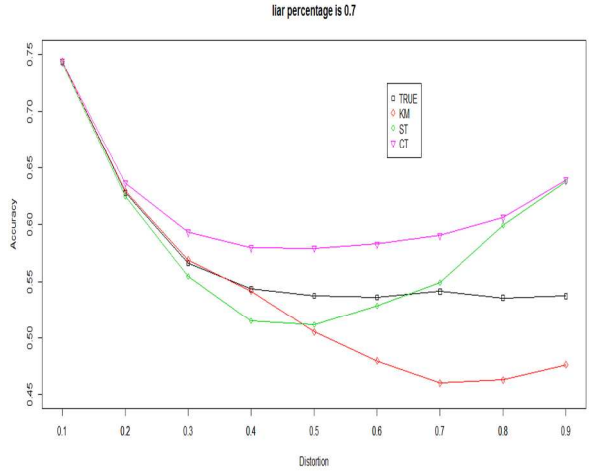

Figure 14 Impact of Distortion on Accuracy (Liar Percentage is 0.7)

We further test the robustness of the methods based on real world data. We adopt a credit card application approval dataset obtained from the UC Irvine machine learning repository (http://www.ics.uci.edu/~mlearn/MLRepository.html). Similar to the process used by Jiang et al. (2005), we first preprocess the dataset to obtain a fully enumerated true decision table for truth tellers, which includes 15 binary variables. The changes of liar percentage and distortion level are the same as what we have done in the experiments with simulated decision trees. The experimental results are similar to those obtained from the simulated experiments. We find that CT always performs better than ST, and ST, in turn, generally outperforms the True tree. True tree mostly outperforms KM when the liar percentage is low. The performance differences between the compared methods are confirmed by t-tests with a p-value

at 0.001. The influence from distortion level or liar percentage is similar to what we have shown in Figures 11~14.

Since KM is the best method proposed by Jiang et al. (2005), the results confirms the benefit of differentiating liars from truth-tellers in designing more intelligent expert systems. Between ST and CT, the latter is not only more efficient, but also more accurate. Therefore, the CT method should be the method of choice for addressing the challenge of input distortion for deductive expert systems.
We are in the process of expanding our experiments to include the two value based methods. Based on their setup, we expect that the value-based methods will lead to lower costs compared to their accuracy-based counterparts.

5. Discussion and Future Research

Despite the prevalence of input distortion by users of expert systems, limited research has been conducted to effectively address them. The methods we propose in this study attempt to differentiate liars from truth-tellers and treat them differently when their information is provided to a deductive expert system. In addition, two of the methods we propose also take into consideration asymmetric misclassification cost, which allows the method to be applicable to more problem domains. The methods we propose in this research can lead to better accuracy or lower cost. Given the wide application of expert systems in various problem domains, these methods can lead to significant financial saving for firms.

We are in the process of conducting experiments to further evaluate and compare the performances of the various methods. Among others, we plan to evaluate how the characteristic of the distortion matrices and misclassification matrices affect the performance of the proposed methods. We have also tried to use multiple verifiable variables. As expected, including multiple verifiable variables further improves the performance of the proposed methods. However, verifying multiple variables will increase the verification costs. We will explore this tradeoff in our further research. Future studies could also examine criteria for selecting the best VAs and the cost and benefits of using more than one VAs when implementing the proposed methods.

References

Biros, D. P., J. F. George, et al. (2002). "Inducing sensitivity to deception in order to improve decision making performance: A field study." *MIS Quarterly* ***26****(2):* 119-144.

Boylu, F. Aytug, H. & Koehler, G, J. (2010). Induction over strategic agents. *Information Systems Research, 21,1,*170-189.

Brodley, C. & Friedl, M. (1999). Identifying mislabeled training data. *Journal of Artificial Intelligence Research, 11,*131-167.

Buller. D. B and Burgoon. J. K. 1996. "Interpersonal deception theory," *Communication Theory* (6:3), pp. 203–242.

Duan, Y., Edwards, J. S., & Xu, M. X. (2005). Web-based expert systems: benefits and challenges. *Information & Management, 42,* 799-811.

Fellegi, I. & Holt, D. (1976). A systematic approach to automatic edit and imputation. *Journal of the American Statistical Association, 71,* 17-35.

George, J. F., D. P. Biros, et al. (2004). "Testing various modes of computer-based training for deception detection." *Intelligence and Security Informatics:* 411-417.

Hoffman, D. L. Novak, T. P. & Peralta, M. (1999). Building con trust online. *Communications of the ACM, 42,4,* 80-85.

Jiang, Z. Mookerjee, V. S. & Sarkar, S. (2005). Lying on the web: Implications for expert systems redesign. *Information Systems Research, 16, 2,* 131-148.

Liao, H. (2005). Expert system methodologies and applications-a decade review from 1995 to 2004. *Expert Systems with Applications, 28,* 93-103.

Mookerjee, V. S., & Dos Santos, B. L. (1993). Inductive expert systems design: Maximizing System Value. *Information Systems Research, 4:2,* 111-140.

Mookerjee, V. S., Mannino, M. V. & Gilson, R. (1995). Improving the performance stability of inductive expert systems under input noise. *Information Systems Research. 6,4,* 328-356.

Power, D. J. (2000). Web-based and model-driven decision support systems: concepts and issues,

Proceedings' of the Americas Conference on Information Systems, Long Beach, CA.
Quinlan, J. R. (1986). The effect of noise on concept learning. Machine *Learning, 2,* 149-166. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.) Morgan Kaufmann, Los Altos, CA..
Rubin, D. (2004). Multiple imputations for nonresponse in surveys. New York: Wiley.
Santos, E., & Johnson, G. (2004). Toward detecting deception in intelligent systems. *Proc. SPIE Defense & Security Symposium, 5423.*
Turban, E., & Aronson, J. E. (2000). *Decision Support Systems and Intelligent Systems* (6th ed). Prentice International Hall.
Worsham, R. (2010). Penalties for a Falsified Credit Application. Available at http://www.ehow.com/Flist _6797816_penalties-falsified-credit-application.html.
Wu, X. & Zhu, X. (2008). Mining with noise knowledge: error-aware data mining. *Systems and Humans, 38,4,* 917-932.
Zhang, Y. & Wu, X. (2010). Integrating induction and deduction for noisy data mining. *Information Sciences, 180,* 2663-2673.
Zhou, L., Twitchell, D., Qin, T., Burgoon, J. and Nunamaker, J. 2003. "An exploratory study into deception detection in text-based computer-mediated communication," in *Proceedings of Thirty-Sixth Hawaii International Conference on System Sciences.*
Zhou, L., Shi, Y., and Zhang, D. 2008. "A Statistical Language Modeling Approach to Online Deception Detection," *IEEE Transactions on Knowledge and Data Engineering (*20:8), pp. 1077-1081.
Zhou, L. and Zhang.D. 2008. "Following Linguistic Footprints: Automatic Deception Detection in Online Communication," *Communications of the ACM* (September), pp. 19-22.
Zhu, X. Wu, X. and Chen, Q. 2003. "Eliminating class noise in large datasets.*Proc," in International Conference on Machine Learning,* pp. 920-927.
Zhu, X. Wu, X. and Yang, Y. 2004. "Error detection and impact-sensitive instant ranking in noisy datasets," in *Proc. Association for the Advancement of Artificial Intelligence,* pp. 378-384.