The impact of measurement error on prediction rule ensembles for classification

Brillaki, E.

The Impact of Measurement Error on Prediction Rule Ensembles for Classification

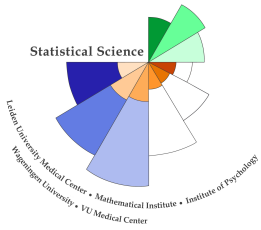
Evangelia Brillaki (s1726625)

Thesis advisor: Prof. Dr. Mark de Rooij

Second advisor: Dr. M. Fokkema

MASTER THESIS

Defended on September 10, 2019



STATISTICAL SCIENCE FOR THE LIFE AND BEHAVIOURAL SCIENCES

Abstract

Prediction rule ensembles (PREs) aim to offer a good compromise between prediction accuracy and interpretability by selecting a small set of the most important prediction rules. The accuracy of tree-based methods, such as single decision trees are known to be negatively affected by measurement error. The PRE algorithm is based on single decision trees, which are turned into an ensemble of multiple rules and may thus inherit the negative effect of measurement error. However, an extensive investigation of the influence of measurement error on the performance of PREs has not been conducted before. Therefore, we evaluated the impact of measurement error on the performance of PREs though two simulation studies: one for data with continuous predictor variables and the other for data with binary predictor variables. In both the focus is solely on binary classification.

We found that the predictive accuracy of PREs, as measured by AUC values, deteriorated in the presence of measurement error. More importantly, it was found that the performance of the PRE method deteriorated with larger amounts of measurement error for both the binary and continuous predictor scenarios. In addition, the performance of PREs in terms of number of correctly selected rules, type I and type II errors was evaluated. We found that, apart from deteriorating the predictive performance of the PREs, measurement error can also deteriorate

the interpretability of the fitted ensemble by selecting wrong rules, resulting in unreliable and wrong conclusions.

Keywords: RuleFit, prediction rule ensembles, measurement error, classification error, reliability, type I error, type II error.

Acknowledgments

I would like to thank my first supervisor Prof. Dr. Mark de Rooij for his thorough guidance and extraordinary support during this project. Especially for giving feedback to my work and steering me in the right direction whenever I needed it. Also, I would like to express my appreciation to my second supervisor Dr. Marjolein Fokkema for her extremely helpful additional suggestions for the final version of my thesis.

Contents

Chapter 1

Introduction

Ensemble learning methods, which are frequently used in data mining, machine learning and pattern recognition, are supervised learning algorithms that combine the predictions of multiple weak learning methods to produce a final strong ensemble learner (Rokach, 2010). Although, ensemble learning models can achieve high prediction accuracy, they are often difficult to interpret because of their "black box" nature, which stems from an ensemble learning model being compiled by a committee of base learners, e.g., tree-based models (Shimokawa et al., 2014; Strobl et al., 2009). While a single decision tree can provide an intuitive and interpretable model, an ensemble of decision trees can no longer be grasped.

In certain applications interpretability can be the main focus. In scientific disciplines such as psychometrics, binary responses are usually analyzed by logistic regression to construct a high-precision prediction model (Shimokawa et al., 2014). However, as Fokkema et al. (2015) mentioned in their paper, a simple linear relationship may not adequately resemble the reasoning process of human decision makers in clinical practice. Moreover, many authors (e.g., Dhami, 2003; Green &

Mehr, 1997; Gigerenzer & Goldstein, 1996) have found the weighing of explanatory variables (cues) in human judgment, to be non-linearly related to the outcome in logistic regression, i.e, the logarithmic odds ratio. They also support that psychologists in particular are likely to make decisions by weighting the values of a smaller number of variables than the number of variables included in a logistic regression model. Also, the weights human decision makers assign to cues may be dependent on other cue values. Zeileis et al. (2008) mentioned in their paper that fitted decision trees are interpretable models, enhanced by the ability to visualize them. Furthermore, decision trees can flexibly accommodate non-linear relationships and interactions. At the same time ensemble methods, which construct more than one decision tree, are often found to outperform trees in purely predictive settings (Meyer et al., 2003). However, a method that provides both predictive accuracy of ensemble methods, but at the same time provides easy-to-interpret results would be desirable.

Decision trees belong to a class of non-parametric predictive models in supervised learning used for regression and classification. One of the most popular algorithms for decision trees is CART (Breiman et al., 1984). CART performs an exhaustive search over all possible variables and all of their possible split points, maximizing an information measure of node impurity and selecting the variable showing the best split. The splitting continues recursively in each node until some stop condition is reached. The result is a decision tree, consisting of branches and nodes, starting at a single root node and ending in the terminal nodes of the tree. Decision trees are easy to interpret, but they have two fundamental problems; biased variable selection and instability (e.g., Hastie et al., 2009; Strobl, 2008).

Although, the variable selection bias problem has been addressed by the condi-

tional inference trees suggested by Hothorn et al. (2006), the problem of instability is common for all decision tree algorithms. The source of instability is that a small change in the data can result in a very different series of splits, and hence construct a quite different tree (Hastie et al., 2009; Fokkema, 2017). This problem is mitigated through tree ensemble methods, by ensembling the predictions of many (slightly) different trees. Tree ensembles deliver superior prediction powery (e.g., Breiman, 1996; Strobl et al., 2009), but are much more difficult to interpret.

Prediction rule ensembles (PREs) aim to strike a balance between the accuracy of ensembles and the interpretability of single trees. The ensemble members in PREs have a much simpler structure than those of other ensemble methods (e.g., random forests) by deriving a small set of prediction rules from the branches of decision trees. The resulting model is highly interpretable, because the decision rules have a format that is easy to understand, but is still a flexible enough approach to capture interactions and obtain a good fit (Fokkema et al., 2015).

Several algorithms for deriving PREs have been developed, such as the RuleFit algorithm of Friedman & Popescu (2008). RuleFit derives an ensemble of simple prediction rules in two stages; it generates a large initial ensemble of decision rules from a boosted tree ensemble and second, it selects a sparse final rule ensemble using lasso regression (Tibshirani, 1996). However, the PREs of Fokkema (2017) was opted as an alternative package for fitting rule-based ensembles, and it is an improved version of the RuleFit method. There are several differences between the original RuleFit implementation and `pre`, the most important one being that `pre` employs unbiased recursive partitioning methods for rule induction while RuleFit grows CART trees. Fokkema (2017) also showed that PREs derives rule ensembles with predictive accuracy similar to that of random forests while using a smaller

number of variables for prediction. PREs can be used for regression and classification tasks. In this thesis, we focus on classification of binary responses.

It has been acknowledged in some papers that tree-based methods are sensitive to measurement error, and tree-based algorithms for data with measurement error have been developed (e.g., Tsang et al., 2009; Qin et al., 2009; Sexton & Laake, 2007). Tsang et al. (2009) extended traditional decision tree classifiers to handle predictor variables with uncertain information, such as measurement/quantization errors. Qin et al. (2009) developed a new rule-based classification and prediction algorithm called uRule for classifying and predicting based on both certain and uncertain data. Sexton & Laake (2007) extended boosted regression trees when predictor variables are measured with error.

According to Buonaccorsi (2010) measurement error occurs whenever we cannot exactly observe the value of one or more of the variables that enter into a model of interest and we can only observe an error-contaminated version instead. There are many factors contributing to such errors during the data collection process, the most common ones being errors in the instrument of measurement, in the measuring process, and in the skill of the investigator. The presence of measurement error can deteriorate the accuracy of the fitted model and its predictions. The measurement error problem typically involves specifying a model for the true values and the measurement errors, and how they are associated with the observed values. For notation purposes, we will denote the true value by X and denote the variable observed in place of X by W, which is also called the observed or measured value.

A fundamental issue in specifying measurement error in a continuous or measured variable is whether we make an assumption about the distribution of the observed values given the true values, which is specified by the classical measure-

ment error model (Carroll, 2005). In our study, the X is determined directly, and hence the classical measurement error model is used for the measurement error process, in which the true value is measured with additive error and with constant variance.

Measurement error can also arise in binary data, where it often referred to as classification error (Aigner, 1973; Savoca, 2000). For example, tumors are typically classified as benign or malignant, but can sometimes be misclassified, which leads to false positives or false negatives. The aforementioned measurement error models were developed to account regression estimates when binary or continuous variables are measured with error.

The purpose of this thesis is to evaluate the impact of measurement error on the performance of the PRE method of Fokkema (2017) in classification problems. The main aim is to examine whether the predictive performance of the method deteriorates when introducing measurement error in one of the predictor variables in a dataset. Moreover, whether this method applied on a dataset derives the same set of rules used for the data generation. We formulate the following research questions:

- How close is the accuracy of a derived PRE model to the accuracy of an unpenalized logistic regression model fitted with the true rules as predictors (the "oracle")?

- How does the performance of the PRE method change with different amounts of measurement error in a dataset? How do data characteristics, such as sample size, underlying distribution of continuous and discrete features, and proportion of class outcome labels affect the predictive performance of the method for different amounts of measurement error?

The outline of the thesis is as follows: In the Methods (Chapter 2), we present the PRE method, as well as the measurement error models for both continuous and binary variables. Those are incorporated in a simulation study in order to try to answer the research questions within the limits of a research study. The design of this simulation study is also described in this chapter. In Chapter 3, we present the results of the simulation separately for the continuous and binary predictor variable case. In the Discussion (Chapter 4), the results will be summarized and interpreted, strengths and limitations of the current study will be discussed and suggestions for future research will be provided.

Chapter 2

Methods

To gain an understanding of how the PRE algorithm performs with and without measurement error, a simulation study is employed. First, the methods used in this simulation study are described in detail, that is the PRE algorithm (Section 2.1) as first developed by Friedman & Popescu (2008), and later by Fokkema (2017). Secondly, the measurement error models for continuous predictors (Section 2.2) or binary predictors (Section 2.3). Then, the simulation set-up is presented (Section 2.4), and the measures of model performance are defined in Section 2.4.3.

2.1 Prediction rule ensemble algorithm

The R package `pre` (Fokkema, 2017) is used to derive prediction rule ensembles. In general, it derives a sparse ensemble of rules and/or linear functions for prediction of binary, multinomial, continuous and count outcome variables. The predictor variables may be numeric, ordinal and nominal. The package provides R-based implementation of the RuleFit algorithm of Friedman & Popescu (2008). Therefore,

a deep dive into the technicalities of the RuleFit algorithm is given to transition later to the `pre` implementation.

The RuleFit algorithm derives an ensemble of simple prediction rules (base learners) in two stages: first, it generates a large initial ensemble of decision rules, and second, it estimates the weight coefficients for the prediction rules in the final ensemble using lasso regression.

In the first stage, RuleFit draws a large number of subsamples of predetermined size from the training dataset, and grows a CART decision tree on each of the subsamples. Subsamples can be drawn with replacement (bootstrap sampling) or without replacement (subsampling).

To apply boosting, the learning rate of the ensemble can be controlled by setting a shrinkage parameter. This parameter determines the weight given to previously induced ensemble members (i.e., decision trees), when learning new ensemble members. Friedman & Popescu (2003) found a shrinkage parameter value of $\nu = .01$ to provide the best results ($\nu = 0$ indicates no dependency between previously induced ensemble members, and $\nu = 1$ maximizes dependency on previous ensemble members).

Each resulting CART tree is turned into multiple rules by turning each path to a node in a tree into a decision rule. Rule ensembles use the resulting rules and/or linear models as base learners to construct an ensemble. This study focuses on tree-based ensembles with only rules as base learners, and hence no description of including linear terms to complement a rule ensemble is provided.

The final rule ensemble has the general ensemble model form:

$$F(x) = \alpha_0 + \sum_{m=1}^{M} \alpha_m r_m(x), \tag{2.1}$$

where M is the size of the ensemble, i.e., the number of rules, $r_m(x)$ is a rule, and $F(x)$ is the ensemble predictor consisting of a linear combination of the predictions of the rule ensemble members, with $\{\alpha_m\}_{m=0}^{M}$ being the corresponding parameters specifying the particular linear combination[1].

Decision rules that the algorithm generates are binary features that have a simple form: a value of 1 when all the conditions of the rule are met, otherwise the value is 0 when any of the conditions of the rule is not met. Each rule term r_m takes the form:

$$r_m(x) = \prod_{p=1}^{P} I(x_p \in s_{pm}), \tag{2.2}$$

where P is the number of input features used in the m-th rule and $I(\cdot)$ is the indicator function, which is 1 when the input feature x_p is within its specified subset of values s_{pm}, otherwise is 0.

For continuous or ordered features, s_{pm} is an interval in the value range of the feature, i.e, the interval $s_{pm} = (x_{pm}^{-}, x_{pm}^{+}]$, where x_{pm}^{-} and x_{pm}^{+} are the lower and upper limits of the rule, respectively. If x_p is a categorical variable, s_{pm} is a subset of the categories of the feature.

To illustrate this rule generation, Figure 2.1 depicts an example tree defined by three variables with four terminal nodes, and the following set of rules can be

[1]The notation $\{\alpha_m\}_{m=0}^{M}$ is similar to α_m with $m = 0, \ldots, M$.

derived:

$$r_1(x) = I(x_4 \leq 34.5)$$

$$r_2(x) = I(x_4 \leq 34.5) \cdot I(x_2 \in \{B, D, E\})$$

$$r_3(x) = I(x_4 \leq 34.5) \cdot I(x_2 \in \{A, C\})$$

$$r_4(x) = I(x_4 > 34.5)$$

$$r_5(x) = I(x_4 > 34.5) \cdot I(x_5 \leq Rarely)$$

$$r_6(x) = I(x_4 > 34.5) \cdot I(x_5 > Rarely)$$

All nodes without the root node in the example tree produce a rule of the form (2.2). This also results in redundant rules, such as, the r_1 and r_4 rule which are perfectly collinear: $r_4(x) = 1 - r_1(x)$. Therefore r_4 will be omitted from the initial ensemble. Similarly, rules that are identical to earlier generated rules are also removed from the initial ensemble. Furthermore, input variables may be continuous (like x_4), unordered categorical (like x_2) or ordered categorical (like x_5) using a Likert scale example; {Never, Rarely, Sometimes, Very Often, Always}.

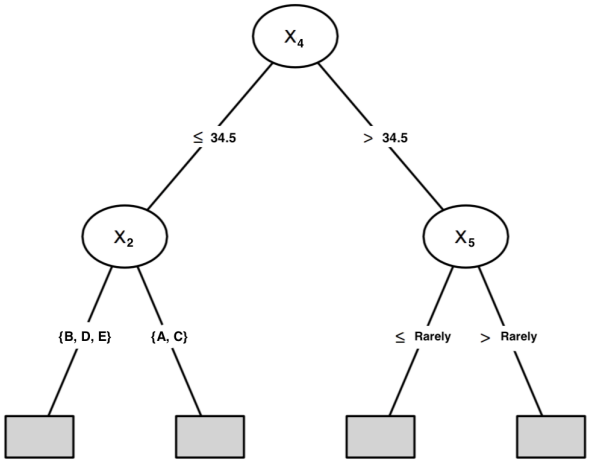


Figure 2.1: An example tree to illustrate the rule generation. A rule is generated for every node in the tree, except the root, and only the tree structure is used for generating rules.

In total,

$$M = \sum_{k=1}^{K} 2(t_k - 1), \tag{2.3}$$

rules are derived from all trees $\{T_k\}_{k=1}^{K}$, with t_k terminal nodes in the k-th tree.

In the second stage, RuleFit creates a final ensemble of prediction functions by applying lasso regression with all the decision rules, and every rule receives a weight estimate. The lasso penalty is selective, hence many of the less influential input rules may have zero regression coefficient, and be excluded from the final predictive model. This step resembles pruning in classification and regression trees (Breiman et al., 1984).

Friedman and Popescu (2008) estimated the parameters $\{\alpha_m\}_{m=0}^{M}$ by the lasso method as follows:

$$\{\hat{\alpha}_m\}_{m=0}^{M} = \underset{\{\alpha_m\}_{m=0}^{M}}{\operatorname{argmin}} \sum_{n=1}^{N} L\left(y_n, \alpha_0 + \sum_{m=1}^{M} \alpha_m r_m(\mathbf{x}_n)\right) + \lambda \left(\sum_{m=1}^{M} |\alpha_m|\right),$$

where $\lambda \geq 0$ is the lasso penalty, $L(\cdot)$ is a loss function, y_n is the outcome variable with $n = 1, \ldots, N$ and $\mathbf{x}_n$ is a vector of the predictor variables for the n-th observation. The optimal lasso penalty $\hat{\lambda}$ is estimated by (multi-fold) cross-validation. For the choice of a loss function $L(y, F)$, Friedman & Popescu (2003) present a variety of different loss criteria that are appropriate in different settings.

In this thesis, we employ the `pre` package that implements the algorithm of Friedman & Popescu (2008), with some improvements and adjustments. The ones most relevant for our study are as follows. First, conditional inference trees of Hothorn et al. (2006) is used for deriving prediction rules, instead of CART trees.

This non-parametric model selects a variable in each node by permutation-based significance tests, which avoids the variable selection bias towards variables with many possible splits or with many missing values. Second, rules from perfectly correlated pairs of rules are removed from the initial ensemble, as this does not affect predictive accuracy, but does improve the sparsity of the final ensemble. Third, the `pre` package is completely R based, allowing better access to the results and more control over the parameters used for generating the PREs, but it is computationally inferior to the RuleFit program.

Related to the two stages of the RuleFit algorithm that were described, the function `pre` from the `pre` package by default employs the function `ctree` from the `partykit` package (Hothorn & Zeileis, 2015), and the `cv.glmnet` function from the `glmnet` package (Friedman et al., 2010). For our classification purposes a classification conditional inference tree is fitted by `ctree`. For the second stage of weight estimation, the function `cv.glmnet` fits a generalized linear model via penalized maximum likelihood. A sequence of models is fitted given a grid of values for the regularization parameter λ through coordinate descent. For classification tasks, the loss function is constructed from the deviance residuals by default within the framework of the logistic regression model.

2.2 Measurement error model for quantitative variables

For a quantitative variable, the classical measurement error model has the form $W = X + U$, where the observed W is an unbiased measure of the true X, and the

measurement error U is assumed independent of the explanatory variable X. The error structure of U could be homescedastic (constant variance) or heteroscedastic, and for the former it is assumed approximately normal with constant variance, i.e., $U \sim N(0, \sigma_u^2)$ (Carroll et al., 2006).

In the field of Psychometrics, Classical test theory (CTT) which predicts outcomes of psychological testing, uses the classical measurement error model for the true and observed scores. That is because test users never observe a person's true score, only an observed score. Classical test theory is concerned with the relations between the three variables W, X, and U in the population. These relations are closely associated with the quality of test scores, and that quality can be assessed through the concept of reliability, i.e., the degree of consistency of the measure. The reliability of the observed variable W, which is denoted as ρ, is defined as the proportion of true variance σ_X^2 over the observed variance σ_W^2:

$$\rho = \frac{\sigma_X^2}{\sigma_W^2} = \frac{\sigma_X^2}{\sigma_X^2 + \sigma_U^2}. \tag{2.4}$$

The last equality holds because U and X are independent, and hence the obtained variance is equal to the sum of the true variance and the error variance. From this signal-to-noise ratio equation, it is intuitive to see that the reliability coefficient of test scores becomes higher as the proportion of error variance in the test scores becomes lower and vice versa. The reliability is equal to the proportion of the variance in the test scores that we could explain if we knew the true scores. More about the reliability coefficient can be found in the book of McDonald (1999).

2.3 Measurement error model for binary variables

As illustrated first in the paper of Aigner (1973) and later in Savoca (2000), starting from the single variable case, the binary variable W is observed which is the true predictor X measured with error U according to the relationship $W = X + U$. Let's suppose that X indicates the presence or absence of a disease, and W is the diagnosis according to the survey instrument. Let the following quantities be defined:

- P represent the proportion of people in a population of N who truly have the disease.

- $\tilde{P}$ is the proportion of people diagnosed with the disease according to the survey, then $\tilde{Q} = 1 - \tilde{P}$ is the proportion of those diagnosed as healthy.

- η is the proportion of people who truly have the disease but are classified as not having the disease, out of the $N\tilde{Q}$ number of persons diagnosed negative (false negative rate).

- ν is the proportion of people who are truly healthy but who are classified as having the disease (false positive rate) of which there are $N\tilde{P}$. Then in the total population, $N\tilde{P} - \nu N\tilde{P} + \eta\tilde{Q}N$ actually have the disease.

The errors-in-variables framework is then $P = (1-\nu)\tilde{P} + \eta\tilde{Q}$ in order to allow misclassification into both directions. From the set-up the marginal distributions of X and W are Bernoulli with parameter P and $\tilde{P}$, respectively.

Unlike the classical measurement error model, the measurement error U does not have zero mean, unless $E(W) = E(X) = P$ which would mean that there is no misclassification in the diagnosis. Moreover, the measurement error U is

negatively correlated with the true diagnosis X.

2.4 Simulation setup

In the following paragraphs, the simulation setup is presented. In Section 2.4.1, the data generation was described, and in Section 2.4.2, how we included measurement error to one of the predictors of the generated datasets for the continuous and binary case. Then, the models that were fitted and analysed on those datasets are mentioned in Section 2.4.3.

During the data generation phase, four design factors were opted: training set size, underlying distribution of continuous only features (data type), proportion of class outcome labels, and amount of measurement error in one of the features. Moreover, two different simulation studies were considered because of the different framework in applying measurement error in a continuous and binary predictor variable, hence making the effect of measurement error not comparable for those two settings.

2.4.1 True underlying model for data generation

The data was composed of ten predictor variables $\{X_p\}_{p=1}^{10}$ simulated by a normal distribution in the continuous case, or a binomial distribution in the binary case. Sample sizes of $N = 200$, 500 and 1000 were chosen for data with continuous predictors and $N = 200$, 500, 1000, 1500 and 2000 for data with binary predictors. Larger sample sizes were explored in the binary case as the algorithm has to consider only a single binary split point for every possible predictor variable, which makes the procedure faster.

Generating continuous predictor variables

Each continuous predictor follows a normal distribution $\{X_p\}_1^{10} \sim N(\mu_p, \sigma_p^2)$, with mean μ_p randomly chosen from 10 to 200, i.e., $\mu = (19, 89.6, 62.2, 173, 166.4, 107.7, 152.6, 48.6, 1.7, 33.6)^\mathsf{T}$, and for the standard deviation σ_p^2 the following data setting options were explored:

I. common standard deviation $\sigma_p = 9.2$ for all the predictors which was randomly chosen from 5 to 20, hence the predictors are independent and identically distributed.

II. each predictor has different standard deviation σ_p randomly chosen from .2 to 10, i.e., $\sigma = (2.9, 8.7, 1.9, 5.3, 3.6, 9.8, 7.2, 10, 2.4, 8)^\mathsf{T}$ was always specified.

III. the vector of the predictors follows a multivariate normal distribution, with σ^2 be a 10 by 10 covariance matrix. The covariance for each pair of variables was set at 0.5 and the diagonal values of the covariance matrix (i.e., the variances) was set at 1.

Then, the four rules were defined (as in 2.2), with a different set of split points for each choice of σ^2 and sample size N:

$$r_1 = I(X_6 > s_{l1}) \cdot I(X_8 > s_{l2})$$

$$r_2 = I(X_5 \leq s_{l3}) \cdot I(X_7 \leq s_{l4})$$

$$r_3 = I(X_3 > s_{l5}) \cdot I(X_{10} > s_{l6})$$

$$r_4 = I(X_6 \leq s_{l7}) \cdot I(X_9 \leq s_{l8})$$

with $l = 1, \ldots, 9$ (3 sample sizes, 3 settings for each sample size). The split point of a predictor variable was randomly selected from any value between the 25% and

75% quantile of that predictor. Three different sets of split points were selected for each combination of sample size and data setting. We assume in each data setting that roughly the same split points will be generated for all the sample sizes as those data follow the same distribution.

Generating binary predictor variables

In the binary case, a fourth data setting (IV) is specified. Each predictor follows a binomial distribution $\{X_j\}_1^{10} \sim Bin(N, p_j)$, always with probability $p = (.22, .45, .36, .69, .67, .48, .61, .31, .79, .26)^\mathsf{T}$ which was randomly chosen from .2 to .8. The four rules were defined (as in 2.2) as follows:

$$r_1 = I(X_6 = 0) \cdot I(X_8 = 0)$$

$$r_2 = I(X_5 = 1) \cdot I(X_7 = 1)$$

$$r_3 = I(X_3 = 0) \cdot I(X_{10} = 1)$$

$$r_4 = I(X_6 = 1) \cdot I(X_9 = 0)$$

Generating class variable

In each of the four data settings, four different rules $\{r_m\}_1^4$ were defined, all formed by two features (nodes), and consequently five alpha parameters were selected to form the linear predictor (2.1) as $F(x) = \alpha_0 + \sum_{m=1}^4 \alpha_m r_m(x)$.

The values $\alpha = (\alpha_0, 9.28, 6.85, -4.62, 15.27)^\mathsf{T}$ were opted to be used for each simulation setting, which were randomly drawn from a uniform distribution, i.e. $\alpha_m \sim U(a = -25, b = 25)$ with $m = 1, \ldots, 4$. The intercept α_0 received a different

value for a given data type in order to construct data with class proportion labels that can be either balanced or unbalanced. To yield the balanced class proportion $P(Y = 1) = .5$, the thresholds displayed in the left part of Table 2.1 were used as an intercept, whereas the right part lists the thresholds used to form the unbalanced class proportion $P(Y = 1) = .8$ for each combination of sample size and data type.

Table 2.1: The intercept of the linear predictor as it was defined for each combination of sample size, data type (1-4) and proportion of class outcome labels.

N	$P(Y=1)=.51$				$P(Y=1)=.81$			
	1	2	3	4	1	2	3	4
200	-1.60	-3.80	-14.50	-6.80	1.40	0.78	-4.70	0.00
500	-5.50	-10.00	-9.00	-6.80	0.80	-1.50	1.00	0.00
1000	-1.50	-5.50	-9.20	-6.80	1.72	0.62	-2.50	0.00
1500				-6.80				0.00
2000				-6.80				0.00

For generating the outcome variable a logit function is used with $F(x_i) = \log \frac{p_i}{1+p_i}$ and $p_i = P(Y_i = 1)$. The dependent class variable, $\{Y_i\}_1^N$ taking the values 0 or 1, is defined using a binomial distribution, i.e

$$Y_i \sim Bin(1, p_i) \equiv Bin\left(1, \frac{e^{F(x_i)}}{1 + e^{F(x_i)}}\right)$$

with $i = 1, \ldots, N$. The p_i is obtained by the inverse-logit function that converts the linear predictor (the log-odds) to probabilities.

2.4.2 Incorporating measurement error

In the previous Section 2.4.1, we described how we generated a dataset without adding measurement error. Here, we describe how we included measurement error to one of the predictors for the continuous and binary case. The predictor variable

X_6 was chosen for this purpose, which is contained in two out of the four true rules. The dependent variable Y was generated as described before, using the true predictor variable. Hence, the same dataset was generated as the one with no measurement error, except the column that corresponds to the predictor X_6 is changed. The following subsections describe how measurement error was added in the X_6 predictor for each study.

Incorporating measurement error on continuous predictors

The predictor X_6 follows a normal distibution, i.e, $X_6 \sim N(0, \sigma^2_{X_6})$, where $\sigma^2_{X_6}$ is the true standard deviation of X_6. However, instead of the true values X_6, the variable W_6 is measured. The classical measurement error model is used from Section 2.2 to model the error as follows:

$$W_6 = X_6 + U_6$$

with error structure $U_6 \sim N(0, \sigma_{u_6})$.

The reliability coefficient ρ (2.4) was employed as a measure to insert measument error into the predictor X_6 and control the amount of the applied measument error though that coefficient.

The reliability coefficient ρ of the predictor W_6 is given by $\rho = \frac{\sigma^2_{X_6}}{\sigma^2_{W_6}} = \frac{\sigma^2_{X_6}}{\sigma^2_{X_6}+\sigma^2_{U_6}}$. Thus, σ_{U_6} can be obtained by $\sigma_{U_6} = \sigma_{X_6}\sqrt{\frac{(1-\rho)}{\rho}}$ with ρ taking three values, i.e., $\rho = .9, .7, .5$, indicating high, moderate and low reliability.

Incorporating measurement error on binary predictors

For the binary setting, we allow misclassification in the binary predictor X_6 using the errors-in-variables framework described in Section 2.3.

Here, the true predictor X_6 follows a binomial distribution, i.e., $X_6 \sim Bin(N, p_6)$, where p_6 is the true probability of X_6. Then, we generate some error to be used in constructing the observed variable W_6, i.e.,

$$error = \begin{cases} 1, & \text{if } e < -z \text{ or } e > z \\ 0, & \text{otherwise,} \end{cases}$$

where $e \sim N(0, 1)$. The parameter z indicates the value of a standard normal distribution in order to find the probability (area) of the specified range, i.e., outside $-z$ and z, of the distribution. Two values of z were considered, namely $z = \{1.5, 1.15\}$, corresponding to a (theoretical) probability of 13% and 25% respectively. These percentages give the expected misclassification rate (the percentage of classifications that were incorrect) in the mismeasured binary predictor. The observed W_6, which is the true predictor measured with error, is generated as follows:

$$W_6 = \begin{cases} 1, & \text{if } X_6 = 0 \text{ and error} = 1 \\ 0, & \text{if } X_6 = 1 \text{ and error} = 1 \\ X_6, & \text{otherwise.} \end{cases}$$

2.4.3 Specification of models and design

In the continuous case, 3 sample sizes were specified along with 3 different data types, 2 proportions of class outcome labels and 4 measurement error sizes (including no measurement error). For the data with binary predictors, 5 sample sizes were explored, with the same 2 proportions of class outcome labels, and 3 measurement error sizes (including no measurement error).

Hence, a full factorial design resulted in $3 \times 3 \times 2 \times 4 = 72$ cells for the continuous case and $5 \times 2 \times 3 = 30$ cells for the binary case in total. In each cell, 100 datasets were generated to train the "oracle"[2] and the PRE method. In addition, a test dataset of size 200 was generated for every training dataset with similar structures, i.e, with same data type, proportions of class outcome labels and measurement error size.

"Oracle" analysis

In each design cell, a dataset is generated as described in Section 2.4.1. Using the four rules $\{r_m\}_1^4$ of that dataset as regressors to predict the binary class variable $\{Y_n\}_1^N$, we aim to obtain the true underlying accuracy and AUC. Hence, logistic regression was fitted through a 10-fold cross-validation method using the functions `train_control` and `train` from the `caret` package (Kuhn et al., 2018). The 'glm' method was specified with the binomial family.

These logistic regression models act as a benchmark (the "oracle") of how best their predictive performance can be, and they will be compared to the PRE models in the case where no measurement error was applied. In particular, the

[2] A logistic regression model given as input the four rules on which the data generation was based, instead of the predictor variables.

"oracle's" accuracy and AUC were calculated using the predictions on the test set to be compared with the same metrics of the PRE models. A binary outcome was predicted to be 1, if the resulted predicted probabilities were higher than .5, otherwise class 0 was predicted.

PRE analysis

The `pre` function was used to derive a sparse ensemble of rules to predict the binary outcome. Linear functions were not considered, as they were not part of the data generation scheme. The algorithm uses fractions of randomly selected training observations to produce each tree, and these subsamples were drawn without replacement (i.e., subsampling). The maximal depth of trees to be grown by default is 3, which is the maximum number of conditions in the rules. The default learning rate of $\nu = .01$ was specified for the sequentially induced trees. Basically using the suggestion of Friedman & Popescu (2003) that found that this shrinkage parameter value could potentially increase the performance of the method. In the ensemble, rules which have the exact same support in the training data were removed. Rules were not standardized before estimating the lasso model, and 10-fold cross-validation using the deviance as loss function was performed for determining the lasso penalty parameter. The final model was selected by using the penalty parameter criterion that gives cross-validation error that is within 1 standard error of minimum cross-validation error.

In each design cell, data was generated (described in Section 2.4.1 and 2.4.2) to be used as input in the `pre` function. After a PRE model was fitted, its derived rules were compared to the four true rules (details will follow), used to generate that data. The performance of the fitted model, i.e, accuracy and AUC, was

assessed through predictions on a test set of size 200.

Measures of model performance

For each PRE model that was fitted, its accuracy and AUC was recorded on the test set, whether the correct rules were selected, and how many rules are constructed. Apart from tracking if the derived rules are similar to the true rules, we define the type I and II errors in the following paragraphs to check overall if at least the same variables that formed the true rules are the ones that the method derived. Hence, each derived rule is expected to contain any of the seven variables that the true rules contain.

One of the performance measures is the correct classification rate (CCR). This is computed as the number of correctly classified instances divided by the total sample size.

The AUC statistic is an indication of the discriminative ability of a classification method. For binary classification this value equals the area under the receiver operating characteristic (ROC) curve, which plots sensitivity (true positive rate) versus 1-specificity (false positive rate). The AUC value of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example (with outcome $Y = 1$) higher than a randomly chosen negative example (with outcome $Y = 0$). The AUC is an appropriate evaluation metric when class imbalance is present; it does not depend on a threshold, and is therefore a better overall evaluation metric compared to accuracy (Burez & Van den Poel, 2009).

After a PRE model was fitted, its derived rules will be compared to the true rules which are considered the same, if both contain the same predictors with the same corresponding inequality symbols, and the split points in the derived and

true rule are "quite close". That is, the split points of the predictors in the true and derived rule were regarded as the same, if the derived split point is within 1 standard deviation from the true split point, with the standard deviation being obtained from the corresponding predictor.

The type I error was defined by counting, for each repetition, if the variables X_{p_1} with $p_1 \in \{1, 2, 4\}$ were included in the derived rules, whereas they are not part of the true rules. Similar to how the term is used in statistical hypothesis testing, we will have a "false positive" finding. For instance, by fitting a PRE model we would expect that the predictor variables that were included in the derived rules would be identical to the ones used to form the four true rules in Section 2.4.1. If that is the case and the model returns rules without any of the variables X_1, X_2, and X_4, the type I error of each X_{p_1} gets a 0 value for a given repetition, or if a X_{p_1} is appeared, it will receive a value of 1. In each cell formed by the design factors and after 100 repetitions, a score between 0 and 1 will be calculated by counting for each variable X_{p_1} how many times it appeared in the derived rules out of the 100 repetitions. If any of the three variables appears more than once in a derived model, it still receives a value of 1.

The type II error was calculated by counting for each repetition, if the rest of the seven variables X_{p_2} with $p_2 \in \{3, 5, 6, 7, 8, 9, 10\}$ were not included in the derived rules, whereas they are part of the true rules. Here we follow the logic of a "false negative" finding. All the variables from the derived rules should be one of those seven variables that were part of the true rules. If at any of those seven variables are not included in the derived rules of a model, it is recorded with the value 1, and after 100 repetition the type II error of each of the seven variable is calculated, i.e., the percentage that the variable X_{p_2} was not part of the derived

model. If any of those seven variables appears more than once in a derived model, it still receives a value of 1.

All the results from the simulation studies were collected in a final dataset with all the performance measures per replicate for further analysis. In this dataset, the rows were considered as the subjects, with one subject representing a single simulation dataset, and the columns were the design factors (predictor variables) and the response (performance measures). The design groups contain measurements from different set ups as a different dataset was formed per combination of factors, except for the measurement error factor that was applied on the same dataset multiple times. A repeated measures ANOVA was applied for each performance measure in order to determine whether the main effect of measurement error, and the interaction between the level of measurement error and the other design factors, were statistically significant. The within-subjects factor in the analyses was the amount of measurement error in one of the features, and the between-subjects factors were the other design factors.

Results were not reported based on p values, because they do not reveal the size of the effect and they depend upon both sample size and effect size (Sullivan & Feinn, 2012). For our ANOVA tests, *partial* η^2 is opted to determine the effect size of the design factors in influencing the changes in outcomes. This effect size index was opted over the *classical* η^2 as it stays unaffected by the inclusion of variables that introduce additional variation and it can be applied to all research designs (Richardson, 2011). Only factors with $\eta^2 \geq .06$ were considered, which indicates medium effect size according to Cohen's guidelines (Cohen, 1988). The effect size is considered large when $\eta^2 \geq .14$.

Therefore, for each performance measure we look into the *partial* η^2 of each

factor and two-way interactions between the factors, and in the next sections we report the effect size of the measurement error factor and its two-way interactions with $\eta^2 \geq .06$. In the repeated measures ANOVA concerning the number of correct rules, the specific rule that each number corresponds to was included as an extra between-subjects factor, while for type I and II error, the predictor variable that each error corresponds was included as an extra predictor.

Similarly, an additional repeated measures ANOVA was applied to assess whether differences between the AUC of the PRE models and the "oracle", with the rest of the design factors were statistically significant.

2.4.4 Software

Simulations were performed in R studio version 3.5.1. The main used R package was the `pre` package version 0.6.0 (Fokkema & Christoffersen, 2018). The R script of the main functions used for generating data and performing simulations is given in the Appendix A. Throughout the simulations random seeds were specified for reproducibility of the experiments. The ANOVA analyses were performed using the `aov` function and the *partial* η^2 was obtained from the `DescTools` package (Signorell, 2019) and the `EtaSq` function.

Chapter 3

Results of simulation study

In this section, the results of the simulation studies with the continuous and binary predictors are presented, each in its own subsection. In both cases, the most important effects contributing to differences in performances between amounts of measurement error are reported for every performance measure.

3.1 Results concerning continuous predictors

In Table 3.1, the predictive accuracies for the PRE method and the "oracle" are presented, for each cell of the data-generating design. As expected the CCR and AUC in the balanced scenario are almost the same in each cell of the design, therefore we focus on the AUC for comparisons cross the design factors from now on.

Comparing the AUC of the PRE models with that of the "oracle" when no measurement error is present, we observe that both metrics appear quite close in some instances, especially when the sample sizes are larger. The within-subject

effects that explained most of the variance of average AUC were the main effect of fitting method, the interaction between fitting method and sample size, and the interaction between fitting method and proportion of class outcome labels. The interaction between fitting method and sample size has a large effect size (*partial* $\eta^2 = .73$). This indicates that the relationship between PRE's and "oracle's" AUC depends on the sample size, which is graphically depicted in Figure 3.1(A). The contrast between PRE's and "oracle's" AUC is the highest when $N = 200$, and then the differences become smaller as the sample size increases.

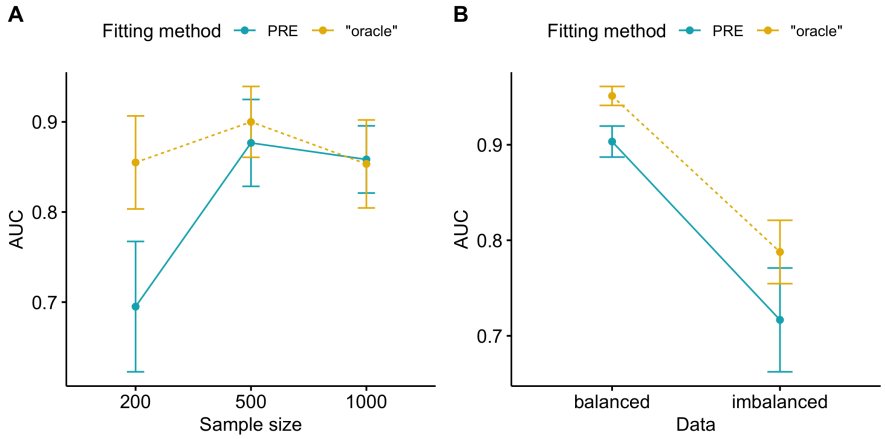
It is surprising to see that AUC increases from $N = 200$ to 500, and then goes down for $N = 1000$, both for the "oracle" and PRE models. This could be a robust finding, or it could be just sampling fluctuation (perhaps due to small test sample size). In the plot, error bars are also produced for each mean estimate (based on standard error of each mean). We see that the error bars of the PRE's AUC estimates overlap for sample size of 500 and 1000. Hence, the downward trend that we observe from $N = 500$ to 1000 could be a result of the variability of the estimates. Similarly, this could be the case also for the "oracle's" AUC.

In Figure 3.1(B), the interaction between fitting method and proportion of class outcome labels has a medium effect size (*partial* $\eta^2 = .07$), where the PRE's and "oracle's" AUC is slightly closer in the case of the balanced class outcomes.

Overall, the large main effect of fitting method (*partial* $\eta^2 = .65$) shows that the means of PRE's and "oracle's" AUC are not equal, with the overall "oracle's" AUC being higher than the AUC from the PRE models ($mean_{\text{"true" AUC}} = .87 > mean_{\text{derived AUC}} = .81$).

Moving to our second research question, we focus on how measurement error affects the AUC metric of the method. In each cell, we notice that as the value

Figure 3.1: Two-way interaction plots, which plot the mean of the AUC for two-way combinations of fitting method and sample sizes (A), or proportion of class outcome labels (B), thereby illustrating possible interactions. In plot A, the difference of PRE's and "oracle's" AUC become smaller with larger sample sizes. In plot B, the AUCs differ slightly less for balanced class outcomes.



of the reliability coefficient ρ decreases, i.e., the amount of measurement error increases, the AUC decreases, or stays more or less constant in the case of sample size 200 and unbalanced data (where the AUC is already very low).

Only the main effect of measurement error ρ has a large effect size (*partial* $\eta^2 = .51$), confirming that there are differences in the AUC values for the different levels of ρ, regardless of the other design factors ($mean_{\rho=1} = .81 > mean_{\rho=.9} = .77 > mean_{\rho=.7} = .74 > mean_{\rho=.5} = .73$). Thus, 51% of the within subjects variance is accounted for by measurement error.

Table 3.1: Predictive accuracy as measured by CCR and AUC for different amounts of measurement error ρ, proportion of class outcome labels ($P(Y = 1) = .5$ or .8), data types and sample sizes for the continuous data. The values in the parentheses are the accuracies for the oracle. Data type is denoted by 1: predictors follow a normal distribution with common standard deviation, 2: each has a normal distribution with different standard deviations, and 3: all follow a multicollinear normal distribution.

Sample size	Data type	ρ	$P(Y=1)=.5$		$P(Y=1)=.8$	
			CCR	AUC	CCR	AUC
200	1	1	.81 (.92)	.81 (.92)	.82 (.91)	.54 (.76)
		.9	.75	.76	.81	.53
		.7	.71	.71	.81	.53
		.5	.69	.69	.81	.53
	2	1	.84 (.97)	.84 (.98)	.81 (.87)	.51 (.67)
		.9	.77	.77	.81	.51
		.7	.73	.73	.81	.51
		.5	.71	.71	.81	.51
	3	1	.91 (.97)	.91 (.98)	.81 (.87)	.56 (.82)
		.9	.86	.86	.81	.53
		.7	.84	.84	.81	.53
		.5	.83	.83	.81	.53
500	1	1	.94 (.97)	.94 (.97)	.84 (.86)	.64 (.71)
		.9	.89	.89	.84	.61
		.7	.86	.86	.84	.60
		.5	.83	.83	.84	.59
	2	1	.91 (.93)	.91 (.90)	.94 (.92)	.90 (.93)
		.9	.86	.86	.91	.82
		.7	.83	.83	.89	.77
		.5	.80	.80	.87	.72
	3	1	.96 (.97)	.96 (.96)	.95 (.99)	.91 (.93)
		.9	.91	.91	.93	.85
		.7	.88	.88	.92	.83
		.5	.86	.86	.91	.81
1000	1	1	.90 (.91)	.90 (.92)	.92 (.93)	.80 (.80)
		.9	.86	.87	.90	.78
		.7	.84	.84	.89	.77
		.5	.82	.82	.88	.76
	2	1	.94 (.96)	.94 (.97)	.87 (.86)	.70 (.66)
		.9	.88	.88	.85	.67
		.7	.85	.85	.85	.65
		.5	.82	.82	.85	.64
	3	1	.92 (.96)	.92 (.96)	.95 (.96)	.89 (.81)
		.9	.90	.90	.91	.82
		.7	.88	.88	.88	.76
		.5	.87	.87	.86	.73

In Table 3.2, we report the number of times the derived rules match the four true rules, out of 100 fitted models per cell. The average final ensemble size of the fitted models is also reported. We see that the ensemble size of the PRE models is way higher than the four rules that were used to generate the data. It seems that for a larger sample size, the ensemble size is larger.

Focusing on the average percent across the four rules that are correctly selected, or for each rules separately, we notice that the number of correctly selected rules is overall declining when the amount of measurement error rises. The within-subject interaction effect of the reliability coefficient ρ and data type[1] ($partial\ \eta^2 = .09$) has a medium effect size, which is illustrated with an interaction plot in Figure 3.2. The different amounts of measurement error and the mean number of correctly derived rules have an inverse relationship, which is consistent across all three data types. The largest contrasts were between data type 1 (all predictors follow a normal distribution with common standard deviation) and 3 (multicollinear normal distribution), with data type 1 yielding the highest accuracy.

The main effect of measurement error ρ has a large effect size ($partial\ \eta^2 = .71$), and indicates that when measurement error rises, then the number of correctly derived rules falls ($mean_{\rho=1} = .54 > mean_{\rho=.9} = .51 > mean_{\rho=.7} = .47 > mean_{\rho=.5} = .44$).

[1]Data type 1: all predictors follow a normal distribution with common standard deviation, 2: each has a normal distribution with different standard deviations, and 3: all follow a multicollinear normal distribution.

Figure 3.2: Two-way interaction plot, which plots the average number of correctly derived rules for two-way combinations of reliability coefficient ρ and data type. As the values of ρ decrease, i.e., the measurement error increases, the mean number of correctly derived rules drops for each of the three data types. Even the main effect of data type follows the same order as in the plot: data type 1 > data type 2 > data type 3, i.e., data with predictors following a normal distribution with common standard deviation, then following a normal distribution with different standard deviations, and then following a multicollinear normal distribution.

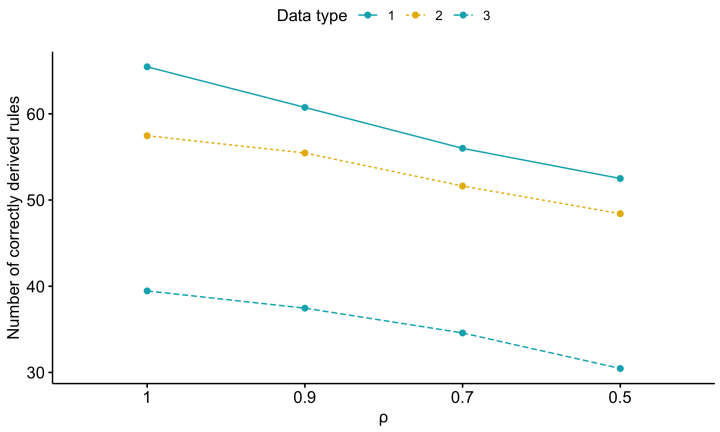
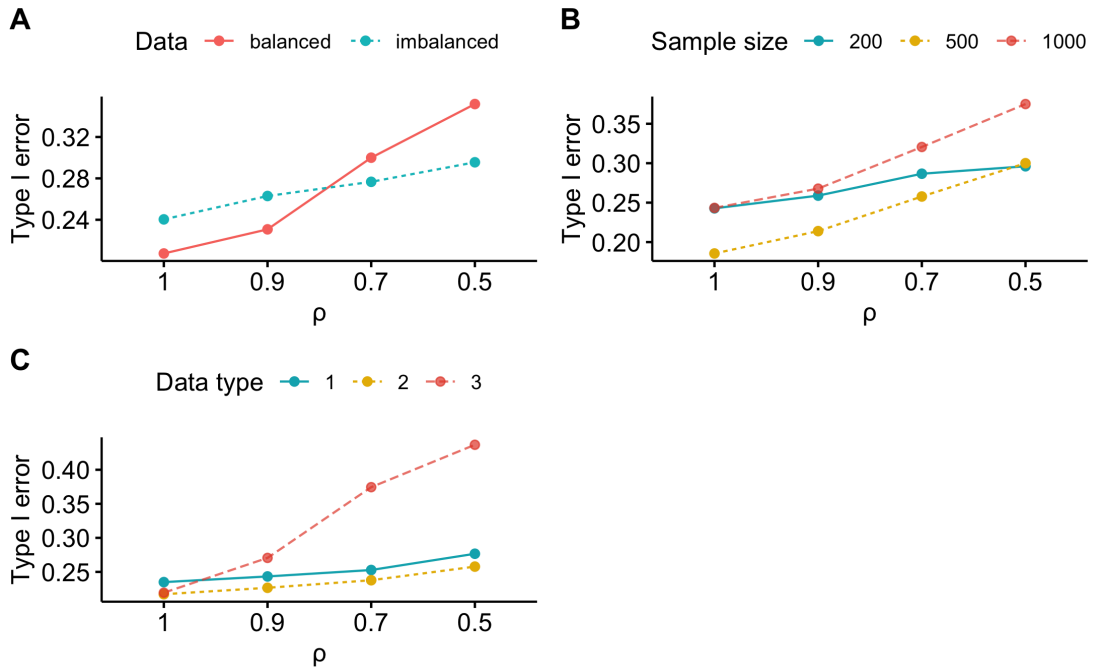


Table 3.2: The number of correctly derived rules from the model for different amounts of measurement error ρ, proportion of class outcome labels, data types and sample sizes for the continuous data. The numbers are also shown per matching true rule. We denote with "avg r" the average percent of the four correctly derived rules, and with "size r" the average ensemble size of prediction rules from the *pre* output.

Sample size	Data type	ρ	$P(Y=1)=.5$						$P(Y=1)=.8$					
			r_1	r_2	r_3	r_4	avg r	size r	r_1	r_2	r_3	r_4	avg r	size r
200	1	1	74	50	3	71	50	16	2	0	70	1	18	6
		.9	61	45	1	65	43	16	0	1	73	1	19	6
		.7	43	44	2	50	35	15	0	1	73	0	18	6
		.5	35	41	2	41	30	14	0	1	73	1	19	6
	2	1	53	79	1	37	42	17	0	6	21	1	7	5
		.9	47	76	3	36	40	16	2	3	25	0	8	5
		.7	31	74	2	24	33	15	2	3	25	0	8	5
		.5	14	74	2	19	27	14	2	3	26	0	8	5
	3	1	0	62	32	98	48	14	0	37	1	92	32	9
		.9	0	49	38	98	46	15	0	38	4	84	32	9
		.7	0	45	41	89	44	16	0	38	5	74	29	9
		.5	0	42	37	77	39	16	0	41	6	54	25	9
500	1	1	96	98	22	100	79	36	92	31	65	98	72	24
		.9	82	90	17	100	72	41	70	17	59	90	59	22
		.7	75	86	11	99	68	40	58	18	56	81	53	21
		.5	83	88	14	96	70	39	43	15	55	67	45	20
	2	1	15	32	19	98	41	32	99	96	2	92	72	29
		.9	7	29	14	98	37	34	96	98	0	88	70	31
		.7	3	22	10	99	34	32	93	96	0	75	66	30
		.5	3	30	8	97	34	30	80	95	2	59	59	28
	3	1	0	1	74	100	44	13	19	1	4	100	31	25
		.9	0	0	71	99	42	19	24	3	0	97	31	25
		.7	0	0	64	98	40	23	17	1	0	89	27	23
		.5	0	2	63	93	40	24	3	0	0	80	21	21
1000	1	1	91	99	55	100	86	42	91	98	69	95	88	38
		.9	98	100	47	100	86	53	77	100	73	91	85	41
		.7	98	98	38	98	83	52	67	100	67	81	79	37
		.5	99	99	35	97	82	50	47	100	61	67	69	34
	2	1	98	98	94	99	97	45	99	96	83	61	85	38
		.9	88	96	84	100	92	62	98	99	88	56	85	41
		.7	86	96	85	100	92	60	97	97	85	34	78	40
		.5	75	91	88	98	88	59	94	97	81	24	74	36
	3	1	0	51	39	100	48	26	1	36	1	98	34	28
		.9	0	46	20	98	41	33	0	40	1	89	32	32
		.7	0	36	19	98	38	34	0	35	0	81	29	32
		.5	0	28	18	89	34	34	0	39	0	59	24	32

Table 3.3 and 3.4 contain the type I and type II error as calculated for each predictor variable in a cell. The within-subject effects that explained most of the variance of type I error were the interactions between ρ and proportion of class outcome labels (*partial* $\eta^2 = .34$), between ρ and sample size (*partial* $\eta^2 = .19$), and between ρ and data type (*partial* $\eta^2 = .61$). Their interaction plots are shown in Figure 3.3, and for all of them, the higher the amounts of measurement error, the larger the type I error for every level of the other three factors.

Figure 3.3: Two-way interaction plots, which plot the type I error for two-way combinations of reliability coefficient ρ and proportion of class outcome labels, (plot A), ρ and sample size (plot B), and ρ and data type (plot C). For all the plot, as the amount of measurement error increases, the type I error increases for any of the levels of the other factors.



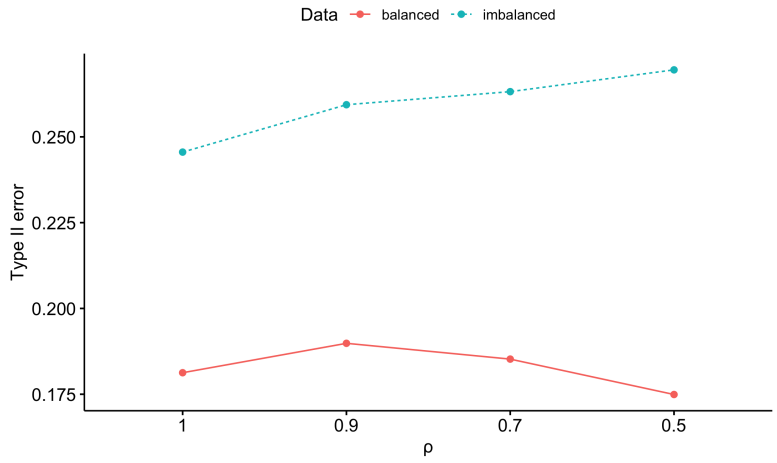
More specifically, the type I error increases rapidly for data type 3 (correlated predictor variables) and slightly for the independent predictor variable datasets (Fig. 3.3, C). It seems the larger the sample size, the higher the error across the ρ

values (Fig. 3.3, B), whereas for balanced data it starts having smaller error and ends up having larger than the imbalanced after $\rho = .7$ (Fig. 3.3, A). Also, the within-subject main effect of measurement error ρ has a large effect size (*partial* $\eta^2 = .68$), showing overall the same upward trend ($mean_{\rho=1} = .22 < mean_{\rho=.9} = .25 < mean_{\rho=.7} = .29 < mean_{\rho=.5} = .32$).

The only discrepancy in the reported results concerning type I error is observed for data type 3 and imbalanced data where the error values for $N = 500$ are much lower than $N = 200$ and $N = 1000$.

Moving on to the last performance measure, type II error, only the interaction between ρ and proportion of class outcome labels (*partial* $\eta^2 = .06$) has a medium effect size. As demonstrated in Figure 3.4, for both data scenarios, when the amount of measurement error increases from $\rho = 1$ to .9, the type II error also rises. After that the type II error still grows for the balanced scenario, while it goes down for the imbalanced class outcomes.

Figure 3.4: The average type II error per value of the reliability coefficient ρ was plotted for each proportion of class outcome labels. A different trend is presented for the imbalanced and balanced class outcomes, with the former having overall an upward trend while the later having a downward trend after the inclusion of measurement error.



Furthermore, from Table 3.3 and 3.4, it seems that there is a beneficial effect of sample size on Type II error only when the predictor variables are independent (i.e., data type 1 and 2).

Table 3.3: The type I and type II error for each of the corresponding predictor variables for different amounts of measurement error, data types and sample sizes for the continuous data. The results from specifying balanced class proportion labels are listed. We denote with "avg" the average values of the variables for the type I and II error per row.

Sample size	Data type	ρ	$P(Y=1)=.5$											
			Type I error				Type II error							
			X_1	X_2	X_4	avg	X_3	X_5	X_6	X_7	X_8	X_9	X_{10}	avg
200	1	1	.29	.32	.33	.31	.67	.16	.01	.09	.01	.01	.48	.20
		.9	.26	.28	.37	.30	.63	.23	.00	.06	.00	.01	.55	.21
		.7	.28	.29	.34	.30	.62	.26	.05	.07	.00	.02	.52	.22
		.5	.24	.27	.38	.30	.60	.26	.09	.08	.00	.01	.53	.22
	2	1	.32	.31	.30	.31	.72	.09	.03	.00	.01	.07	.54	.21
		.9	.30	.27	.34	.30	.63	.14	.02	.00	.02	.08	.49	.20
		.7	.29	.26	.34	.30	.65	.16	.03	.00	.01	.09	.49	.20
		.5	.27	.28	.34	.30	.67	.17	.18	.00	.01	.11	.48	.23
	3	1	.25	.23	.26	.25	.31	.17	.01	.13	.91	.00	.24	.25
		.9	.34	.32	.34	.33	.28	.17	.02	.16	.90	.00	.23	.25
		.7	.43	.48	.46	.46	.19	.14	.05	.15	.91	.00	.21	.24
		.5	.48	.47	.53	.49	.17	.13	.16	.11	.85	.00	.20	.23
500	1	1	.14	.15	.15	.15	.48	.01	.00	.01	.00	.00	.29	.11
		.9	.21	.19	.16	.19	.47	.03	.00	.03	.00	.00	.26	.11
		.7	.25	.24	.16	.22	.49	.04	.00	.04	.00	.00	.21	.11
		.5	.30	.31	.20	.27	.43	.03	.00	.05	.01	.00	.22	.11
	2	1	.21	.19	.30	.23	.21	.23	.00	.33	.03	.00	.40	.17
		.9	.20	.19	.18	.19	.29	.28	.00	.25	.06	.00	.38	.18
		.7	.25	.21	.23	.23	.30	.29	.00	.25	.01	.00	.44	.18
		.5	.28	.27	.27	.27	.37	.28	.00	.16	.02	.00	.40	.18
	3	1	.11	.07	.10	.09	.12	.79	.00	.74	.66	.00	.15	.35
		.9	.20	.22	.19	.20	.19	.67	.00	.72	.89	.00	.17	.38
		.7	.49	.37	.36	.41	.15	.54	.02	.50	.85	.00	.12	.31
		.5	.51	.51	.43	.48	.09	.37	.03	.37	.76	.00	.07	.24
1000	1	1	.28	.17	.19	.21	.33	.01	.00	.00	.01	.00	.04	.06
		.9	.19	.23	.15	.19	.28	.00	.00	.00	.00	.00	.10	.05
		.7	.24	.28	.22	.25	.35	.00	.00	.00	.00	.00	.13	.07
		.5	.34	.32	.32	.33	.30	.00	.00	.01	.00	.00	.09	.06
	2	1	.10	.15	.17	.14	.04	.00	.00	.01	.00	.00	.03	.01
		.9	.17	.19	.14	.17	.10	.01	.00	.01	.00	.00	.04	.02
		.7	.21	.24	.17	.21	.10	.01	.00	.02	.00	.00	.05	.03
		.5	.23	.30	.31	.28	.06	.03	.00	.02	.00	.00	.02	.02
	3	1	.20	.20	.11	.17	.19	.14	.00	.29	.87	.00	.35	.26
		.9	.22	.20	.18	.20	.41	.15	.01	.29	.91	.00	.34	.30
		.7	.37	.27	.37	.34	.40	.18	.02	.34	.83	.00	.37	.31
		.5	.50	.39	.45	.45	.40	.17	.05	.25	.76	.00	.39	.29

Table 3.4: The type I and type II error for each of the corresponding variables for different amounts of measurement error, data types and sample sizes for the continuous data. The results from specifying unbalanced class proportion labels are listed.

Sample size	Data type	ρ	$P(Y=1)=.8$											
			Type I error				Type II error							
			X_1	X_2	X_4	avg	X_3	X_5	X_6	X_7	X_8	X_9	X_{10}	avg
200	1	1	.13	.11	.12	.12	.22	.80	.78	.72	.52	.45	.07	.51
		.9	.08	.17	.15	.13	.20	.69	.78	.73	.49	.47	.08	.49
		.7	.10	.17	.15	.14	.19	.69	.79	.74	.50	.44	.07	.49
		.5	.10	.17	.13	.13	.17	.69	.81	.73	.5	.45	.06	.49
	2	1	.12	.16	.17	.15	.61	.65	.82	.41	.49	.62	.37	.57
		.9	.13	.13	.14	.13	.58	.77	.84	.49	.45	.71	.34	.60
		.7	.13	.13	.15	.14	.56	.75	.81	.47	.45	.72	.35	.59
		.50	.12	.12	.15	.13	.55	.77	.84	.48	.46	.72	.37	.60
	3	1	.29	.34	.32	.32	.53	.22	.03	.23	.69	.00	.37	.30
		.9	.29	.34	.41	.35	.60	.35	.10	.22	.71	.01	.44	.35
		.7	.29	.39	.48	.39	.56	.30	.20	.23	.72	.00	.37	.34
		.50	.34	.42	.52	.43	.52	.30	.35	.20	.71	.01	.40	.36
500	1	1	.29	.34	.31	.31	.25	.22	.00	.20	.00	.00	.03	.10
		.9	.36	.34	.29	.33	.18	.10	.01	.30	.00	.00	.02	.09
		.7	.35	.34	.25	.31	.19	.12	.03	.28	.00	.00	.04	.09
		.50	.41	.34	.27	.34	.19	.14	.08	.29	.00	.00	.04	.11
	2	1	.20	.19	.10	.16	.54	.01	.00	.03	.00	.00	.70	.18
		.9	.20	.21	.19	.20	.54	.00	.00	.00	.00	.00	.72	.18
		.7	.18	.28	.18	.21	.55	.01	.00	.01	.00	.00	.66	.18
		.5	.24	.30	.20	.25	.52	.00	.00	.00	.00	.00	.64	.17
	3	1	.20	.13	.16	.16	.32	.57	.00	.61	.12	.00	.43	.29
		.9	.17	.15	.20	.17	.53	.67	.00	.67	.09	.01	.55	.36
		.7	.13	.16	.21	.17	.65	.71	.06	.75	.15	.06	.65	.43
		.5	.15	.19	.22	.19	.72	.71	.12	.76	.26	.09	.79	.49
1000	1	1	.32	.27	.32	.30	.07	.01	.00	.01	.04	.00	.03	.02
		.9	.29	.33	.33	.32	.06	.00	.01	.00	.07	.02	.04	.03
		.7	.25	.32	.32	.30	.06	.00	.04	.00	.06	.01	.05	.03
		.5	.22	.34	.32	.29	.10	.00	.12	.00	.07	.02	.03	.05
	2	1	.25	.35	.32	.31	.07	.01	.00	.00	.00	.01	.00	.01
		.9	.32	.45	.33	.37	.07	.00	.00	.00	.00	.04	.00	.02
		.7	.26	.44	.33	.34	.08	.00	.00	.00	.00	.03	.00	.02
		.5	.25	.42	.29	.32	.08	.00	.01	.00	.00	.01	.00	.01
	3	1	.32	.31	.35	.33	.41	.10	.01	.40	.21	.00	.46	.23
		.9	.34	.36	.40	.37	.53	.11	.03	.36	.09	.00	.47	.23
		.7	.5	.45	.53	.49	.49	.08	.04	.35	.06	.00	.40	.20
		.5	.56	.52	.67	.58	.39	.01	.04	.30	.05	.00	.31	.16

3.2 Results concerning binary predictors

Similar to the previous section, we assess using only binary predictor variables how the PRE method performs for the different amounts of measurement error regarding the other two design factors; proportion of class outcome labels and sample size. Here, we address measurement error with the misclassification rate (MR) from Section 2.3, instead of the reliability coefficient ρ. In the following paragraphs, the most important effects coming from the ANOVAs are reported for every performance measure.

Starting with the results for the prediction performance of the method, the average CCR and AUC within a cell is reported in Table 3.5. Alongside the PRE's AUC is also the AUC from the "oracle" (Section 2.4.3). We assess whether those values are different or not, by a repeated measures ANOVA, from which the main effect of fitting method has a medium effect size ($partial\ \eta^2 = .07$). This supports that overall the "oracle's" AUC is higher than the one from the PRE models ($mean_{\text{"true" AUC}} = .84 > mean_{\text{derived AUC}} = .81$).

Moreover, the interaction between the fitting method and sample size has a medium effect size ($partial\ \eta^2 = .6$), indicating that the relationships between PRE's and "oracle's" AUC strongly depends on sample size. Figure 3.5 shows that the difference between PRE's and "oracle's" AUC becomes smaller with increasing sample size, especially when $N = 1500$. Also, for $N = 1500$ the AUC values of PRE and "oracle" slightly dropped, with the error bars (based on standard error of each mean) showing an overlap especially for PRE with the ones for $N = 1000$ and $N = 2000$. This could be a result of the variability of the estimates.

Furthermore, we notice that in each cell as the misclassification rate (MR) is in-

creasing, i.e., the amount of measurement error is increasing, the AUC is declining. This is confirmed by the within-subject main effect of misclassification rate, which has a large effect size ($partial\ \eta^2 = .61$), and by calculating the mean AUC per misclassification rate ($mean_{MR=0} = .81 > mean_{MR=13} = .74 > mean_{MR=25} = .71$).

Figure 3.5: The average AUC per fitting method was plotted for each sample size. Overall, the differences between PRE's and "oracle's" AUC become smaller with increasing sample size.

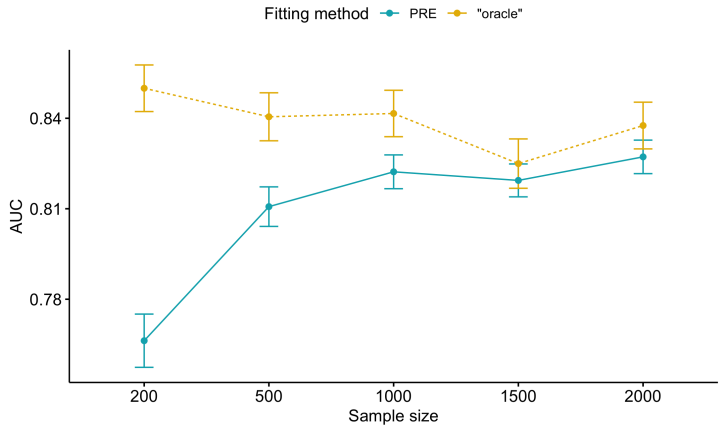


Table 3.5: Predictive accuracy as measured by CCR and AUC for different amounts of measurement error (misclassification rate), proportion of class outcome labels ($P(Y = 1) = .5$ or .8) and sample sizes for the binary data. The values in the parentheses are the accuracies for the oracle.

Sample size	Misclassification rate	$P(Y = 1) = .5$		$P(Y = 1) = .8$	
		Accuracy	AUC	Accuracy	AUC
200	0	.87 (.89)	.87 (.90)	.84 (.87)	.66 (.80)
	13	.81	.81	.82	.58
	25	.77	.77	.82	.55
500	0	.89 (.89)	.89 (.90)	.86 (.87)	.73 (.79)
	13	.83	.83	.84	.66
	25	.79	.79	.84	.62
1000	0	.89 (.89)	.89 (.89)	.86 (.87)	.76 (.79)
	13	.84	.84	.85	.66
	25	.80	.79	.84	.64
1500	0	.88 (.89)	.88 (.89)	.86 (.87)	.76 (.76)
	13	.84	.84	.85	.68
	25	.80	.80	.85	.66
2000	0	.89 (.89)	.89 (.89)	.86 (.86)	.76 (.78)
	13	.84	.84	.85	.68
	25	.80	.80	.85	.65

In Table 3.6, we report the percentage of datasets in which each of the true rules were selected per cell. It is noteworthy that most numbers are very high, especially for larger sample sizes and balanced class outcomes.

The within-subject effects that explained most of the variance of the number of correct rules, were the interactions of MR and proportion of class outcome labels (*partial* $\eta^2 = .41$), MR and sample size (*partial* $\eta^2 = .67$), and MR and rule (*partial* $\eta^2 = .09$). Their interaction plots are shown in Figure 3.6, and for all of them, the higher the amount of measurement error, the smaller the number of correct rules for every level of the other three factors. More specifically, the balanced class outcomes had higher number of correct rules (Fig. 3.6, A). Rule r_2 has the higher number of correct rules which is not affected by MR as the rest

rules, then follows r_1, r_4, and last r_3 having the lowest numbers (Fig. 3.6, B). Moreover, the higher the sample size the larger the number of correct rules (Fig. 3.6, C). The main effect of MR has a large effect size (*partial* $\eta^2 = .87$) with a decreasing mean number of correctly selected rules per MR ($mean_{MR=0} = .92 > mean_{MR=13} = .85 > mean_{MR=25} = .78$).

Figure 3.6: The average number of correctly recovered rules is plotted for two-way combinations of misclassification rate (MR) and proportion of class outcome labels (plot A), MR and prediction rule (plot B), and MR and sample size (plot C). For all the plots, as the amount of MR increases, the number of correct rules drops for the levels of the other factors. The numbers in the prediction rules correspond to: $r_1 = I(X_6 = 0) \cdot I(X_8 = 0)$, $r_2 = I(X_5 = 1) \cdot I(X_7 = 1)$, $r_3 = I(X_3 = 0) \cdot I(X_{10} = 1)$, $r_4 = I(X_6 = 1) \cdot I(X_9 = 0)$.

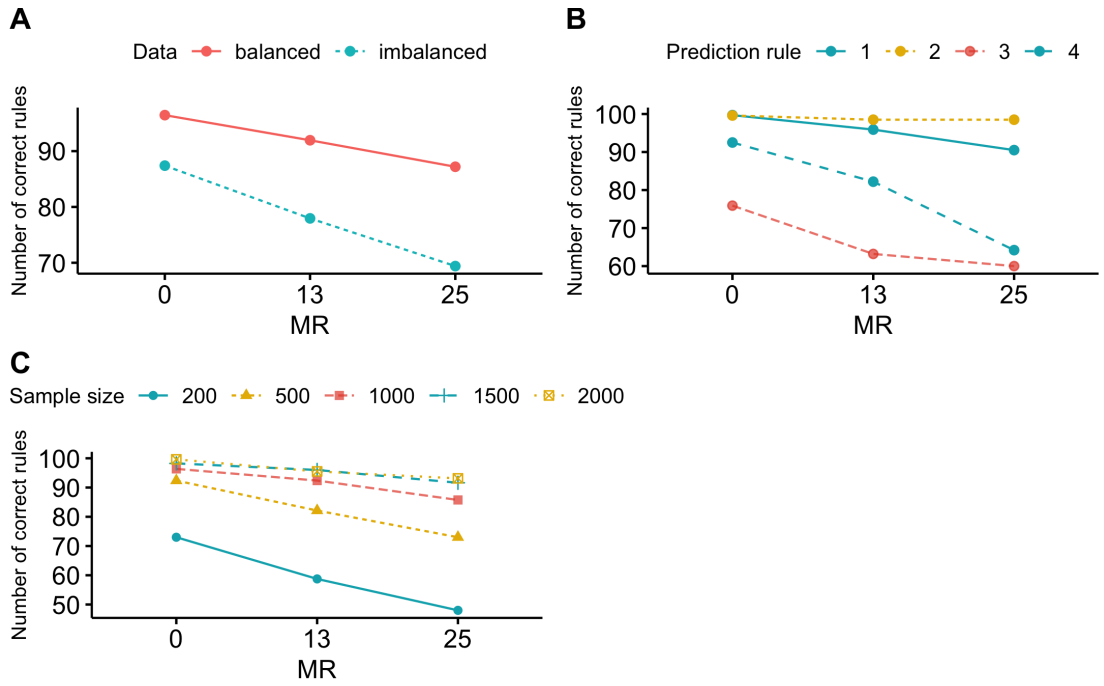


Table 3.6: The number of correctly derived rules from the model for different misclassification rates (MR), proportion of class outcome labels and sample sizes for the binary data. The numbers are also shown per matching true rule. We denote with "avg r" the average percent of the four correctly derived rules, and with "size r" the average ensemble size of prediction rules from the *pre* output.

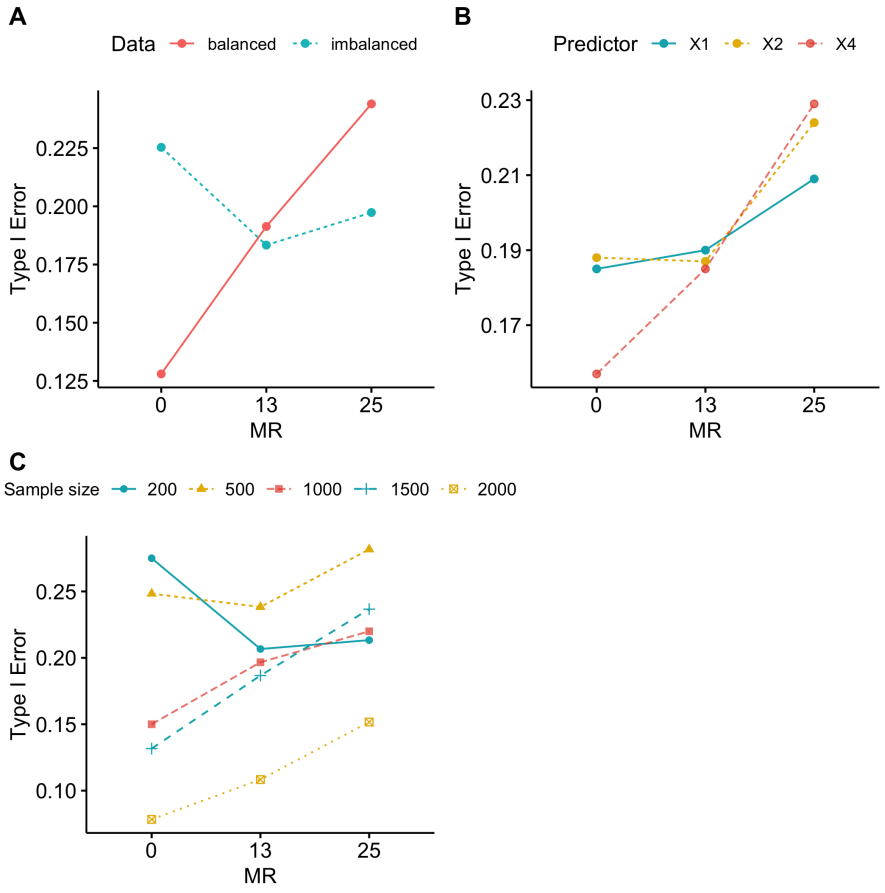
Sample size	MR	$P(Y=1)=.5$						$P(Y=1)=.8$					
		r_1	r_2	r_3	r_4	avg r	size r	r_1	r_2	r_3	r_4	avg r	size r
200	0	97	96	43	99	84	16	100	100	18	31	62	11
	13	76	85	29	79	67	15	83	100	13	5	50	9
	25	65	85	25	39	54	13	56	100	12	2	42	8
500	0	100	100	94	100	98	20	100	100	49	96	86	17
	13	100	100	77	100	94	20	100	100	36	44	70	15
	25	92	100	70	81	86	20	95	100	33	13	60	14
1000	0	100	100	100	100	100	20	100	100	72	99	93	19
	13	100	100	94	100	98	23	100	100	51	94	86	19
	25	98	100	94	99	98	24	100	100	45	50	74	18
1500	0	100	100	100	100	100	19	100	100	86	100	96	19
	13	100	100	99	100	100	23	100	100	69	100	92	20
	25	100	100	99	100	100	26	100	100	64	70	84	21
2000	0	100	100	100	100	100	18	100	100	97	100	99	20
	13	100	100	100	100	100	25	100	100	64	100	91	20
	25	99	100	99	99	99	27	100	100	59	89	87	21

Table 3.7 and 3.8 contains the type I and type II error per predictor and design factor combination. Focusing on type I error, the repeated measures ANOVA revealed that the interactions between MR and proportion of class outcome labels (*partial* $\eta^2 = .84$), and between MR and sample size (*partial* $\eta^2 = .77$) have a large effect size, while the interaction between MR and predictor (*partial* $\eta^2 = .11$) has a medium effect size. Type I error increases rapidly for the balanced class outcomes, whereas for the imbalanced class outcomes drops when $MR = 13$ and then increases for $MR = 25$ (Fig. 3.7, A). For most predictor variables, type I error rises, with X_4 having the sharpest increase (Fig. 3.7, B). The predictor variables X_1 is more similar to X_2, which could be because the distribution of X_1 is closer to the the distribution of X_2. Moreover, the type I error increases for almost all sample size with larger misclassification rates, except for $N = 200$, which ends up

with values closer to those from large sample sizes of 1000 and 1500. Sample size of 2000 had the lowest Type I error rates (Fig. 3.7, C).

The main effect of MR has a large effect size (*partial* $\eta^2 = .66$) with the mean of type I error dropping as misclassification rate rises ($mean_{MR=0} = .18 < mean_{MR=13} = .19 < mean_{MR=25} = .22$).

Figure 3.7: Two-way interaction plots, which plot the type I error for two-way combinations of misclassification rate (MR) and proportion of class outcome labels, (plot A), MR and predictior variable (plot B), and MR and sample size (plot C). In most contrasts, as the amount of MR increases, the type I error increases.



Lastly, we can observe the type II errors, where the majority of the values are zero, indicating that all of the true rules were selected in the fitted PREs. The

interactions between MR and proportion of class outcome label (*partial* $\eta^2 = .71$) and between MR and sample size (*partial* $\eta^2 = .69$) have a large effect size. Their interaction plots are shown in Figure 3.8, and for every level of the factors, it seems that the type II errors stay the same for both non-zero MR. The main effect of MR has a large effect size (*partial* $\eta^2 = .85$) with the mean number of correctly selected rules per MR to be decreasing ($mean_{MR=0} = .03 < mean_{MR=13} = .05, mean_{MR=25} = .05$).

Figure 3.8: Two-way interaction plots, which plot the means of type II errors for two-way combinations of MR and proportion of class outcome labels, (plot A), and MR and sample size (plot B). In both, as the amount of measurement error increases, the type II error increases, or remains the same when measurement error is present.

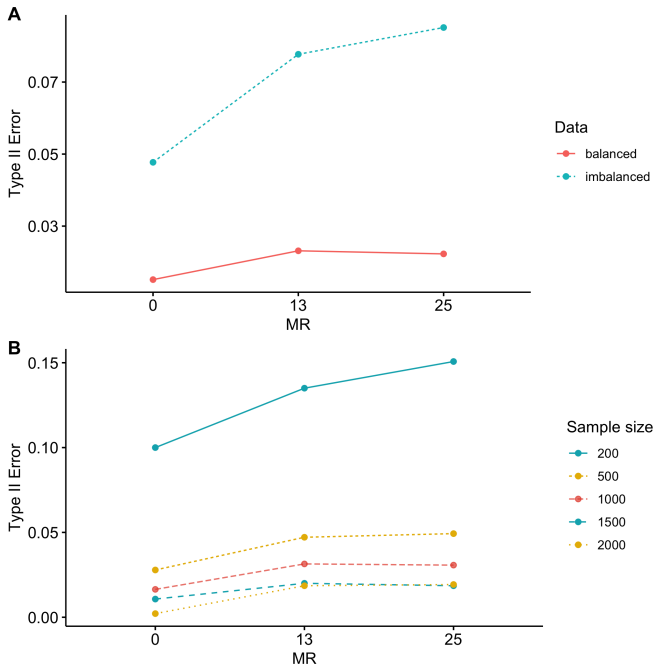


Table 3.7: The type I and type II error for each of the corresponding variables for different misclassification rates (MR) and sample sizes for the binary data. The results from specifying balanced class proportion labels are listed.

Sample size	MR	$P(Y=1)=.5$											
		Type I error				Type II error							
		X_1	X_2	X_4	avg	X_3	X_5	X_6	X_7	X_8	X_9	X_{10}	avg
200	0	.30	.25	.21	.25	.30	.00	.00	.00	.00	.00	.17	.07
	13	.22	.25	.22	.23	.39	.02	.01	.05	.00	.00	.15	.09
	25	.23	.24	.23	.23	.39	.04	.04	.03	.00	.00	.10	.09
500	0	.20	.24	.15	.20	.03	.00	.00	.00	.00	.00	.03	.01
	13	.26	.21	.27	.25	.11	.00	.00	.00	.00	.00	.02	.02
	25	.36	.28	.34	.33	.13	.00	.00	.00	.00	.00	.02	.02
1000	0	.11	.07	.05	.08	.00	.00	.00	.00	.00	.00	.00	.00
	13	.27	.19	.21	.22	.02	.00	.00	.00	.00	.00	.02	.01
	25	.28	.26	.18	.24	.00	.00	.00	.00	.00	.00	.02	.00
1500	0	.05	.12	.10	.09	.00	.00	.00	.00	.00	.00	.00	.00
	13	.19	.14	.17	.17	.01	.00	.00	.00	.00	.00	.01	.00
	25	.24	.25	.31	.27	.01	.00	.00	.00	.00	.00	.00	.00
2000	0	.05	.01	.01	.02	.00	.00	.00	.00	.00	.00	.00	.00
	13	.09	.11	.07	.09	.00	.00	.00	.00	.00	.00	.00	.00
	25	.13	.16	.17	.15	.00	.00	.00	.00	.00	.00	.00	.00

Table 3.8: The type I error and type II error for each of the corresponding variables for different misclassification rates (MR) and sample sizes for the binary data. The results from specifying unbalanced class proportion labels are listed.

Sample size	MR	$P(Y=1)=.8$											
		Type I error				Type II error							
		X_1	X_2	X_4	avg	X_3	X_5	X_6	X_7	X_8	X_9	X_{10}	avg
200	0	.24	.32	.33	.30	.56	.00	.00	.00	.00	.18	.19	.13
	13	.18	.19	.18	.18	.58	.00	.06	.00	.05	.24	.34	.18
	25	.17	.23	.18	.19	.56	.00	.26	.00	.07	.24	.38	.22
500	0	.33	.33	.24	.30	.26	.00	.00	.00	.00	.00	.07	.05
	13	.20	.25	.24	.23	.42	.00	.00	.00	.00	.01	.10	.08
	25	.19	.26	.26	.24	.43	.00	.00	.00	.00	.00	.11	.08
1000	0	.24	.23	.20	.22	.17	.00	.00	.00	.00	.00	.06	.03
	13	.16	.17	.18	.17	.31	.00	.00	.00	.00	.00	.09	.06
	25	.20	.20	.20	.20	.34	.00	.00	.00	.00	.00	.07	.06
1500	0	.22	.15	.15	.17	.10	.00	.00	.00	.00	.00	.05	.02
	13	.19	.24	.19	.21	.20	.00	.00	.00	.00	.00	.06	.04
	25	.17	.19	.26	.21	.17	.00	.00	.00	.00	.00	.08	.04
2000	0	.11	.16	.13	.13	.02	.00	.00	.00	.00	.00	.01	.00
	13	.14	.12	.12	.13	.16	.00	.00	.00	.00	.00	.10	.04
	25	.12	.17	.16	.15	.18	.00	.00	.00	.00	.00	.09	.04

Chapter 4

Discussion

In this thesis, we studied the impact of measurement error on prediction rule ensembles through simulation studies. We examined whether the performance measures of the fitted PREs deteriorates when introducing measurement error in one of the predictor variables of a dataset. Those measures were the predictive performance measure, AUC, and others that measure the correctness of prediction rules, such as the number of correctly selected rules, type I and II error.

We found that with larger sample sizes, the predictive accuracy of PREs became more similar or even the same as that of the "oracle". Altogether, measurement error not only deteriorates the predictive performance of PREs, it can also aggravate the interpretability of the method, by selecting wrong rules, resulting in unreliable and wrong conclusions. We discuss further the findings in the following paragraphs.

4.1 Discussion of results regarding the ”oracle”

We saw that the predictive accuracy of PREs was lower than that of the ”oracle”. This was expected, since the ”oracle” was used as a benchmark, indicating the theoretically best possible predictive accuracy. However, this method was not the perfect predictor, although the true underlying rules were used. A possible explanation could be that the estimated coefficient values from the GLM deviated somewhat from the true underlying models used for the data generation.

The most important finding was that with larger sample sizes, the predictive accuracy of PREs became more similar or even the same as that of the ”oracle”, for both continuous and binary predictor settings.

4.2 Discussion of results regarding measurement error

In pursuit of answering whether measurement error affects the AUC metric of the method, we concluded that with larger amount of measurement error, the AUC decreased. The AUC metric did not seem to be influenced by the other design factors for the continuous and binary predictor variable case.

In the study with only continuous predictors, different factors had an influence on the other metrics that were deteriorated with larger amounts of measurement error. For multicollinear predictor variables, the numbers of correctly selected rules and type I error are more negatively affected with larger amounts of measurement error compared to the cases with independent predictor variables. This could be because more correlated predictors with the mismeasured predictor are picked

instead. Type I error was also higher for large sample size across the ρ values, and for balanced class outcomes when ρ value is also high. Type II error grew for balanced data, whereas declined for imbalanced data with higher ρ values.

In the study with only binary predictors, sample size and proportion of class outcome labels had an effect on the relationship between measurement error and each of the three correctness-related measures. Mostly, higher sample sizes or balanced class outcomes would result in better performances across the misclassification rates. In addition to those factors, for some prediction rules the number of correctly selected rules would not be affected much from the misclassification rates. The one that seems to not be affected does not include the mismeasured predictor. The type I error also increases differently for each of its predictors with larger misclassification rates.

It is worth mentioning that the number of correctly selected rules and type II error reached the ideal values of 100 and 0 correspondingly for higher sample sizes and balanced data in the binary setting. However, type I errors remained substantial, meaning although almost always the correct rules were selected, also additional rules were selected having predictors that were not used to construct the true rules. Hence, adding some noise in the derived ensemble of prediction rules.

Furthermore, we noticed that the ensemble size of the PRE models was way higher than the four rules that were used to generate the data. The ensemble size was particularly large for larger sample sizes. This makes difficult to interpret the output of the fitted models when the number of prediction rules is large, whereas only four rules would be enough to adequately explain the model.

To conclude with the results from both studies, the predictive performance of

PRE models (via AUC) was deteriorated after introducing measurement error in one of its predictor variables. In general, although the number of correctly selected rules was from the start not great, after the inclusion of measurement error the numbers got even worse. The same applies for type I and II error, where those measures were not focused on whether the correct rules were selected, but whether the right or wrong predictors were included in the derived rules of the model. Hence, measurement error can cause serious implications in the interpretability of the model, and result to unreliable and wrong conclusions.

4.3 Limitations and suggestions for future research and development

Certain suggestions can be given to expand the simulation set-up further. Starting with the data generation, more covariance matrix settings could be tried for generating the predictor variables based on a normal distribution. For instance, the covariance of pairs of interacting variables could be set to different values. Using only the three data settings that were opted for this study, the findings could only be generalized to data samples of the same underlying population of similar characteristics. This means studies with data of different sample sizes that come from a normal distribution with similar descriptive characteristics (e.g., means, standard deviations or covariances) and amounts of measurement error will not differ in their effect size estimates. Hence, the results can be considered indicative, but not definitive. Replication with other populations or conditions could help to generalize further the findings. For example, either a non-symmetrical distri-

butions could be opted, where the data points are skewed either to the left or to the right (e.g., chi-squared distribution), or a multimodal distribution, such as a mixture of multiple normal distributions.

The design of the experiment could also be adjusted with the "oracle" getting the true rule memberships, but based on variables with measurement error. In this way, we could assess whether also the "oracle" is affected by measurement error. A penalized logistic regression could also be used to be more comparable with the PRE model that uses lasso regression. The current results indicate that more noise variables are selected with increasing sample size and with increasing multicollinearity, which is likely due to the use of lasso penalized regression in PREs.

During the model fitting, instead of having different simulation studies concerning continuous and binary predictors separately, datasets with both kind of predictors could be simulated to assess the performance of the method, including measurement error either in a continuous or binary predictor, or in both at the same time. More extreme amounts of measurement error could be included. Smaller or larger sample sizes than the current ones can be explored as either scenario is relevant for certain fields where subjects are limited or widely available. Different number of predictor variables could also be explored, such as a combination of a small dataset with a large number of predictor variables would result in a high-dimensional setting. A higher test sample size could also be opted in order to decrease the variability and sample error in the reported mean estimates. Another suggestion would be to repeat this study implementing regression or multiclass classification instead of binary classification.

In the analyses, because rules from perfectly correlated pairs of rules are re-

moved from the initial ensemble, recovery measures of the fitted PREs are negatively affected. Hence, if one of the true rules is omitted by the method but its negatively perfectly correlated pair is included in the final ensemble, this will not be picked up by the number of correctly selected rules. However, this does not affect type I and II error.

4.4 Contribution and novelty

This thesis aimed primarily to contribute in assessing the impact of measurement error on the performance of prediction rule ensembles. In some papers (e.g., Quinlan, 1986; Quinlan,1993), decision tree classification on uncertain data has been addressed for decades in the form of missing values, and solutions approximating missing values have been developed. In other papers (e.g., Tsang et al.; 2009; Qin et al., 2009; Sexton & Laake, 2007), tree-based algorithms for data with measurement error have also been developed. However, through the `pre` package conditional inference trees were used for deriving prediction rules which use a predictor variable selection scheme that is based on statistical theory that address the variable selection bias problem, but not measurement error. Studies focusing on the effect of measurement error on the conditional inference trees, or on the prediction rule ensembles have not been conducted before. Therefore, in this thesis extensive experiments have been conducted to investigate whether the performance of the method deteriorates after introducing measurement error in various design settings. The predictive performance of the method was recorded, together with measures related to the correctness of prediction rules which are used to make inference. Moreover, a measurement error methodology was presented and applied

both on a continuous and binary predictor.

Appendix A

Main R code

```
if (!"pre" %in% installed.packages()) {
  if (!"devtools" %in% installed.packages()) {
    install.packages("devtools")}
  require(devtools)
  install_github("marjoleinF/pre")}
require(pre)

if (!"mvtnorm" %in% installed.packages()) {
  install.packages("mvtnorm")}
require(mvtnorm)

if (!"plotrix" %in% installed.packages()) {
  install.packages("plotrix")}
require(plotrix)
```

```
if (!"caret" %in% installed.packages()) {
  install.packages("caret")}
require(caret)

if (!"pROC" %in% installed.packages()) {
  install.packages("pROC")}
require(pROC)

SP <- function(i, X.mat) {
  # selects a value between the 1st and 3th quantile of the i given
  #  variable to be used as the split point of the variable in a
  # decision rule
  #
  # Args:
  #   i: an index that indicates which predictor (column) will be
  #      selected from X.mat
  #   X.mat: a matrix containing all the predictors in its columns
  #
  # Returns:
  #   a value indicating the relevant split point of a variable in
  #   a decision rule

  x <- X.mat[, i]
  return(round(sample(quantile(x, 0.25):quantile(x, 0.75), 1), 1))
}
```

```
InitPar <- function(N, set.sd, p = 10, balanced = TRUE) {
  # Initializes parameters to be used for simulating data
  #
  # Args:
  #   N: sample size of data set
  #   set.sd: type of predictor variables in the data,
  #           1 - normally distributed with common sd;
  #           2 - normally distributed with different sd each;
  #           3 - normally distributed given a covariance matrix;
  #           4 - binomially distributed with different probabilities
  #   p: number of predictor variables (default is 10)
  #   balanced: whether the classes of the simulated binary target
  #             variable are balanced or not
  #
  # Returns:
  #   a list with two distribution parameters; the mean and par2
  #   (i.e., probabilities for set.sd=4, or the sd otherwise), and
  #   a vector of parameters alpha to be used in the linear
  #   predictor F(x)


  # A different intercept of the ensemble member is given for each
  # combination of sample size and data type depending if the
  #classes of the binary target variable are balanced or not
  if (balanced) {
    a0 <- matrix(c(-1.6, -3.8, -14.5, -6.8,
```

```
                    -5.5, -10, -9, -6.8,
                    -1.5, -5.5, -9.2, -6.8,
                    0, 0, 0, -6.8,
                    0, 0, 0, -6.8), nrow =5, byrow = TRUE)

  } else {
    a0 <- matrix(c(1.4, 0.78, -4.7, 0,
                    0.8, -1.5, 1, 0,
                    1.72, 0.62, -2.5, 0,
                    0, 0, 0, 0,
                    0, 0, 0, 0), nrow =5, byrow = TRUE)
  }
  colnames(a0) <- c(1, 2, 3, 4)
  rownames(a0) <- c(200, 500, 1000, 1500, 2000)

  # parameters of each ensemble member
  intercept <- a0[as.character(N), set.sd]
  alpha <- c(intercept, runif(4, -25, 25))

  if(set.sd == 4) {
    par2 <- sample(seq(0.2, 0.8, 0.01), p)
    mean <- NULL

  } else {
    mean <- sample(seq(10, 200, 0.1), p)
```

```
    if(set.sd == 1) par2 <- sample(seq(5, 20, 0.1), 1)
    if(set.sd == 2) par2 <- sample(seq(0.2, 10, 0.1), p)
    if(set.sd == 3) par2 <- matrix(rep(0.5, p^2), ncol = p)
                            + diag(0.5, p)

  }
  return(list(mean = mean, par2 = par2, alpha = alpha))
}


SimData <- function(N, mean, par2, set.sd, alpha,
                    sp = NULL, rho = NULL, p = 10) {
  # Simulates data containing a binary target outcome and
  # 10 predictors for given alpha, mean and sd or probabilities.
  # The target variable is based on a binomial distribution
  # with probability computed from a standard logistic function
  # with parameter the linear predictor formed by 4 ensemble rules.
  # Also, measurement error can be added in one predictor.
  #
  # Args:
  #   N: sample size of data set
  #   mean: vector of means to generate N values from
  #         a normal distribution
  #   par2: a vector of probabilities for set.sd=4,
  #         or one sd for set.sd=1, or a vector of
  #         sd's for set.sd=2, or a covariance matrix for set.sd=3
```

```
#   set.sd: type of predictor variables in the data,
#           1 - normally distributed with common sd;
#           2 - normally distributed with different sd's;
#           3 - normally distributed given a covariance matrix;
#           4 - binomially distributed with different probabilities
#   alpha: a vector of parameters of each of the 4 ensemble
#          members (intercept is set to 0)
#   sp: vector of split points for each predictor;
#       if NULL, then SP() is called to compute the split points
#   rho: the reliability coefficient, determines the sizes
#        of measurement error; for set.sd = 4, if rho=1, then
#        misclassification is around 13%, if rho=2, then is around
#        25%; for set.sd != 4, rho can be numeric with 0 < rho < 1
#   p: number of predictors (default is 10)
#
# Returns:
#   a list with data: data with a target outcome variable and 10
#                     predictors to be used for
#                     training a prediction rule ensemble method;
#               data.LR: data with a target outcome variable and
#                        the 4 rules as predictors;
#               sp: a vector with the i-th element indicating the
#                   sampled split point for the i-th predictor
#                    (the 11-th element indicates a second split
#                   point for X6 as it is used twice in the rules);
#               misclas: number of misclassifications,
```

```
#                          if set.sd = 4 and measurement error
#                          is added, otherwise NULL

if(set.sd == 1) X.mat <- round(as.data.frame(sapply(1:p,
                          function(i) rnorm(N, mean[i], par2))), 1)

if(set.sd == 2) X.mat <- round(as.data.frame(sapply(1:p,
                          function(i) rnorm(N, mean[i], par2[i]))), 1)

if(set.sd == 3) X.mat <- round(as.data.frame(
                                    rmvnorm(N, mean, par2)), 1)

if(set.sd == 4) X.mat <- as.data.frame(sapply(1:p,
                          function(i) rbinom(N, 1, par2[i])))
colnames(X.mat) <- paste0("X", 1:p)

# specify 4 ensemble rules
if(set.sd != 4) {
  if(is.null(sp) & !is.null(mean)) {
    sp <- c(sapply(1:p, SP, X.mat), SP(6, X.mat))
    names(sp) <- c(colnames(X.mat), "X6.2")}

  F1 <- X.mat$X6 > sp[6] & X.mat$X8 > sp[8]
  F2 <- X.mat$X5 <= sp[5] & X.mat$X7 <= sp[7]
  F3 <- X.mat$X3 > sp[3] & X.mat$X10 > sp[10]
  F4 <- X.mat$X6 <= sp[11] & X.mat$X9 <= sp[9]
```

```
} else {
  F1 <- X.mat$X6 == 0 & X.mat$X8 == 0
  F2 <- X.mat$X5 == 1 & X.mat$X7 == 1
  F3 <- X.mat$X3 == 0  & X.mat$X10 == 1
  F4 <- X.mat$X6 == 1 & X.mat$X9 == 0}

# forming linear predictor F(x)
Fx <- cbind(1, F1, F2, F3, F4) %*% alpha
# probability of each sample to get the class label 1
pr <- exp(Fx) / (1 + exp(Fx))

# adding measurement error if it is specified by rho
data <- X.mat
# for the continuous cases
if(!is.null(rho) & set.sd != 4) data[, 6] <-
                MeasurementError(X.mat = X.mat, N = N,
                                 rho = rho, par2 = par2,
                                 set.sd = set.sd)
# for the binary predictors
misclas <- NULL
if(!is.null(rho) & set.sd == 4) {
  me <- MeasurementError(X.mat = X.mat, N = N,
                         rho = rho, par2 = par2,
                         set.sd = set.sd)
  data[, 6] <- me[[1]]
  misclas <- me[[2]]
```

```
  }

  # create the binary target variable based on a
  # binomial distribution with probability pr
  data$label <- factor(sapply(1:N,
                              function(i) rbinom(1, 1, pr[i])))

  # data with predictors the 4 rules
  if (is.null(rho)) {
    dat <- data.frame(F1 = factor(ifelse(F1 == T, 1, 0)),
                      F2 = factor(ifelse(F2 == T, 1, 0)),
                      F3 = factor(ifelse(F3 == T, 1, 0)),
                      F4 = factor(ifelse(F4 == T, 1, 0)),
                      label = data$label)
  } else {
    dat <- NULL
  }
  return(list(data = data, data.LR = dat, sp=sp, misclas = misclas))
}



MeasurementError <- function(X.mat, N, rho, par2, set.sd, x.me = 6) {
  # Adding measurement error in predictor X6
  #
  # Args:
```

```
#   X.mat: a matrix with all the predictors
#   N: sample size of data set
#   rho: the reliability coefficient, determines the sizes of
#        measurement error; for set.sd == 4, if rho=1, then
#        misclassification is around 13%, if rho=2, then is
#        around 25%; for set.sd != 4, rho can be numeric
#        with 0 < rho < 1
#   par2: a vector of probabilities for set.sd=4, or one sd
#         for set.sd=1, or a vector of sd's for set.sd=2,
#         or a covariance matrix for set.sd = 3
#   set.sd: type of predictor variables in the data,
#           1 - normally distributed with common sd;
#           2 - normally distributed with different sd's;
#           3 - normally distributed given a covariance matrix;
#           4 - binomially distributed with different probabilities
#   x.me: the 6-th predictor is chosen to add measurement error
#
# Returns:
#   a list with a vector of the values of 6-th predictor with
#   measurement error, and if set.sd = 4, also the number of
#   misclassifications occured in X6

X <- X.mat[, x.me]

if(set.sd == 4) {
  q <- c(1.5, 1.15)[rho]
```

```
    e <- rnorm(N)
    U <- as.numeric((e > q) | (e < -q))
    W <- X
    W[U==1 & X==1] <- 0
    W[U==1 & X==0] <- 1
    return(list(W, sum(U)))

  } else {
    if(set.sd == 1) sd.X <- par2
    if(set.sd == 2) sd.X <- par2[6]
    if(set.sd == 3) sd.X <- sqrt(par2[6, 6])

    sigma.u <- sd.X * sqrt( (1 - rho) / rho)
    U <- round(rnorm(N, 0, sigma.u), 2)
    W <- X + U
    return(W)
  }
}


LogisticRegressionCV <- function(dt, k = 10) {
  # Given a dataset, the accuracy and the AUC score were obtained
  # through a 10-fold cross-validated Logistic Regression model,
  # i.e., the binary class variable and the four rules that were
  # obtained by a given dataset are used to build a logistic
  # regression model.
```

```
  #
  # Args:
  #   dt: data with a target outcome variable and the 4 rules
  #       as predictors
  #   k: number of fold of cross-validation
  #
  # Returns:
  #   the accuracy and AUC score of the model

  # define training control
  train_control<- trainControl(method = "cv",
                               number = 10, savePredictions = T)
  # train the model
  model<- train(label ~ ., data = dt,
                method = "glm", family = binomial,
                trControl = train_control)

  y <- dt$label  # true labels
  fitpred <- ifelse(model$finalModel$fitted.values > 0.5, 1, 0)
  roc_obj <- roc(y, fitpred)

  return(list(Accuracy = model$results$Accuracy,
              AUC = as.numeric(auc(roc_obj))))

}
```

```
Oracle <- function(N, balanced) {
  # "Oracle" simulations are performed for all the data combinations
  #  of sample sizes and data types; The true underlying accuracy/AUC
  #  score was obtained from the same model on which the data generation
  #  was based. Hence, logistic regression was fitted, and the
  #  accuracy/AUC score was obtained by 10-fold cross-validation.
  #
  # Args:
  #   N: sample size of data set
  #   balanced: whether the classes of the simulated binary target
  #             variable are balanced or not
  #
  # Returns:
  #   the accuracy, AUC score and the corresponding SDs across the
  #   100 fitted models for each design factor combination

  # results are saved in a dataframe for each data type and sample size
  res_acc <- res_acc_sd <- res_auc <- res_auc_sd <- data.frame()

  # a loop for data type
  for(set.sd.n in 1:4) {

    sp1 <- NULL
    if (set.sd.n == 4) {
      Z <- 5  # for binary predictors 5 sample sizes were explored
    } else {
```

```
  Z <- 3  # for continuous only 3
}
# a loop for sample size
for(j in 1:Z) {
  N.n <- N[j]

  ch <- acc <- auc <- numeric(100)
  set.seed(1278)

  par <- InitPar(N = N.n,
                 set.sd = set.sd.n,
                 balanced = balanced)
  for (i in 1:100) {
    data <- SimData(N = N.n, mean = par$mean,
                    par2 = par$par2, alpha = par$alpha,
                    set.sd = set.sd.n, sp = sp1)
    # for each data type and sample size, the same split point
    # vector is selected for all 100 replications
    if(i == 1 & j == 1 & set.sd.n != 4) sp1 <- data$sp


    # test set is not used here, just to get the same dataset
    # as in PreFit() function
    test.set <- SimData(N = 200, mean = par$mean,
                        par2 = par$par2, alpha = par$alpha,
                        set.sd = set.sd.n, sp = sp1)$data
```

```
      try(oracle_acc <- LogisticRegressionCV(dt = data$data.LR))
      acc[i] <- oracle_acc$Accuracy
      auc[i] <- oracle_acc$AUC
    }


    res_acc <- rbind(res_acc,
                     data.frame(N = N.n,
                                type = set.sd.n,
                                true_accuracy = mean(acc)))
    res_acc_sd <- rbind(res_acc_sd,
                        data.frame(N = N.n,
                                   type = set.sd.n,
                                   sd_accuracy = sd(acc)))
    res_auc <- rbind(res_auc,
                     data.frame(N = N.n,
                                type = set.sd.n,
                                true_AUC = mean(auc)))
    res_auc_sd <- rbind(res_auc_sd,
                        data.frame(N = N.n,
                                   type = set.sd.n,
                                   sd_AUC = sd(auc)))
  }
}
```

```
return(list(acc = res_acc,
            acc_sd = res_acc_sd,
            auc = res_auc,
            auc_sd = res_auc_sd))
}



N <- c(200, 500, 1000, 1500, 2000)

res.balanced <- Oracle(N, balanced = TRUE)
res.unbalanced <- Oracle(N, balanced = FALSE)

true_accuracy <- rbind(cbind(bal = 1, res.balanced$acc),
                       cbind(bal = 2, res.unbalanced$acc))
true_auc <- rbind(cbind(bal = 1, res.balanced$auc),
                  cbind(bal = 2, res.unbalanced$auc))
```

```
EvaluatePreModel <- function(str, j, match, corX, nrule, set.sd,
                             sdsp = NULL, sp  = NULL, print){
  # Compares the rules of a pre model with the theoretical ones,
  # both for binary and continuous predictors.
  #
  # Args:
  #   str: a character vector with all the rules that the pre model
  #        output from the iteration step j-1
  #   j: indicates the iteration step
  #   match: a matrix (k x 5) that contains for up until replication
  #          j-1 how many times each of the 4 specified rules is
  #          selected by the model; the 5-th column contains the
  #          total number of derived from the model
  #   corX: a matrix (k x 10) that contains for up until replication
  #         j-1 how many times each predictor is selected by the model
  #   nrule: number of derived rules from the pre model
  #   set.sd: type of predictor variables in the data,
  #           1 - normally distributed with common sd;
  #           2 - normally distributed with different sd's;
  #           3 - normally distributed given a covariance matrix;
  #           4 - binomially distributed with different probabilities
  #   sdsp: a vector with the sd of each predictor,
  #         if applicable is specified
```

```
#   sp: a vector with the sampled split points that correspond to
#       the predictors (the 11-th element indicates a second
#       split point for X6 as it is used twice in the rules);
#
# Returns:
#   a list with corX: a matrix (k x 10) that contains for up until
#                     replication j how many times each predictor
#                     is selected by the model
#               match: a matrix (k x 5) that contains for up until
#                      replication j how many times each of the 4
#                      specified rules is selected by the model; the
#                      5-th column contains the total number of
#                      derivedfrom the model


# Extracting decimal numbers from a string, one output rule
# The concern about negative numbers can be address with
# optional perl style
str1 <- regmatches(str,
                   gregexpr("(?>-)*[[:digit:]]+\\.*[[:digit:]]*",
                            str, perl=TRUE))

# extract < or > from output rules
spp1 <- regmatches(str, gregexpr("[<|>]", str, perl = TRUE))

# to count one time each rule (duplicate or multiplicate are not)
```

```
j1 <- j2 <- j3 <- j4 <- 0

# for each output (obtained from each j), i goes through each rule
# to identify which predictor, split and inequality symbol is used
for (i in seq(nrule)) {

  # i-th rule with predictor (1-10) and split value
  vec <- as.numeric(str1[[i]])
  indx <- c(1,3,5,7)[seq(spp1[[i]])]
  # using odd index values to extract predictors of the i-th rule
  var <- vec[indx]

  if(length(var) > 2) next

  # indentify and count predictors in i-th rule
  corX[j, var] <- corX[j, var] + 1
  # '<' becomes l(ess) and '>' g(reater)
  spp2 <- ifelse(unlist(spp1[[i]]) == "<", "l", "g")

  if (set.sd != 4) {
    split <- vec[-indx] # extract split values

     # if i-th rule matches the theoretical one
    if(all(c(6, 8) %in% var)) {
      if(all(spp2 %in% c("g", "g")) &
         (sp[6]-sdsp[6] <= split[var==6]) &
```

```
      (split[var==6] <= sp[6]+sdsp[6]) &
      (sp[8]-sdsp[8] <= split[var==8]) &
      (split[var==8] <= sp[8]+sdsp[8]) & j1 == 0) {
    match[j, 1] <- match[j, 1] + 1
    j1 <- 1  # to not be counted twice
    if(print) print(c(j,var))
  }
}

if(all(c(5, 7) %in% var)){
  if(all(spp2 %in% c("l", "l")) &
      (sp[5]-sdsp[5] <= split[var==5]) &
      (split[var==5] <= sp[5]+sdsp[5]) &
      (sp[7]-sdsp[7] <= split[var==7]) &
      (split[var==7] <= sp[7]+sdsp[7]) & j2 == 0) {
    match[j, 2] <- match[j, 2] + 1
    j2 <- 1
    if(print) print(c(j,var))
  }
}

if(all(c(3, 10) %in% var)) {
  if(all(spp2 %in% c("g", "g")) &
      (sp[3]-sdsp[3 ]<= split[var==3]) &
      (split[var==3] <= sp[3]+sdsp[3]) &
      (sp[10]-sdsp[10] <= split[var==10]) &
```

```
          (split[var==10] <= sp[10]+sdsp[10]) & j3 == 0) {
        match[j, 3] <- match[j, 3] + 1
        j3 <-  1
        if(print) print(c(j,var))
      }
    }

    if(all(c(9, 6) %in% var)) {
      if(all(spp2 %in% c("l", "l")) &
          (sp[9] - sdsp[9] <= split[var==9]) &
          (split[var==9] <= sp[9] + sdsp[9]) &
          (sp[11] - sdsp[6] <= split[var==6]) &
          (split[var==6] <= sp[11] + sdsp[6]) & j4 == 0) {
        match[j, 4] <- match[j, 4] + 1
        j4 <- 1
        if(print) print(c(j,var))
      }
    }

  } else {

    if(all(c(6, 8) %in% var)) {
      if(all(spp2 %in% c("l", "l")) &  j1 == 0) {
        match[j, 1] <- match[j, 1] + 1
        j1 <- 1
        if(print) print(c(j,var))
```

```
  }
}

if(all(c(5, 7) %in% var)){
  if(all(spp2 %in% c("g", "g")) &  j2 == 0) {
    match[j, 2] <- match[j, 2] + 1
    j2 <- 1
    if(print) print(c(j,var))
  }
}

if(all(c(3, 10) %in% var)) {
  if(spp2[var==3] == "l" & spp2[var==10] =="g" & j3 == 0) {
    match[j, 3] <- match[j, 3] + 1
    j3 <- 1
    if(print) print(c(j,var))
  }
}

if(all(c(9, 6) %in% var)) {
  if(spp2[var==6] == "g" & spp2[var==9] =="l" & j4 == 0) {
    match[j, 4] <- match[j, 4] + 1
    j4 <- 1
    if(print) print(c(j,var))
  }
}
```

```
    }
  }
  return(list(corX = corX, match = match))
}



PreFit <- function(N, set.sd, rho = NULL, k = 100, p = 10,
                   balanced = TRUE, print = TRUE) {
  # Fitting the prediction rule ensemble method for a given
  # data set, evaluated over 100 replications
  #
  # Args:
  #   N: sample size of data
  #   set.sd: type of predictor variables in the data,
  #           1 - normally distributed with common sd;
  #           2 - normally distributed with different sd's;
  #           3 - normally distributed given a covariance matrix;
  #           4 - binomially distributed with different probabilities
  #   rho: the reliability coefficient, determines the sizes of
  #        measurement error; for set.sd == 4, if rho=1, then
  #        misclassification is around 13%, if rho=2, then is around
  #        25%; for set.sd != 4, rho can be numeric with 0 < rho < 1
  #   k: number of replications
  #   p: number of predictors (default is 10)
  #   balanced: whether the classes of the simulated binary target
```

```
#             variable are balanced or not
#   print: which rules are selected; a vector with the iteration
#          number and the indices of the selected variables
#
# Returns:
#   a list with predictors: a matrix (k x 10) that contains for
#                           each replication k how many times
#                           each predictor is selected by the
#                           model
#               rules: a matrix (k x 5) that contains for each
#                      replication k how many times each of the
#                      4 specified rules is selected by the
#                      model; the 5-th column contains the
#                      total number of derived from the model
#               mean_error: mean cv error per k, sd_error: the
#                           sd of the mean cv error per k,
#               alpha: the specified parameter vector for each
#                      of the 4 ensemble members
#               check: returns 1, if each split point belongs to
#                      the domain of the corresponding predictor,
#                      otherwise 0; If measurement error was added
#                      to the binary predictor, then the number
#                      of misclassification was returned as
#                     "misclas.error" instead of "check".

par <- InitPar(N = N, set.sd = set.sd, balanced = balanced)
```

```
# if correct rules are selected & total number of rules
match <- matrix(0, k, 5)
# if correct predictor is selected
corX <- matrix(0, k, p)
# mean & sd cv error
misclas.all <- check <- sderror <- merror <- numeric(k)
accuracy <- AUC <- numeric(k)

sp <- sdsp <- NULL
rm <- NULL


for (j in 1:k){

  data.all <- SimData(N = N, mean = par$mean,
                      par2 = par$par2, alpha = par$alpha,
                      set.sd = set.sd, sp = sp, rho = rho)
  if (!is.null(rho) & set.sd == 4) misclas.all[j] <- data.all$misclas
  data <- data.all$data
  # split points based on the 1st data,
  # for the rest the same values were used

  if (set.sd != 4){
    if (j == 1)  sp <- data.all$sp

    X.mat <- data[, -11]
```

```
  check[j] <- all((apply(X.mat, 2, min) <= sp[-11]) &
                  (apply(X.mat, 2, max) >= sp[-11]) &
                  min(X.mat[, 6])<= sp[11] &
                  sp[11] <= max(X.mat[, 6]))
  sdsp <- apply(X.mat, 2, sd) # sd of each predictor
}

## test set
test.set <- SimData(N = 200, mean = par$mean,
                    par2 = par$par2, alpha = par$alpha,
                    set.sd = set.sd, sp = sp, rho = rho)$data



# Fit a prediction rule ensemble
fitpre <- NULL
try(fitpre <- pre(label ~ ., data = data,
                  type = "rules", type.measure = "deviance"))

pred_prob <- predict(fitpre, newdata = test.set,
                     type = 'response',
                     penalty.par.val = "lambda.1se")

predictions <- ifelse(pred_prob > 0.5, 1, 0)
y <- test.set$label  # true labels
accuracy[j] <- mean(y == predictions)
```

```
roc_obj <- roc(y, predictions)
AUC[j] <- as.numeric(auc(roc_obj))

# In which iteration the prediction rule ensemble method failed
if (is.null(fitpre)) {
  rm <- c(rm, j)
  next
}

# find mean cv error
ind <- which(fitpre$glmnet.fit$lambda ==
               fitpre$glmnet.fit$lambda.1se)
merror[j] <- fitpre$glmnet.fit$cvm[ind]
sderror[j] <- fitpre$glmnet.fit$cvsd[ind]

z <- coef(fitpre, penalty.par.val = "lambda.1se") # output
z <- na.omit(z[z$coefficient != 0, ]) # exclude intercept
match[j, 5] <- nrule <- nrow(z) # number or rules
if(nrule == 0) next # if output contains no rule, then next j
str <- z[,3] # all the output rules

out <- EvaluatePreModel(str = str, j = j, match = match,
                        corX = corX, nrule = nrule,
                        set.sd = set.sd, sdsp = sdsp,
                        sp = sp, print = print)
corX <- out$corX
```

```
    match <- out$match
  }

  colnames(corX) <- paste0("X", 1:p)
  colnames(match) <- c(paste0("rule", 1:4), 'N_rule')

  if(!is.null(rho) & set.sd == 4) {
    if (!is.null(rm)) {
      list <- list(predictors = corX[-rm, ], rules = match[-rm, ],
                   mean_error = merror[-rm], sd_error = sderror[-rm],
                   accuracy = accuracy[-rm], AUC = AUC[-rm],
                   alpha = par$alpha, misclas.error = misclas.all[-rm])
    } else {
      list <- list(predictors = corX, rules = match,
                   mean_error = merror, sd_error = sderror,
                   accuracy = accuracy, AUC = AUC,
                   alpha = par$alpha, misclas.error = misclas.all)
    }

  } else {
    if(!is.null(rm)){
      list <- list(predictors = corX[-rm, ], rules = match[-rm, ],
                   mean_error = merror[-rm], sd_error = sderror[-rm],
                   accuracy = accuracy[-rm], AUC = AUC[-rm],
                   alpha = par$alpha, check = check[-rm])
    } else {
```

```
      list <- list(predictors = corX, rules = match,
                   mean_error = merror, sd_error = sderror,
                   accuracy = accuracy, AUC = AUC,
                   alpha = par$alpha, check = check)
    }

  }
  return(list)
}
```

```
TypeOfError <- function(x, type) {
  # Calculates Type I or II error from the output of the PreFit
  # function using the matrix with the predictors that are
  # selected in each replication. Type I error was calculated
  # by counting, for each repetition k, if the variables X_1, X_2,
  # and X_4 were included in the derived rules, while they are
  # not part of the true rules. The type II error was calculated
  # by counting, if the rest of the variables were not included
  # in the derived rules, whereas they are part of
  # the true rules.
  #
  # Args:
```

```
  #   x: the output of the PreFit function
  #   type: 1 for type I error, or 2 for type II error
  #
  # Returns:
  #   a vector of Type I errors for the variables X_1, X_2, X_4,
  #   or a vector of Type II errors for the rest of the variables

   if (type == 1) {
    y <- x$predictors[, c(1,2,4)]
    return(apply(y, 2, function(x){sum(x != 0)}) / nrow(y))

  } else {
    y <- x$predictors[, c(3, 5:10)]
    return(apply(y, 2, function(x){sum(x == 0)}) / nrow(y))
  }
}




# run simulations for the continuous predictors
for(N in c(200, 500, 1000)) {
  for(set.sd in 1:3){
    for(i in 1:4) {
      rho <- list(NULL, 0.9, 0.7, 0.5)[[i]]
```

```
      set.seed(1278)
      x <- PreFit(N = N, set.sd = set.sd,
                  rho = rho, print = FALSE)
      assign(paste0("bal.", N, ".", set.sd,
                    c("", ".9", ".7", ".5")[i]), x)

      set.seed(1278)
      x <- PreFit(N = N, set.sd = set.sd,
                  rho = rho, balanced = FALSE, print = FALSE)
      assign(paste0("unbal.", N, ".", set.sd,
                    c("", ".9", ".7", ".5")[i]), x)
    }
  }
}


# run simulations for the binary predictors
for(N in c(200, 500, 1000, 1500, 2000)) {

  for(i in 1:3) {
    rho <- list(NULL, 1, 2)[[i]]

    set.seed(1278)
    x <- PreFit(N = N, set.sd = 4,
                rho = rho, print = FALSE)
```

```
    assign(paste0("bal.", N, ".", 4,
                  c("", ".1", ".2")[i]), x)


    set.seed(1278)
    x <- PreFit(N = N, set.sd = 4,
                rho = rho, balanced = FALSE, print = FALSE)
    assign(paste0("unbal.", N, ".", 4,
                  c("", ".1", ".2")[i]), x)
  }
}
```

Bibliography

Aigner, D. J. (1973). Regression with a binary independent variable subject to errors of observation. *Journal of Econometrics*, *1*(1), 49–59. Retrieved from https://econpapers.repec.org/article/eeeeconom/v_3a1_3ay_3a1973_3ai_3a1_3ap_3a49-59.htm

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. Retrieved from http://link.springer.com/10.1023/A:1018054314350 doi: 10.1023/A:1018054314350

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees.*

Buonaccorsi, J. P. (2010). *Measurement error: models, methods, and applications.* CRC Press.

Burez, J., & Van den Poel, D. (2009). Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, *36*(3), 4626–4636. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417408002121 doi: 10.1016/J.ESWA.2008.05.027

Carroll, R. J. (2005). Measurement Error in Epidemiologic Studies. In *Encyclopedia of biostatistics.* Chichester, UK: John Wiley & Sons, Ltd. Retrieved from http://doi.wiley.com/10.1002/0470011815.b2a03082 doi: 10.1002/0470011815.b2a03082

Carroll, R. J., Ruppert, D., Stefanski, L. A., & Crainiceanu, C. M. (2006). *Measurement error in nonlinear models: a modern perspective.* (Vol. 105). Chapman and Hall/CRC. Retrieved from https://www.taylorfrancis.com/books/9781420010138 doi: 10.1201/9781420010138

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale N.J.: L. Erlbaum Associates. Retrieved from http://www.worldcat.org/title/statistical-power-analysis-for-the-behavioral-sciences/oclc/17877467

Dhami, M. K. (2003). Psychological models of professional decision making. *Psychological Science*, *14*(2), 175–180. Retrieved from http://journals.sagepub.com/doi/10.1111/1467-9280.01438 doi: 10.1111/1467-9280.01438

Fokkema, M. (2017). *pre: An R Package for Fitting Prediction Rule Ensembles.* (Tech. Rep.). Retrieved from https://arxiv.org/pdf/1707.07149.pdf

Fokkema, M., & Christoffersen, B. (2018). pre: Prediction rule ensembles [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=pre (R package version 0.6.0)

Fokkema, M., Smits, N., Kelderman, H., & Penninx, B. W. J. H. (2015). Connecting clinical and actuarial prediction with rule-based

methods. *Psychological Assessment*, *27*(2), 636–644. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/25642935http://doi.apa.org/getdoi.cfm?doi=10.1037/pas0000072 doi: 10.1037/pas0000072

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software*, *33*(1), 1–22. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/20808728http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2929880

Friedman, J., & Popescu, B. E. (2003). *Importance Sampled Learning Ensembles* (Tech. Rep.). Retrieved from https://pdfs.semanticscholar.org/966f/fe536f84efd15c1379dad9adffe90b20676f.pdf

Friedman, J., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *Annals of Applied Statistics*, *2*(3), 916–954. Retrieved from http://projecteuclid.org/euclid.aoas/1223908046 doi: 10.1214/07-AOAS148

Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, *103*(4), 650–669. Retrieved from http://doi.apa.org/getdoi.cfm?doi=10.1037/0033-295X.103.4.650 doi: 10.1037/0033-295X.103.4.650

Green, L., & Mehr, D. (1997). What alters physicians' decisions to admit to the coronary care unit? , *45*(3), 219–226. Retrieved from http://web.missouri.edu/~segerti/capstone/CCUdecisions.pdf

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction* (2nd ed.). New York: Springer.

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674. Retrieved from http://statmath.wu-wien.ac.at/~zeileis/papers/Hothorn+Hornik+Zeileis-2006.pdf doi: 10.1198/106186006X133933

Hothorn, T., & Zeileis, A. (2015). *partykit: A Modular Toolkit for Recursive Partytioning in R* (Vol. 16; Tech. Rep.). Retrieved from http://cran.r-project.org/package=

Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., ... Hunt, T. (2018). *Package 'caret' Title Classification and Regression Training Description Misc functions for training and plotting classification and regression models* (Tech. Rep.). Retrieved from https://cran.r-project.org/web/packages/caret/caret.pdf

McDonald, R. P. (1999). *Test Theory : a Unified Treatment.* Taylor & Francis. Retrieved from https://books.google.nl/books/about/Test_Theory.html?id=acww1KB6BdEC&redir_esc=y

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, *55*(1-2), 169–186. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231203004314 doi: 10.1016/S0925-2312(03)00431-4

Qin, B., Xia, Y., Prabhakar, S., & Tu, Y. (2009). A Rule-Based Classification Algorithm for Uncertain Data. In *2009 ieee 25th international conference on*

data engineering (pp. 1633–1640). IEEE. Retrieved from http://ieeexplore.ieee.org/document/4812586/ doi: 10.1109/ICDE.2009.164

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Richardson, J. T. (2011). Eta squared and partial eta squared as measures of effect size in educational research. *Educational Research Review*, *6*(2), 135–147. Retrieved from https://www.sciencedirect.com/science/article/pii/S1747938X11000029#bib0050 doi: 10.1016/J.EDUREV.2010.12.001

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1-2), 1–39. Retrieved from http://link.springer.com/10.1007/s10462-009-9124-7 doi: 10.1007/s10462-009-9124-7

Savoca, E. (2000). Measurement errors in binary regressors: An application to measuring the effects of specific psychiatric diseases on earnings. *Health Services and Outcomes Research Methodology*, *1*(2), 149–164. Retrieved from https://www4.stat.ncsu.edu/~stefanski/MEM_Reports_2011/MeasurementErrorsinBinaryRegressors_Art.pdf doi: 10.1023/A:1012541005920

Sexton, J., & Laake, P. (2007). Boosted Regression Trees with Errors in Variables. *Biometrics*, *63*, 586–592. Retrieved from https://pdfs.semanticscholar.org/317c/e4ea063eea1b4f3b39015a71c662d1d9c668.pdf doi: 10.1111/j.1541-0420.2006.00718.x

Shimokawa, T., Li, L., Yan, K., Kitamura, S., & Goto, M. (2014). *MODIFIED RULE ENSEMBLE METHOD FOR BINARY DATA AND ITS APPLICATIONS* (Vol. 41; Tech. Rep. No. 2). Retrieved from https://www.jstage.jst.go.jp/article/bhmk/41/2/41_225/_pdf/-char/en

Signorell, A. (2019). *DescTools: Tools for Descriptive Statistics* (Tech. Rep.). R package version 0.99.27. Retrieved from https://cran.r-project.org/package=DescTools

Strobl, C. (2008). *Statistical issues in machine learning: towards reliable split selection and variable importance measures*. Cuvillier. Retrieved from https://books.google.co.uk/books/about/Statistical_Issues_in_Machine_Learning.html?id=-ZLfY4m5IPkC&redir_esc=y

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, *14*(4), 323–48. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/19968396http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2927982 doi: 10.1037/a0016973

Sullivan, G. M., & Feinn, R. (2012). Using Effect Size—or Why the P Value Is Not Enough. *Journal of Graduate Medical Education*, *4*(3), 279–282. Retrieved from http://dx.doi.org/10.4300/JGME-D-12-00156.1http://www.jgme.org/doi/abs/10.4300/JGME-D-12-00156.1 doi: 10.4300/JGME-D-12-00156.1

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288. Retrieved from https://www.jstor.org/stable/2346178 doi: 10.2307/2346178

Tsang, A., Tsang, S., Kao, B., Yip, K. Y., Ho, W.-S., & Dan Lee, S. (2009). Title Decision trees for uncertain data Decision Trees for Uncertain Data. , 441–444. Retrieved from http://hdl.handle.net/10722/136223 doi: 10.1109/TKDE.2009.175

Zeileis, A., Hornik, K., & Wien, W. (2008). Model-based Recursive Partitioning Torsten Hothorn. Retrieved from https://eeecon.uibk.ac.at/~zeileis/papers/Zeileis+Hothorn+Hornik-2008.pdf doi: 10.1198/10618600SX319331