CDQM, Volume 15, Number 4, 2012, pp. 5-15

COMMUNICATIONS
IN
DEPENDABILITY AND
QUALITY
MANAGEMENT
An International Journal

UDC 004.032.26

Modeling Response Variables in Taguchi Design Parameters Using CART and Random Forest Based Systems

Adriana Villa M.[1]***, Andres Carrion***[2*] *and* ***Susana San Matías***[2]

[1] *Lisandro Alvarado University, Barquisismeto, Venezuela, E-Mail: avilla.@ucla.edu.ve*
[2] *Polytechnic University of Valencia, Camino de Vera s/n^o, 46022 Valencia, Spain*
E-Mail: acarrion@eio.upv.es

accepted October 26, 2012

Summary
Taguchi parameter design is a quality approach to design better products and processes, less sensitive to changes of the environmental and productive conditions. Robustness against changes in factors affecting processes is the key concept. Some recent papers have used a two steps methodology to improve parameter design. The first step determines the objective function using Artificial Neural Networks (ANN) to predict the value of the response variable when factors are in some specific levels (different to those included in the experiments). The second step looks for the optimal parameter combination. Our proposal here is centered in improving the first of these two steps, and consists in the development of new systems to model the response variable, based in Classification and Regression Trees (CART) and in Random Forest (RF), as an alternative to ANN and with the aim of creating a more robust strategy.

Key words: *Artificial neural networks (ANN), classification and regression trees (CART), random forests (RF), Taguchi experimental design.*

1. INTRODUCTION

Quality engineering is a disciple oriented to detect and prevent quality problems from early design and development steps, and to take care about problems related with production costs and market questions, covering the full cycle of product design-production-use. Genichi Taguchi proposed an approach to off-line quality control, emphasizing in obtaining quality from the design with a

[*] corresponding author

methodology based in Design of Experiments (DOE) to find the optimal design of product/process characteristics. "Parameter design" was the name under which these techniques were widely known and applied in industry.

The goal of parameter design method is to identify factors affecting some key quality characteristic, and with this information to establish the process parameters combination that produces the better output with the minimum sensibility to changes in the values of these process parameters (or conditions) (Chang, 2008). That is, the better robust solution.

This approach is more focused in practical criteria of cost reduction than in the application of orthodox statistical rules and methods. Among its limitations, this methodology has the problem of considering only discrete level for factors, even if the factor is continuous, to find the better levels for parameters. This implies that, if a good sensitivity is to be obtained, the number of factors' levels must increase, a the number of experiments to perform will be too high. The alternative to this great number of experiments is some lack of accuracy in the conclusions (Maghsoodloo et al., 2004).

Su and Chang (2000) present as alternative for improving parameter design with a two steps method combining ANN with Simulated Annealing (SA). The first step corresponds to the determination of the relationship between process inputs and output, which is the target function. The second step optimizes parameters level's combination. Basically with this methodology the search for the optimal conditions is conducted out of the experimented parameter values (the factor levels) for those continuous factors.

In this paper, we define the basis for an alternative to the first step of this methodology. Our proposal consists in using CART techniques to model and predict response variable values, in a more economic, clear and efficient way than the method based in ANN. As methods based in Trees produce instable predictors, we have contrasted the use of Cross-Validation, Bootstrap, Bagging and RF, to obtain a more robust procedure.

This paper is structured as follows: In section 2 we present a brief description of the theoretical basis for CART, Cross-Validation, Bootstrap, Bagging and RF, origin of this study. Section 3 presents the predictive schemes proposed. A case study is presented in section 4, to establish numerical comparisons with the ANN based method. And finally, section 5 presents the conclusions showing the efficiency of our methodology.

2. RELATED WORKS

2.1 Classification and Regression Trees (CART)

Classification and Regression Trees (CART) algorithm is based in a recursive partitioning of the predictor variables set (space X) in disjoint regions (nodes) and in the assignment of a class to each of the regions resulting from the segmentation process. Root node, representing the complete population, is divided in classes defined by a partition based in a predictive variable and the mean value of the response variable in each class is computed, generating new nodes that also will be divided by a predictive variable and again new mean values will be calculated. This recursive process os repeated until one of the stop conditions appears. Nodes that aren't divided are terminal nodes.

In each terminal region, response value y(t) and predictors d(x_n) are constant. Previously, in the construction of the regression trees, intermediate nodes classification ways are established, as well as the rule to identify when a node is terminal and the rule for assigning y(t) values in each terminal node (Breiman et al., 1984). The starting point to assign y(t) values in each terminal node is the reduction of the forecast mean square error, defined as:

$$R(d) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - d(x_n)\right)^2 .$$

Then, the mean of y(t) values, for all those cases (x_n, y_n) in node t, is considered as the value minimizing R(d). That is,

$$\bar{y}(t) = \frac{1}{N(t)}\sum_{x_n \in t}^{N} y_n ,$$

where N(t) represents the total of cases in node t.

With the aim of determining the moment to stop data partitioning, CART algorithm proposes to build a tree as big as possible, T0, stopping the splitting process only when some minimum node size (say 5) is reached. This large tree T0 is pruned by removing some branch. Finally, of all possible trees built, the one that provides the lowest error rate or classification cost is selected. But this criterion is problematic from the point of view of Supervised Learning, as if the chosen classifier goodness of fit is validated in a new data set whose response is known (test data set), then error rate or classification cost tend to be high, appearing over-fitting problems. Then, some cutting criteria for tree branches need to be defined, to minimize error rate and also to penalize too complex tree structures. Also good estimator for classification error must be used, as Test Sample Estimates and 1-SE rules, among others, allowing a good classifier selection criterion, optimum among all sub-trees. Breiman et al. (1984) suggest this one as the most appropriate since it is the least affected by the number of established test sets.

CART serve not only as a prediction model, but also as a performance standard for the observations pertaining each of the classes. This is useful for interpreting the structure linked to the data set. Nevertheless, information coming from the tree structure is affected by random variation caused, many times, by the definition of the training and test sets. Small changes in the composition of these sets may cause very different trees. As re-sampling techniques are a powerful tool to reduce sample variability, we have decided to combine techniques as Cross Validation, Bootstrap and Bagging with CART algorithms. Their most relevant aspects are presented in the following paragraphs.

Cross-Validation: Known usually as k-fold Cross Validation, under the concept of supervised learning, is used to evaluate and compare two or more algorithms in each iteration. To do that, data are divided in k subsets and in each iteration, two or more algorithms use k-1 data subsets to train one or more models. Once established the model, the last subset (called test data subset) is used to predict model values and to compare the different algorithms performance. Assigning value to k is responsibility of the analyst, but Breiman and Spector (1992) recommends to use k in the range 5 to 10 when the objective is to minimize learning variance.

Bootstrap: This technique was proposed by Efron and Tibshirani (1996) and is based in the Glivenko-Cantelli theorem, which establishes that there is an almost sure but asinthotic convergence among an unknown distribution F (for the variable of interest) and an empiric distribution F_n (calculated from the sample) when $n\rightarrow\infty$. Once the statistic of interest $\hat{\theta}$ is fixed, a random sample $\{x_1, x_2,\ldots,x_n\}$ is obtained and then, using Monte Carlo to warranty a random re-sampling a value of the random variable $\mathbf{X}$ is obtained. This process is repeated n times, with replacement, to obtain a data set that forms the "Bootstrap sample" $\{x_{1i}, x_{2i}, \ldots x_{ni}\}$, where i indicates the i-sime bootstrap sample. For each of these samples, the statistic of interest $\hat{\theta}$ is calculated: $\hat{\theta}^{i*} = \hat{\theta}\left\{x_1^*, x_2^*, ..., x_n^*\right\}$. Repeating this process a pre-established number of times, a number of estimates for the statistic of interest is obtained, forming the bootstrap sample distribution of $\hat{\theta}$, that will be used to compute different estimates of the statistical error.

Bagging: Breiman (1996) propones a method of assembling random trees, combining multiple predictors to reduce the variance associated with forecasts in regression and classification methods using bootstrap samples generated from a training set Ω_{train} of size n, prodicing m additional training sets of size n^* ($n^*<n$) by uniform sampling of the training set Ω_{train} without replacement. Then, model is adjusted to each bootstrap sample and the mean of forecasts is computed. The Bagging estimation is defined as:

$$\hat{f}_{\prime bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{\prime * b}\left(x_i^*\right)$$

where B is the number of bootstrap samples from Ω_{train} and $\hat{f}^{\prime * b}\left(x_i^*\right)$ is the fitting of the model to each sample b_i (i=1, …B).

2.2 Random Forest

This technique is based in the construction of forecast trees using Bootstrap and Bagging, which guarantees process stability. Each tree is constructed using bootstrap samples and a set of variables in randomly selected in each step. Trees aren’t pruned to obtain unbiased trees, and the random selection of variables guarantees low correlation among trees. Formally, RF is a collection of forecast trees $h(x; \theta_k)$, k = 1, …, K where x is the inputs vector of size p, associated to vector X, and θ_k is is a random, independent identically distributed vector (Breiman, 2001). The training set if formed by the independent n(p+1)-uplas (x_1, y_1), … (x_n, y_n), corresponding to a joint distribution (X,Y). As the interest of our proposal is centered in RF for regression, vector Y is formed by numeric response values, and the forecast is the unweighted mean:

$$\bar{h}(x) = \frac{1}{K}\sum_{k-1}^{K} h(x, \theta_k).$$

Thus, by the General Limit Theorem, when $k \to \infty$

$$E_{XY}\left(Y - \bar{h}(x)\right)^2 \to E_{XY}\left(Y - E_\theta h(X;\theta)^2\right),$$

where $E_{XY}\left(Y - E_\theta h(X;\theta)^2\right) = PE_{RF}^*$ represents the forecast error, and in consequence this expression avoids over fitting in RF.

3. PROPOSED APPROACH

This research proposes an alternative method to the proposal by Su and Chang (2000) first phase. We propose to combine data mining techniques, presented in section 2, and contrast them with ANN, to establish a robust procedure for determining the objective function to forecast the response under a set of parameters.

Our proposal is defined by a series of steps that finally will produce forecasts of response values for each set of control factor conditions. As starting phase, it is needed to adjust tree specific characteristics that will define re-sampling algorithms and RF parameters.

3.1 Regression tree based strategies (CART)

3.1.1 Selection and development of modeling algorithms

Below is presented the procedure, step by step, to analyze predictive algorithms. We have called this procedure as A_0, as it will help to select algorithm A_p (p=1, …, 4), more robust that will be compared later with the ANN based strategy.

First we have to adjust CART parameters, as they are the basis to develop the proposed predictive algorithms A_p. We determine the minimum number of observations in each intermediate node and in final nodes. These parameters will be referred to as misplit and minbucket, respectively. Finally, the complexity parameter (cp=α) must be adjusted. It will be estimated with cross validation in 10-fold. The objective of these adjustments is to design a complex tree with the minimum possible cost of the error rate.

Procedure A0: Analysis of CART based strategies.

Inicialization: p=1

Step 1: Adjust CART with the complete normalized sample to fix parameters minsplit, minbucket and cp. Obtain its RMSE, that will be identified as RMSE^{i}_{tree}.

Step 2: Train predictive algorithm A_p.

Step 3: Calculate RMSE_{Ap} for the forecasts obtained for the test set with A_p. If p<4 then p=p+1 and go back to step 3. Otherwise go to step 4.

Step 4: Obtain $A^{t}_{p} = \arg \min_{p=1,2,3,4} \left(\mathrm{RMSE}_{Ap}\right)$ where A^{t}_{p} is the algorithms that minimizes RMSE.

Predictive algorithms.

The step by step development of each predictive algorithm A_p (p=1,2,3,4) of the A_0 procedure is as follows:

A_1: PRED-T+CV. Algorithm combining CART with cross validation.
A_2: PRED-T+Boot. Algorithm combining CART with Bootstrap.
A_3: PRED-T+Bagg. Algorithm combining CART with Bagging.
A_4: PRED-RF. Random Forest based algorithm.

Remember that all these algorithms receive, as input, the same values for parameters minsplit, minbucket, cp and RMSE_{tree}, obtained in step 2 of A_0 procedure.

Algorithm A1: PRED-T+CV.

Initialization: Select the value for k, the number of subsets in which dataset will be splited. Go to step 1.

Step 1: Randomly split sample in k subsets of the same size. Identify them as {$fold_1$, …, $fold_k$}. Assign j=1 and go to step 2.

Step 2: Determine training set and test set for j, defined as $train_j$={$fold_1$, …, $fold_{j-1}$, …, $fold_k$}, $test_j$=$fold_j$. Go to step 3.

Step 3: Train CART model defined in step 1 of procedure A_0 with $train_j$. Obtain response forecasts for $test_j$ and go to step 4.

Step 4: Calculate $RMSE_{T+CV(j)}$ form responses predicted for testj. If $RMSE_{T+CV(j)} < RMSE_{tree}$ go to step 5. Otherwise go back to step 1 of procedure A_0 to obtain new values for parameters minsplit, minbucket, cp and $RMSE_{tree}$, assign j=1 and go to step 2.

Step 5: If j<k then j=j+1 and go back to step 2. Otherwise calculate:

$$RMSE_{T+CV} = \frac{\sum_{j=1}^{k} RMSE_{T+CV(j)}}{k}$$

and stop.

Algorithm A2: PRED-T+Boot.

Initialization: Select the value for k, the number of replicates with replacement from the original sample. Go to step 1.

Step 1: Randomly select with replacement the m r replicates $\{R_1, \ldots, R_m\}$ from the original sample and define resample complementary sets in the original sample $\{T_1, \ldots, T_m\}$. Assign j=1 and go to step 2.

Step 2: Train CART model defined in step 1 of procedure A_0 with R_j. Obtain response forecasts for T_j and go to step 3.

Step 3: Calculate $RMSE_{T+Boot(j)}$ from responses predicted for T_j. If $RMSE_{T+Boot(j)} < RMSE_{tree}$ go to step 4. Otherwise go back to step 1 of procedure A_0 to obtain new values for parameters minsplit, minbucket, cp and $RMSE_{tree}$, assign j=1 and go to step 2.

Step 4: If j<k then j=j+1 and go to step 2. Otherwise calculate:

$$RMSE_{T+CV} = \frac{\sum_{j=1}^{m} RMSE_{T+Boot(j)}}{m}$$

and stop.

Algorithm A3: PRED-T+Bagg.

Initialization: Select the value for m, the number of replicates and n*, the sample size of each replicates without replacement (m>n*). Go to step 1.

Step 1: Randomly divide the original sample in train and test. Randomly select without replacement the m replicates {R1, …, Rm} from the train set. Assign j=1 and go to step 2.

Step 2: Train CART model defined in step 1 of procedure A_0 with R_j. Obtain response forecasts for test set and go to step 3.

Step 3: Calculate $RMSE_{T+Bagg(j)}$ from responses predicted for T_j. If $RMSE_{T+Bagg(j)} < RMSE_{tree}$ go to step 4. Otherwise go back to step 1 of procedure A_0 to obtain new values for parameters minsplit, minbucket, cp and $RMSE_{tree}$, assign j=1 and go to step 2.

Step 4: If j<k then j=j+1 and go to step 2. Otherwise calculate:

$$RMSE_{T+CV} = \frac{\sum_{j=1}^{m} RMSE_{T+Bagg(j)}}{m}$$

and stop.

Algorithm A4: PRED-RF:

```
Initialization: Define values for RF parameters, as the number os trees to assemble (ntree) and the
                number of variables from the sample that will be candidate in each in each division
                (mtry). This parameter must be optimized in a preliminary study of the error rate
                OOB. Go to step 1.
Step 1: Randomly divide the original sample in train and test. Go to step 2.
Step 2: Train RF with train. Obtain response forecasts for test and go to step 3.
Step 3: Calculate $RMSE_{RF}$ from responses predicted for test set. If $RMSE_{RF} < RMSE_{tree}$ then stop.
```

Otherwise go back to step 1 of procedure A_0 to obtain new values for parameters minsplit, minbucket, cp and $RMSE_{tree}$, assign j=1 and go to step 2.

3.2 Artificial Neural Networks (ANN) Based Strategy

Su and Chang (2000) and Chang (2005) propose in their initial step to employ ANN in response forecasting. These authors ensure good adaptability of ANN to both quantitative and qualitative factors and a easy application to industrial engineering problems. For this reason we present the algorithm that they have proposed, to incorporate it to the comparative study.

We have named this algorithm as A_5:PRED-ANNxxx, where xxx is referred to the architecture provide to the lower value of RMSE.

Algorithm A5: PRED-ANNxxx.

```
Initialization: Consider network architecture xxx. Select n, the number of iterations (epochs). Go to
                step 1.
Step 1: Normalize data set with the sigmoid function and go to step 2.
Step 2: Divide the original sample in training subsets (train) and validation subsets (test). Go to step
        3.
Step 3: Adjust ANN under that Backpropagation scheme using train. Go to step 4.
Step 4: Calculate $RMSE_{ANN}^{train}$ and $RMSE_{ANN}^{test}$. Go to step 5.
Step 5: If applicable, select a new value n and go to step 3. Otherwise go back to step 6.
Step 6: Obtain architecture that minimizes RMSE for all values n.
```

4. NUMERICAL EXAMPLE

Table 1 shows the data from a study used by Su and Chang (2000) referred to a gas assisted injection molding process. Data are presented in a orthogonal matrix L_{18} with 8 control factors and five trials, where response y_i, (i=1, …, 5) is the length in the gas channel. Control factors are: mould temperature, melt temperature, injection speed, gas injection time, gas pressure, gas distance, gas delay time and constant pressure time, and they were denoted by A,B,C,D,E,F,G and H, respectively. This study attempts to make the response as small as possible by selecting parameter set values.

Data set has been previously normalized to establish scales homogeneity for ANN, and has been randomly divided in train and test subsets. Train is formed by 72 cases (80% of the data) and test subset is formed by 18 cases.

4.1 Control Parameters

CART based strategies:

- R language, package rpart 3.1-46.
- Minsplit=10, minbucket=2, cp=0.00001.

Algorithm PRED-T+CV:

- R language, package ipred 0.8-8.
- 10-fold, $I_j=n_{CV}$={1000,5000,10000,15000}.

Table 1. Responses and Control Factor values of the experiment

No.	Control factor								Responses				
	A	B	C	D	E	F	G	H	y_1	y_2	y_3	y_4	y_5
1	50	230	50	1	90	64	0	0	42	40	57	68	74
2	50	230	60	1.5	110	65	0.5	3	71	76	74	74	75
3	50	230	70	2	130	66	1	6	84	80	83	80	82
4	50	240	50	1	110	65	1	6	37	29	34	38	41
5	50	240	60	1.5	130	66	0	0	117	115	121	123	116
6	50	240	70	2	90	64	0.5	3	37	36	36	39	36
7	50	250	50	1.5	90	66	0.5	6	85	87	88	93	90
8	50	250	60	2	110	64	1	0	28	26	24	25	29
9	50	250	70	1	130	65	0	3	84	79	84	79	73
10	60	230	50	2	130	65	0.5	0	74	84	64	69	65
11	60	230	60	1	90	66	1	3	84	87	95	88	94
12	60	230	70	1.5	110	64	0	6	71	68	68	70	65
13	60	240	50	1.5	130	64	1	3	25	24	25	28	24
14	60	240	60	2	90	65	0	6	88	88	89	90	79
15	60	240	70	1	110	66	0.5	0	114	124	125	117	118
16	60	250	50	2	110	66	0	3	106	106	104	99	107
17	60	250	60	1	130	64	0.5	6	31	41	43	36	40
18	60	250	70	1.5	90	65	1	0	60	53	58	51	60

Algorithm PRED-T+Boot:

- R language, package ipred 0.8-8.
- $R_j=n_{Boot}$={1000,5000,10000,15000}.

Algorithm PRED-T+Bagg:

- R language, package ipred 0.8-8.
- $R_j=n_{Bagg}$={1000,5000,10000,15000}.

- RMSE estimation with out-of-bag (OOB).

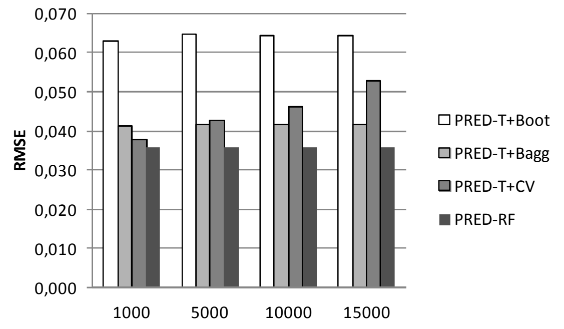
Algorithm PRED-RF:
- R language, package randomForest 4.5-33.
- n_{Tree} = {1000, 5000, 10000, 15000}, mtry = 4.
- RMSE estimation with out-of-bag (OOB).
- Trees are not pruned.

In algorithms PRED-T+VC, PRED-T+Boot, PRED-T+Bagg and PRED-RF the same control parameter (minsplit, minbucket and cp) considered in regressions trees (CART) were maintained.

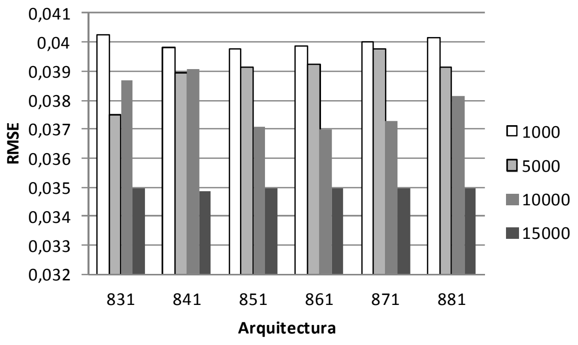
ANN based strategies:
- Software Qnet2000.
- Neural Networks in Backpropagation.
- Architectures: 8-3-1; 8-4-1; 8-5-1; 8-6-1; 8-7-1; 8-8-1.
- These architectures were used, respectively in the epochs {1000, 5000, 10000, 15000}.

4.2. <u>Predictive Algorithm Comparison</u>

Figure 1a shows RMSE for the different CART and RF based strategies. It can be noted that even considering the different number of iterations, RMSE are stable for all the strategies. The lower values of RMSE are obtained with PRED-T+Bagg and PRED-RF. Figure 1b shows RMSE values for ANN based strategies under all architectures considered. As can be noted, ANN improves their RMSE values as the number of iterations increases. Su and Chang (2000) identify 8-5-1 as the better architecture, and we have used this one for comparative purposes, termed as PRED-ANN851.



(a) CART and RF based strategies



(b) ANN based strategies

Figure 1. RMSE comparison

Figure 2 shows RMSE values reached by all the studied strategies. Observing the values corresponding to 1000 iterations it can be easily noted the inefficiency of PRED-ANN851 and PRED-T+Boot in front of the rest of algorithms. In opposition, for 15000 iterations is precisely PRED-ANN851 the option that reaches the best (lower) RMSE, but with values very close to those obtained by PRED-RF, which is stable for under any number of iterations. In what refers to PRED-T+Bagg and PRED-T+VC, their RMSE are stable for the different iteration numbers, but in all cases with values greater than those of PRED-RF.

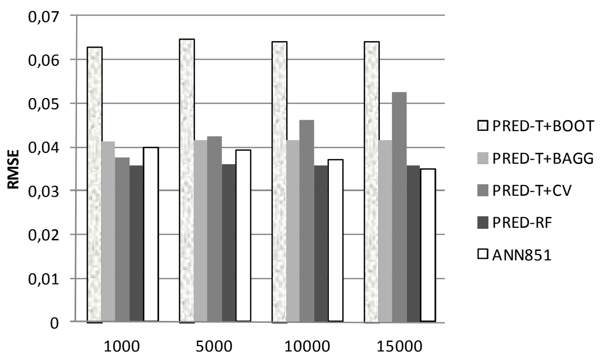


Figure 2. Comparison of RMSE for all strategies

PRED-ANN851 and PRED-RF are the algorithms with better performance in RMSE. We define P% as a measure of the improvement, in terms of RMSE reduction, that the RF algorithm use implies with reference to ANN851. Results are shown in Table 2, where with only 1000 iterations PRED-RF improves ANN results by 73.86%. It also can be verified the stability of RMSE values in PRED-RF for all iterations numbers, in opposition to what occurs in ANN851, which improves as the number of iterations increases, up to became better that PRED-RF for 15000 iterations.

Table 2. RMSE improvement (in %) of PRED-RF with respect to PRED-ANN851

		RMSE	
n	PRED-RF	PRED-ANN851	P%
1000	0.0135	0.0517	**73.86**
5000	0.0134	0.0158	15.45
10000	0.0134	0.0153	12.88
15000	0.0134	0.0099	**-36.14**

5. CONCLUSIONS

This paper presents new systems based in CART and RF to model the experimental response, as a robust alternative to the use of ANN. We have proved with an application case that our algorithms are more stable and robust in presence of different iteration numbers. Specifically, PRED-RF algorithm has obtained the better results for RMSE.

Our working scheme has three important properties:

- CART based strategies successfully combine Cross Validation, Bootstrap, Bagging and Random Forest techniques to reduce the lack of stability of CART algorithms.
- CART and RF based strategies can be used in presence of both continuous and discrete parameters, widening the potential application field.
- PRED-RF strategy guarantees model stability and avoids over-adjustment, by the combination of Bootstrap and Bagging.

REFERENCES

[1] Breiman, Leo (1996). Bagging Predictor. Macnine Learning, (26), pp. 123-140.
[2] Breiman, Leo (2001). Random Fores. Macnine Learning, (45), pp. 5-32.

[3] Breiman, Leo; Friedman, Jerome H.; Olsen, Richard A. and Stone, Charles J. (1984). Classication and Regression Trees. Chapman HALL/CRC.

[4] Breiman, Leo and Spector (1992). Submodel selection and evaluation in regression: the X-random case.International Journal of Systems Science, (3), pp. 291-319.

[5] Chang, Hsu-Hwa (2005). Applications of neural networks and genetic algorithms to taguchi's robust design. International journal of electronic busines management, 3(2), pp. 90-96.

[6] Chang, Hsu-Hwa (2008). A data mining approach to dynamic multiple responses in Taguchi experimental design. Expert Systems with Applications, (35), pp. 1095-1103.

[7] Efron, Bradley and Tibshirani, Rob (1996). Bootstrap Methods for Standard Error, Confidence Intervals and other measures of statistical accuracy. Statistical Science, 1(1), pp. 55-77.

[8] Hastie, Trevor; Tibshirani, Robert and Friedman, Jerome (2009). The elements of statistical learning. Data mining, inference and prediction. Springer series in statistic.

[9] Kohabi, Ron (1995). A study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection. In: International Joint Conference on Artificial Intelligence (IJCAI).

[10] Maghsoodloo, Saed; Ozdemir, Gultekin; Jordan, Victoria and Huang, Chen-Hsiu (2004). Strengths and limitations of Taguchi's contributions to quality, manufacturing and process engineering. Journal of Manufacturing Systems, 23(2), pp. 73-126.

[11] Su, Chao-Ton and Chang, Hsu-Hwa (2000). Optimization of parameter design: an intelligent approach using neural network and simulated annealing. International Journal of Systems Science, 31(12), pp.1543-1549.

[12] Sutton, Clifton D. (2005). Classification and regression tree, Bagging and Boosting. Handbook of Statistic, pp. 303-329.