*BU-CE-0012*

# AN EAGER REGRESSION METHOD BASED ON SELECTING APPROPRIATE FEATURES*

Tolga Aydın and H.Altay Güvenir

Department of Computer Engineering
Bilkent University
Ankara, 06533, TURKEY

**Abstract.** This paper describes a machine learning method, called *Regression by Selecting Best Features* (RSBF). RSBF consists of two phases: The first phase aims to find the predictive power of each feature attribute by constructing simple linear regression lines, one per each continuous feature and number of categories per each categorical feature. Although the predictive power of a continuous feature is constant, it varies for each distinct value of categorical features. The second phase constructs multiple linear regression lines among continuous features, each time excluding the worst feature among the current set, and constructs multiple linear regression lines. Finally, these multiple linear regression lines and categorical features' simple linear regression lines are sorted according to their predictive power. In the querying phase of learning, the best linear regression line and the features constructing that line are selected to make predictions.
**Keywords:** Prediction, Feature Projection, Regression.

## 1 INTRODUCTION

Prediction has been one of the most common problems researched in data mining and machine learning. Predicting the values of categorical features is known as classification, whereas predicting the values of continuous features is known as regression. From this point of view, classification can be considered as a subcategory of regression. In machine learning, much research has been performed for classification. But, recently the focus of researchers has moved towards regression, since many of the real-life problems can be modeled as regression problems.

There are two different approaches for regression in machine learning community: Eager and lazy learning. Eager regression methods construct rigorous models by using the training data, and the prediction task is based on these models. The advantage of eager regression methods is not only the ability to obtain the interpretation of the underlying data, but also the reduced query time. On the other hand, the main disadvantage is their long train time requirement. Lazy regression methods, on the other hand, do not construct models by using the training data. Instead, they delay all processing to prediction phase. The most important disadvantage of lazy regression methods is the fact that, they do not provide an

* This research is supported, in part, by TÜBİTAK (Scientific and Technical Research Council of Turkey) under Grant 198E015

interpretable model of the training data, because the model is usually the training data itself. It is not a compact description of the training data, when compared to the models constructed by eager regression methods, such as regression trees and rule based regression.

In the literature, many eager and lazy regression methods exist. Among eager regression methods, CART [1], RETIS [7], M5 [5], DART [2], and Stacked Regressions [9] induce regression trees, FORS [6] uses inductive logic programming for regression, RULE [3] induces regression rules, and MARS [8] constructs mathematical models. Among lazy regression methods, *k*NN [4, 10] is the most popular nonparametric instance-based approach.

In this paper, we describe an eager learning method, namely *Regression by Selecting Best Feature*s (RSBF). This method makes use of the linear least squares regression.

A preprocessing phase is required to increase the predictive power of the method. According to the Chebyshev's result [13], for any positive number $k$, at least $(1 - 1/k^2) * 100\%$ of the values in any population of numbers are within $k$ standard deviations of the mean. We find the standard deviation of the target values of the training data, and discard the training data whose target value is not within $k$ standard deviations of the mean target. Empiricaly, we reach the best prediction by taking $k$ as $\sqrt{2}$.

In the first phase, RSBF constructs projections of the training data on each feature, and this phase continues by constructing simple linear regression lines, one per each continuous feature and number of categories per each categorical feature. Then, the simple linear regression lines belonging to continuous features are sorted according to their prediction ability. The second phase begins by constructing multiple linear regression lines among continuous features, each time excluding the worst feature among the current set, and continues by constructing multiple linear regression lines. Then these multiple linear regression lines together with categorical features' simple linear regression lines are sorted according to their predictive power. In the querying phase of learning, the target value of a query instance is predicted using the linear regression line, multiple or simple, having the minimum relative error, i.e. having the maximum predictive power. If this linear regression line is not suitable for our query instance, we keep searching for the best linear regression line among the ordered list of linear regression lines.

In this paper, RSBF is compared with three eager (RULE, MARS, DART) and one lazy method (*k*NN) in terms of predictive power and computational complexity. RSBF is better not only in terms of predictive power but also in terms of computational complexity, when compared to these well-known methods. For most data mining or knowledge discovery applications, where very large databases are in concern, this is thought of a solution because of low computational complexity. Again RSBF is noted to be powerful in the presence missing feature values, target noise and irrelevant features.

In Section 2, we review the *k*NN, RULE, MARS and DART methods for regression. Section 3 gives a detailed description of the RSBF. Section 4 is devoted to the empirical evaluation of RSBF and its comparison with other methods. Finally, in Section 5, conclusions are presented.

## 2 REGRESSION OVERVIEW

*k*NN is the most commonly used lazy method for both classification and regression problems. The underlying idea behind the *k*NN method is that the closest instances to the query point have similar target values to the query. Hence, the *k*NN method first finds the closest instances to the query point in the instance space according to a distance measure. Generally, the Euclidean distance metric is used to measure the similarity between two points in the instance space. Therefore, by using Euclidean distance metric as our distance measure, *k* closest instances to the query point are found. Then *k*NN outputs the distance-weighted average of the target values of those closest instances as the prediction for that query instance.

In machine learning, inducing rules from a given train data is also popular. Weiss and Indurkhya adapted the rule-based classification algorithm [11], Swap-1, for regression. Swap-1 learns decision rules in Disjunctive Normal Form (DNF). Since Swap-1 is designed for the prediction of categorical features, using a preprocessing procedure, the numeric feature in regression to be predicted is transformed to a nominal one. For this transformation, the P-class algorithm is used [3]. If we let $\{y\}$ be a set of output values, this transformation can be regarded as a one-dimensional clustering of training instances on response variable $y$, in order to form classes. The purpose is to make $y$ values within one class similar, and across classes dissimilar. The assignment of these values to classes is done in such a way that the distance between each $y_i$ and its class mean must be minimum. After formation of pseudo-classes and the application of Swap-1, a pruning and optimization procedure can be applied to construct an optimum set of regression rules.

MARS [8] method partitions the training set into regions by splitting the features recursively into two regions, by constructing a binary regression tree. MARS is continuous at the borders of the partitioned regions. It is an eager, partitioning, interpretable and an adaptive method.

DART, also an eager method, is the latest regression tree induction program developed by Friedman [12]. It avoids limitations of disjoint partitioning, used for other tree-based regression methods, by constructing overlapping regions with increased training cost.

## 3 REGRESSION BY SELECTING BEST FEATURES (RSBF)

RSBF method tries to determine a subset of the features such that this subset consists of the best features. The next subsection describes the training phase for RSBF, then we describe the querying phase.

### 3.1 Training

Training in RSBF begins simply by storing the training data set as projections to each feature separately. A copy of the target values is associated with each projection and the training data set is sorted for each feature dimension according to their feature values. If a training instance includes missing values, it is not simply ignored as in many regression algorithms. Instead, that training instance is stored for the features on which its value is given. The next step involves

constructing the simple linear regression lines for each feature. This step differs for categorical and continuous features. In the case of continuous features, exactly one simple linear regression line per feature is constructed. On the other hand, the number of simple linear regression lines per each categorical feature is the number of distinct feature values at the feature of concern. For any categorical feature, the parametric form of any simple regression line is constant, and it is equal to the average target value of the training instances whose corresponding feature value is equal to that categorical value.

The training phase continues by constructing multiple linear regression lines among continuos features, each time excluding the worst one. Then these lines, together with categorical features' simple linear regression lines are sorted according to their predictive power. The training phase can be illustrated through an example.

Let our example domain consist of five features, $f_1, f_2, f_3, f_4$ and $f_5$, where $f_1$, $f_2$, and $f_3$ are continuous and $f_4$, $f_5$ are categorical. For categorical features, *No_categories* [$f$] is defined to give the number of distinct categories of feature $f$. In our example domain, let the following values be observed:

*No_categories* [$f_4$] = 2 (values: A, B)
*No_categories* [$f_5$] = 3 (values: X, Y, Z)

For this example domain, 8 simple linear regression lines are constructed: 1 for $f_1$, 1 for $f_2$, 1 for $f_3$, 2 for $f_4$, and finally 3 for $f_5$. Let the following be the parametric form of the simple linear regression lines:

Simple linear regression line for $f_1$: target = $2f_1 - 5$
Simple linear regression line for $f_2$: target = $-4f_2 + 7$
Simple linear regression line for $f_3$: target = $5f_3 + 1$
Simple linear regression line for A category of $f_4$: target = 6
Simple linear regression line for B category of $f_4$: target = -5
Simple linear regression line for X category of $f_5$: target = 10
Simple linear regression line for Y category of $f_5$: target = 1
Simple linear regression line for Z category of $f_5$: target = 12

The training phase continues by sorting the simple linear regression lines belonging to continuous features according to their predictive accuracy. The relative error (RE) of the regression lines is used as the indicator of predictive power: the smaller the RE, the stronger the predictive power. The RE of a simple linear regression line is computed by the following formula:

$$RE = \frac{MAD}{\frac{1}{Q}\sum_{i=1}^{Q}|t(q_i) - \bar{t}|}$$

where $Q$ is the number of training instances used to construct the simple linear regression line, $\bar{t}$ is the median of the target values of $Q$ training instances, $t(q_i)$ is the actual target value the $i^{th}$ training instance The MAD (Mean Absolute Distance) is defined as follows:

$$MAD = \frac{1}{Q}\sum_{i=1}^{Q}|t(q_i) - \hat{t}(q_i)|$$

Here, $\hat{t}(q_i)$ denotes the predicted target value of the $i^{th}$ training instance according to the induced simple linear regression line.

Let's assume that continuous features are sorted as $f_2$, $f_3$, $f_1$ according to their predictive power. The second step of training phase begins by employing multiple linear least squares regression on all 3 features. The output of this process is a multiple linear regression line involving contributions of all three features. This line is denoted by $MLRL_{1,2,3}$ . Then we exclude the worst feature, namely $f_1$, and run multiple linear least squares regression to obtain $MLRL_{2,3.}$ In the final step, we exclude the next worst feature of the current set, namely $f_3$, and obtain $MLRL_2$. Actually the multiple linear least squares regression transforms into simple linear least squares regression in the final step, since we deal with exactly one feature. Let the following be the parametric form of the multiple linear regression lines:

$MLRL_{1,2,3}$ : target = $-f_1 + 8f_2 + f_3 + 3$
$MLRL_{2,3}$ : target = $6f_2 + 6f_3 - 9$
$MLRL_2$ : target = $-4f_2 + 7$

The second phase of training is completed by sorting MLRLs together with categorical features' simple linear regression lines according to their predictive power, the smaller the RE of a regression line, the stronger the predictive power of that regression line.

Let's suppose that the linear regression lines are sorted in the following order, from the best predictive to the worst one:

$MLRL_{2,3} > f_4$=A $> MLRL_2 > f_5$=X $> MLRL_{1,2,3} > f_5$=Y $> f_5$=Z $> f_4$=B.

This shows that any categorical feature's predictive power may vary among its categories. For the above sorting schema, categorical feature $f_4$ 's predictions are reliable among its category A, although it is very poor among category B.

### 3.2 Querying

In order to predict the target value of a query instance $t_i$, the RSBF method uses exactly one linear regression line. This line may not always be the best one. The reason for this situation is explained via an example. Let the feature values of the query instance $t_i$ be as the following:

$$f_1(t_i) = 5, \quad f_2(t_i) = 10, \quad f_3(t_i) = \text{missing}, \quad f_4(t_i) = \text{B}, \quad f_5(t_i) = \text{missing}$$

Although the best linear regression line is $MLRL_{2,3}$ , this line can not be used for our $t_i$ , since $f_3(t_i)$ = missing. The next best linear regression line, which is worse than only $MLRL_{2,3}$ , is $f_4$=A. This line is also inappropriate for our $t_i$ , since, $f_4(t_i) \neq$ A. Therefore, the search for the best linear regression line, continues. The line

constructed by $f_2$ comes next. Since $f_2(t_i) \neq$ missing, we succeed in finding the best linear regression line. So the prediction made for target value of $t_i$ is (-4 * $f_2(t_i)$ + 7) = (-4 * 10 + 7) = -33. Once the appropriate linear regression line is found, remaining linear regression lines need not be dealed anymore.

## 4 EMPIRICAL EVALUATION

RSBF method was compared with the other well-known methods mentioned above, in terms of predictive accuracy and time complexity. We have used a repository consisting of 27 data files in our experiments. The characteristics of the data files are summarized in Table1. Most of these data files are used for the experimental analysis of function approximation techniques and for training and demonstration by machine learning and statistics community.

10 fold cross-validation technique was employed in the experiments. For lazy regression method *k* parameter was taken as 10, where *k* denotes the number of nearest neighbors considered around the query instance.

In terms of predictive accuracy, RSBF performed the best on 13 data files among the 27, and obtained the lowest mean relative error. (Table 2)

In terms of time complexity, RSBF performed the best in the total (training + querying) execution time, and became the fastest method. (Table 3, 4)

In machine learning, it is very important for an algorithm to still perform well when noise, missing featute value and irrelevant features are added to the system. Experimental results showed that RSBF was again the best method whenever we added %20 target noise, %20 missing feature value and 30 irrelevant features to the system. RSBF performed the best on 12 data files in the presence of %20 missing value, the best on 21 data files in the presence of %20 target noise and the best on 11 data files in the presence of 30 irrelevant features. (Table 5, 6, 7)

## 5 CONCLUSIONS

In this paper, we have presented an eager regression method based on selecting appropriate features. RSBF selects the best feature(s) and forms a parametric model for use in querying phase. This parametric model is either a multiple linear regression line involving the contribution of continuous features, or a simple linear regression line of a categorical value of any categorical feature. The multiple linear regression line reduces to a simple linear regression line, if exactly one continuous feature constructs the multiple linear regression line.

RSBF is better than other well-known eager and lazy regression methods in terms of prediction accuracy and computational complexity. It also enables the interpretation of the training data. That is, the method clearly states the most appropriate features that are powerful enough to determine the value of the target feature.

The robustness of any regression method can be determined by analyzing the predictive power of that method in the presence of target noise, irrelevant features and missing feature values. These three factors heavily exist in real life databases, and it is important for a learning algorithm to give promising results in the presence of those factors. Empirical results indicate that RSBF is also a robust method.

**Table1.** Characteristics of the data files used in the empirical evaluations. C: Continuous, N: Nominal

| Dataset | Original Name | Instances | Features (C+N) | Missing Values |
|---|---|---|---|---|
| **AB** | Abalone | 4177 | 8 (7 + 1) | None |
| **AI** | Airport | 135 | 4 (4 + 0) | None |
| **AU** | Auto-mpg | 398 | 7 (6 + 1) | 6 |
| **BA** | Baseball | 337 | 16 (16 + 0) | None |
| **BU** | Buying | 100 | 39 (39 + 0) | 27 |
| **CL** | College | 236 | 25 (25 + 0) | 381 |
| **CO** | Country | 122 | 20 (20 + 0) | 34 |
| **CP** | Cpu | 209 | 7(1 + 6) | None |
| **EL** | Electric | 240 | 12 (10 + 2) | 58 |
| **FA** | Fat | 252 | 17 (17 + 0) | None |
| **FI** | Fishcatch | 158 | 7 (6 + 1) | 87 |
| **FL** | Flare2 | 1066 | 10 (0 + 10) | None |
| **FR** | Fruitfly | 125 | 4 (3 + 1) | None |
| **GS** | Gss2 | 1500 | 43 (43 + 0) | 2918 |
| **HO** | Home Run Race | 163 | 19 (19 + 0) | None |
| **HU** | Housing | 506 | 13 (12 + 1) | None |
| **NO** | Normal Temp. | 130 | 2 (2 + 0) | None |
| **NR** | Northridge | 2929 | 10 (10 + 0) | None |
| **PL** | Plastic | 1650 | 2 (2 + 0) | None |
| **PO** | Poverty | 97 | 6 (5 + 1) | 6 |
| **RE** | Read | 681 | 25 (24 + 1) | 1097 |
| **SC** | Schools | 62 | 19 (19 + 0) | 1 |
| **SE** | Servo | 167 | 4 (0 + 4) | None |
| **ST** | Stock Prices | 950 | 9 (9 + 0) | None |
| **TE** | Televisions | 40 | 4 (4 + 0) | None |
| **UN** | Usnews Coll. | 1269 | 31 (31 + 0) | 7624 |
| **VL** | Villages | 766 | 32 (29 + 3) | 3986 |

**Table2**. Relative errors of algorithms ± standard deviation of 10 folds. Best REs are shown in bold font

| Dataset | RSBF | KNN | RULE | MARS | DART |
|---|---|---|---|---|---|
| **AB** | 0.678 ± 0.06 | **0.661** ± 0.07 | 0.899 ± 0.15 | 0.683 ± 0.17 | 0.678 ± 0.09 |
| **AI** | **0.532** ± 0.19 | 0.612 ± 0.25 | 0.744 ± 0.24 | 0.720 ± 0.58 | 0.546 ± 0.15 |
| **AU** | 0.413 ± 0.09 | **0.321** ± 0.10 | 0.451 ± 0.15 | 0.333 ± 0.10 | 0.346 ± 0.13 |
| **BA** | 0.570 ± 0.04 | **0.443** ± 0.05 | 0.666 ± 0.12 | 0.493 ± 0.06 | 0.508 ± 0.07 |
| **BU** | **0.732** ± 0.26 | 0.961 ± 0.09 | 0.946 ± 0.33 | 0.947 ± 0.38 | 0.896 ± 0.30 |
| **CL** | 1.554 ± 2.34 | 0.764 ± 0.33 | 0.290 ± 0.23 | 1.854 ± 4.51 | **0.252** ± 0.06 |
| **CO** | **1.469** ± 0.43 | 1.642 ± 0.66 | 6.307 ± 5.33 | 5.110 ± 3.93 | 1.695 ± 0.83 |
| **CP** | 0.606 ± 0.35 | 0.944 ± 0.57 | 0.678 ± 0.39 | 0.735 ± 0.35 | **0.510** ± 0.22 |
| **EL** | **1.020** ± 0.03 | 1.194 ± 0.16 | 1.528 ± 0.41 | 1.066 ± 0.09 | 1.118 ± 0.08 |
| **FA** | **0.177** ± 0.13 | 0.785 ± 0.08 | 0.820 ± 0.19 | 0.305 ± 0.29 | 0.638 ± 0.10 |
| **FI** | 0.638 ± 0.37 | 0.697 ± 0.51 | 0.355 ± 0.31 | **0.214** ± 0.18 | 0.415 ± 0.41 |
| **FL** | **1.434** ± 0.21 | 2.307 ± 0.71 | 1.792 ± 0.53 | 1.556 ± 0.32 | 1.695 ± 0.47 |
| **FR** | 1.013 ± 0.05 | 1.201 ± 0.23 | 1.558 ± 0.56 | **1.012** ± 0.05 | 1.077 ± 0.12 |
| **GS** | 0.461 ± 0.17 | 0.654 ± 0.08 | **0.218** ± 0.11 | 0.359 ± 0.10 | 0.410 ± 0.20 |
| **HO** | **0.707** ± 0.15 | 0.907 ± 0.17 | 0.890 ± 0.20 | 0.769 ± 0.13 | 0.986 ± 0.17 |
| **HU** | 0.589 ± 0.13 | 0.600 ± 0.12 | 0.641 ± 0.18 | 0.526 ± 0.17 | **0.522** ± 0.16 |
| **NO** | **0.977** ± 0.10 | 1.232 ± 0.31 | 1.250 ± 0.48 | 1.012 ± 0.07 | 1.112 ± 0.21 |
| **NR** | 0.938 + 0.15 | 1.034 + 0.16 | 1.217 + 0.24 | 0.928 + 0.17 | **0.873** + 0.23 |
| **PL** | 0.444 ± 0.02 | 0.475 ± 0.05 | 0.477 ± 0.05 | **0.404** ± 0.03 | 0.432 ± 0.03 |
| **PO** | 0.715 + 0.25 | 0.796 + 0.72 | 0.916 + 0.43 | 1.251 + 1.69 | **0.691** + 0.34 |
| **RE** | **1.001** ± 0.03 | 1.062 ± 0.05 | 1.352 ± 0.08 | 1.045 ± 0.05 | 1.189 ± 0.12 |
| **SC** | **0.175** ± 0.05 | 0.388 ± 0.13 | 0.341 ± 0.16 | 0.223 ± 0.19 | 0.352 ± 0.12 |
| **SE** | 0.868 ± 0.06 | 0.619 ± 0.16 | **0.229** ± 0.10 | 0.432 ± 0.13 | 0.337 ± 0.13 |
| **ST** | 1.101 ± 1.21 | **0.599** ± 0.46 | 0.906 ± 1.30 | 0.781 ± 0.64 | 0.754 ± 0.62 |
| **TE** | **1.175** + 0.78 | 1.895 + 2.56 | 4.195 + 9.44 | 7.203 + 12.8 | 2.690 + 2.92 |
| **UN** | **0.385** ± 0.03 | 0.480 ± 0.03 | 0.550 ± 0.05 | 0.412 ± 0.07 | 0.623 ± 0.08 |
| **VL** | **0.930** ± 0.08 | 1.017 ± 0.12 | 1.267 ± 0.18 | 1.138 ± 0.32 | 1.355 ± 0.27 |
| **Mean** | **0.789** | 0.900 | 1.166 | 1.167 | 0.841 |

**Table 3.** Train time of algorithms in milliseconds. Best results are shown in bold font

| Dataset | RSBF | KNN | RULE | MARS | DART |
|---|---|---|---|---|---|
| **AB** | 211.5 | **8.9** | 3219 | 10270 | 477775 |
| **AI** | 2 | **0** | 90.8 | 159.2 | 62 |
| **AU** | 12.5 | **0.6** | 248.9 | 570.5 | 1890.1 |
| **BA** | 35.3 | **0** | 181.8 | 915.1 | 3171.1 |
| **BU** | 45.5 | **0** | 67.1 | 761.7 | 794.4 |
| **CL** | 34.1 | **0.5** | 148.2 | 1274.3 | 717.6 |
| **CO** | 17.9 | **0.1** | 108.6 | 475.3 | 481 |
| **CP** | 6.3 | **0** | 52.7 | 575.3 | 286 |
| **EL** | 13.3 | **0.2** | 69.5 | 407.5 | 1017 |
| **FA** | 36.4 | **0** | 161.1 | 985 | 1773.9 |
| **FI** | 4.2 | **0** | 47.8 | 240.2 | 201.4 |
| **FL** | 40.8 | **3.5** | 108.8 | 667.2 | 971.4 |
| **FR** | 1 | **0** | 34.1 | 99.5 | 45.9 |
| **GS** | 691.1 | **13.5** | 862.8 | 10143.9 | 27266 |
| **HO** | 19 | **0** | 57.5 | 616.3 | 893.9 |
| **HU** | 36.3 | **1** | 264.9 | 1413.9 | 8119.7 |
| **NO** | 0.2 | **0** | 30.6 | 69.3 | 18.9 |
| **NR** | 189.9 | **7.4** | 3493 | 5709.9 | 87815 |
| **PL** | 13.7 | **0.2** | 175.3 | 824.8 | 10024.4 |
| **PO** | 2.1 | **0** | 40.9 | 127.3 | 44 |
| **RE** | 104.3 | **3** | 196 | 2744.6 | 33044.6 |
| **SC** | 8.1 | **0** | 45.3 | 260.8 | 84.4 |
| **SE** | 2.2 | **0** | 37 | 116.4 | 83.4 |
| **ST** | 57.1 | **1.4** | 365.1 | 2281.4 | 17346.4 |
| **TE** | 0.2 | **0** | 30.9 | 31.1 | 3.1 |
| **UN** | 245 | **7.4** | 2547.1 | 8435.2 | 168169 |
| **VL** | 136.8 | **4.4** | 513.6 | 3597.8 | 23405 |
| **Mean** | 72.84 | **1.9296** | 488.83 | 1991.61 | 32055.7 |

**Table 4**. Query time of algorithms in milliseconds. Best results are shown in bold font

| Dataset | RSBF | KNN | RULE | MARS | DART |
|---|---|---|---|---|---|
| **AB** | 21.3 | 6547 | 14433.1 | 7.9 | **6.1** |
| **AI** | 1 | 3.4 | 141.7 | **0** | **0** |
| **AU** | 2.1 | 64.5 | 462.2 | **0** | **0** |
| **BA** | 2.2 | 54.6 | 244.8 | **0** | **0** |
| **BU** | **0** | 11.6 | 32.1 | **0** | **0** |
| **CL** | 1 | 38.2 | 40.3 | 1 | **0** |
| **CO** | 0.3 | 8.4 | 98.4 | **0** | 0.1 |
| **CP** | 1 | 11.6 | 87.3 | **0** | **0** |
| **EL** | 1.6 | 21 | 117.5 | **0** | **0** |
| **FA** | 1.4 | 33.1 | 96.4 | **0** | **0** |
| **FI** | 1.2 | 7.9 | 48.8 | **0** | **0** |
| **FL** | 4.2 | 407.8 | 223.6 | 0.4 | **0** |
| **FR** | 0.1 | 2 | 45.4 | **0** | **0** |
| **GS** | 8.1 | 2699.7 | 312.3 | 2.7 | **1.7** |
| **HO** | 0.6 | 13.3 | 43 | **0** | **0** |
| **HU** | 3 | 107.8 | 410.5 | **0** | **0** |
| **NO** | **0** | 1.9 | 30.8 | **0** | **0** |
| **NR** | 12.6 | 3399.4 | 11326.8 | 4.7 | **1.75** |
| **PL** | 9.5 | 571.9 | 2192.7 | **0.2** | 1.2 |
| **PO** | **0** | 2.2 | 37.1 | **0** | **0** |
| **RE** | 3.2 | 265.6 | 627.2 | **0** | 1 |
| **SC** | 0.3 | 2 | 27.8 | 3.7 | **0** |
| **SE** | 0.1 | 4.2 | 49.1 | **0** | **0** |
| **ST** | 5.4 | 303.2 | 1090.9 | 0.1 | **0** |
| **TE** | **0** | **0** | 24 | **0** | **0** |
| **UN** | 7.3 | 1383.2 | 1877.3 | 7 | **2** |
| **VL** | 5.8 | 439 | 1118.2 | 0.3 | **0** |
| **Mean** | 3.456 | 607.574 | 1305.16 | 1.03704 | **0.513** |

**Table5**. Relative errors of algorithms ± standard deviation of 10 folds, where 20% missing feature value are added. Best REs are shown in bold font. (* means result isn't available due to singular variance/covariance matrix)

| Dataset | RSBF | KNN | RULE | MARS | DART |
|---|---|---|---|---|---|
| **AB** | 0.720 ± 0.05 | 0.750 ± 0.08 | 0.961 ± 0.18 | 0.748 ± 0.10 | **0.688** ± 0.08 |
| **AI** | **0.496** ± 0.17 | 0.726 ± 0.27 | 0.676 ± 0.30 | 0.798 ± 0.41 | 0.546 ± 0.21 |
| **AU** | 0.499 ± 0.11 | 0.414 ± 0.15 | 0.526 ± 0.18 | 0.414 ± 0.14 | **0.363** ± 0.10 |
| **BA** | 0.714 ± 0.07 | **0.553** ± 0.07 | 0.833 ± 0.11 | 0.637 ± 0.09 | 0.576 ± 0.05 |
| **BU** | **0.682** ± 0.20 | 0.951 ± 0.10 | 0.878 ± 0.21 | 0.862 ± 0.27 | 1.026 ± 0.29 |
| **CL** | 0.622 ± 0.54 | 0.942 ± 0.38 | **0.399** ± 0.16 | 0.801 ± 0.50 | 0.435 ± 0.16 |
| **CO** | **1.399** ± 0.51 | 1.856 ± 1.13 | 3.698 ± 3.55 | 3.733 ± 1.89 | 2.377 ± 1.61 |
| **CP** | 0.719 ± 0.34 | 0.922 ± 0.47 | 0.832 ± 0.34 | 0.747 ± 0.34 | **0.608** ± 0.23 |
| **EL** | **1.019** ± 0.03 | 1.097 ± 0.09 | 1.537 ± 0.31 | 1.073 ± 0.13 | 1.191 ± 0.14 |
| **FA** | 0.739 ± 0.11 | 0.849 ± 0.10 | 0.948 ± 0.17 | **0.731** ± 0.32 | 0.735 ± 0.09 |
| **FI** | 0.631 ± 0.30 | 0.675 ± 0.43 | 0.543 ± 0.38 | 0.537 ± 0.46 | **0.401** ± 0.50 |
| **FL** | 1.429 ± 0.20 | 1.851 ± 0.46 | 1.751 ± 0.40 | 1.557 ± 0.30 | **1.421** ± 0.13 |
| **FR** | 1.034 ± 0.07 | 1.711 ± 1.42 | 1.557 ± 0.32 | **1.012** ± 0.05 | 1.347 ± 0.37 |
| **GS** | 0.572 ± 0.12 | 0.743 ± 0.07 | **0.497** ± 0.17 | 0.595 ± 0.13 | 0.536 ± 0.11 |
| **HO** | **0.725** ± 0.19 | 0.910 ± 0.14 | 1.040 ± 0.25 | 0.836 ± 0.13 | 0.974 ± 0.17 |
| **HU** | 0.729 ± 0.10 | 0.761 ± 0.19 | 0.748 ± 0.22 | 0.649 ± 0.20 | **0.590** ± 0.19 |
| **NO** | 1.006 ± 0.08 | 1.229 ± 0.22 | 1.363 ± 0.28 | **0.989** ± 0.02 | 1.222 ± 0.18 |
| **NR** | **0.951** ± 0.14 | 1.072 ± 0.16 | 1.272 ± 0.24 | 0.972 ± 0.18 | * |
| **PL** | 0.515 ± 0.02 | 0.733 ± 0.05 | 0.686 ± 0.04 | 0.679 ± 0.02 | **0.420** ± 0.03 |
| **PO** | **0.767** ± 0.26 | 0.976 ± 0.76 | 1.189 ± 0.83 | 1.026 ± 0.80 | 0.792 ± 0.42 |
| **RE** | **0.995** ± 0.03 | 1.059 ± 0.02 | 1.364 ± 0.14 | 1.048 ± 0.05 | 1.229 ± 0.10 |
| **SC** | **0.281** ± 0.14 | 0.449 ± 0.13 | 0.500 ± 0.18 | 0.303 ± 0.08 | 0.370 ± 0.10 |
| **SE** | 0.879 ± 0.08 | 0.921 ± 0.32 | 0.849 ± 0.46 | 0.746 ± 0.21 | **0.495** ± 0.15 |
| **ST** | 1.228 ± 0.93 | 0.744 ± 0.54 | 0.904 ± 0.53 | 0.930 ± 0.71 | **0.707** ± 0.49 |
| **TE** | **1.408** ± 0.86 | 4.398 ± 7.76 | 3.645 ± 6.07 | 16.50 ± 30.3 | 2.512 ± 2.99 |
| **UN** | **0.388** ± 0.04 | 0.558 ± 0.03 | 0.620 ± 0.05 | 0.497 ± 0.04 | 0.844 ± 0.15 |
| **VL** | **0.947** ± 0.07 | 1.056 ± 0.12 | 1.410 ± 0.32 | 1.090 ± 0.29 | * |
| **Mean** | **0.818** | 1.071 | 1.157 | 1.500 | 0.896 |

**Table6**. Relative errors of algorithms ± standard deviation of 10 folds, where 20% target noise are added. Best REs are shown in bold font.

| Dataset | RSBF | KNN | RULE | MARS | DART |
|---|---|---|---|---|---|
| **AB** | **0.726** ± 0.10 | 7.592 ± 2.46 | 9.301 ± 2.27 | 7.602 ± 2.57 | 6.603 ± 1.83 |
| **AI** | 0.906 ± 0.40 | 0.807 ± 0.28 | 1.122 ± 0.46 | 0.856 ± 0.20 | **0.785** ± 0.23 |
| **AU** | **0.398** ± 0.09 | 1.832 ± 0.66 | 2.531 ± 1.32 | 2.107 ± 0.48 | 1.981 ± 0.51 |
| **BA** | 0.675 ± 0.03 | **0.457** ± 0.05 | 0.712 ± 0.10 | 0.537 ± 0.08 | 0.556 ± 0.11 |
| **BU** | **0.935** ± 0.47 | 12.66 ± 4.88 | 12.92 ± 9.44 | 13.30 ± 5.14 | 10.67 ± 5.39 |
| **CL** | **0.834** ± 1.38 | 8.283 ± 2.38 | 11.24 ± 2.51 | 9.393 ± 2.69 | 6.127 ± 3.13 |
| **CO** | 1.702 ± 0.48 | **1.676** ± 0.53 | 3.102 ± 2.05 | 5.874 ± 7.16 | 2.040 ± 1.06 |
| **CP** | 0.720 ± 0.30 | 0.930 ± 0.48 | 0.782 ± 0.33 | 0.745 ± 0.44 | **0.636** ± 0.33 |
| **EL** | **0.995** ± 0.05 | 1.465 ± 0.30 | 1.899 ± 0.64 | 1.148 ± 0.17 | 1.431 ± 0.34 |
| **FA** | **0.170** ± 0.10 | 2.525 ± 0.56 | 3.208 ± 1.48 | 2.447 ± 0.61 | 2.058 ± 0.54 |
| **FI** | 0.653 ± 0.33 | 0.710 ± 0.46 | 0.528 ± 0.30 | 0.501 ± 0.27 | **0.387** ± 0.12 |
| **FL** | **2.366** ± 0.65 | 73.89 ± 31.4 | 77.21 ± 26.6 | 70.90 ± 23.5 | 71.40 ± 30.3 |
| **FR** | **1.036** ± 0.10 | 2.394 ± 1.40 | 3.247 ± 3.23 | 1.710 ± 0.50 | 2.089 ± 1.71 |
| **GS** | **0.430** ± 0.15 | 2.166 ± 0.40 | 2.384 ± 0.67 | 2.164 ± 0.41 | 2.276 ± 0.82 |
| **HO** | **0.754** ± 0.17 | 7.853 ± 1.88 | 11.53 ± 9.88 | 10.29 ± 7.82 | 6.115 ± 2.34 |
| **HU** | **0.575** ± 0.19 | 2.801 ± 1.53 | 3.635 ± 2.45 | 2.893 ± 1.56 | 2.611 ± 1.64 |
| **NO** | **0.909** ± 0.16 | 1.403 ± 0.60 | 2.220 ± 0.63 | 1.037 ± 0.16 | 1.196 ± 0.35 |
| **NR** | **0.947** ± 0.25 | 38.84 ± 10.9 | 42.32 ± 12.0 | 37.66 ± 9.65 | 31.54 ± 11.0 |
| **PL** | **0.411** ± 0.03 | 5.492 ± 0.52 | 5.777 ± 0.68 | 4.921 ± 0.21 | 5.107 ± 0.30 |
| **PO** | **0.692** ± 0.45 | 9.429 ± 15.3 | 9.456 ± 16.3 | 4.213 ± 2.05 | 6.038 ± 9.83 |
| **RE** | **0.958** ± 0.05 | 6.597 ± 1.62 | 10.33 ± 4.09 | 6.759 ± 1.07 | 7.108 ± 1.62 |
| **SC** | **0.533** ± 0.27 | 0.583 ± 0.26 | 0.968 ± 0.84 | 0.700 ± 0.37 | 0.627 ± 0.30 |
| **SE** | **0.697** ± 0.23 | 21.29 ± 6.10 | 27.77 ± 13.5 | 22.01 ± 8.98 | 21.72 ± 9.32 |
| **ST** | **0.646** ± 0.53 | 1.921 ± 1.27 | 3.887 ± 3.03 | 1.966 ± 1.43 | 1.871 ± 1.26 |
| **TE** | **1.747** ± 1.35 | 2.087 ± 2.76 | 4.569 ± 6.64 | 7.267 ± 10.7 | 2.671 ± 2.20 |
| **UN** | 0.634 ± 0.06 | 0.636 ± 0.04 | 0.865 ± 0.06 | **0.541** ± 0.05 | 0.764 ± 0.09 |
| **VL** | **0.976** ± 0.14 | 1.030 ± 0.23 | 1.513 ± 0.38 | 0.977 ± 0.15 | 1.518 ± 0.53 |
| **Mean** | **0.853** | 8.050 | 9.446 | 8.167 | 7.331 |

**Table7**. Relative errors of algorithms ± standard deviation of 10 folds, where 30 irrelevant features are added. Best REs are shown in bold font. (* means result isn't available due to singular variance/covariance matrix)

| Dataset | RSBF | KNN | RULE | MARS | DART |
|---|---|---|---|---|---|
| **AB** | **0.677** ± 0.06 | 0.873 ± 0.06 | 0.934 ± 0.17 | 0.682 ± 0.17 | * |
| **AI** | 0.794 ± 0.28 | 1.514 ± 0.78 | 0.723 ± 0.37 | 0.682 ± 0.29 | **0.657** ± 0.30 |
| **AU** | 0.429 ± 0.08 | 0.538 ± 0.13 | 0.491 ± 0.19 | **0.368** ± 0.12 | 0.511 ± 0.19 |
| **BA** | 0.603 ± 0.03 | 0.568 ± 0.07 | 0.574 ± 0.09 | **0.536** ± 0.05 | 0.628 ± 0.07 |
| **BU** | 1.325 ± 0.68 | 0.968 ± 0.07 | 1.073 ± 0.57 | **0.877** ± 0.50 | 0.969 ± 0.35 |
| **CL** | 1.111 ± 2.00 | 1.162 ± 0.60 | **0.284** ± 0.13 | 2.195 ± 5.52 | 0.306 ± 0.08 |
| **CO** | 2.119 ± 1.02 | 2.854 ± 1.12 | 1.794 ± 0.67 | 4.126 ± 2.69 | **1.662** ± 0.74 |
| **CP** | 0.676 ± 0.39 | 1.107 ± 0.20 | 0.753 ± 0.37 | **0.613** ± 0.26 | 0.668 ± 0.29 |
| **EL** | **1.010** ± 0.12 | 1.037 ± 0.06 | 1.367 ± 0.35 | 1.134 ± 0.16 | 1.236 ± 0.22 |
| **FA** | **0.204** ± 0.13 | 1.026 ± 0.10 | 1.039 ± 0.30 | 0.249 ± 0.16 | 0.877 ± 0.06 |
| **FI** | 0.694 ± 0.36 | 0.917 ± 0.24 | 0.456 ± 0.42 | **0.247** ± 0.23 | 0.420 ± 0.37 |
| **FL** | **1.429** ± 0.21 | 1.454 ± 0.27 | 1.765 ± 0.37 | 1.629 ± 0.31 | 1.490 ± 0.26 |
| **FR** | 1.096 ± 0.18 | **1.063** ± 0.14 | 1.513 ± 0.42 | 1.777 ± 0.76 | 1.430 ± 0.35 |
| **GS** | 0.461 ± 0.17 | 0.802 ± 0.04 | **0.268** ± 0.09 | 0.404 ± 0.09 | 0.573 ± 0.23 |
| **HO** | **0.800** ± 0.19 | 0.932 ± 0.13 | 1.049 ± 0.22 | 0.847 ± 0.21 | 1.165 ± 0.26 |
| **HU** | 0.601 ± 0.12 | 0.920 ± 0.26 | 0.701 ± 0.19 | **0.521** ± 0.17 | 0.653 ± 0.23 |
| **NO** | **1.070** ± 0.14 | 1.079 ± 0.06 | 1.484 ± 0.41 | 1.370 ± 0.34 | 1.156 ± 0.15 |
| **NR** | 0.938 ± 0.14 | 1.076 ± 0.17 | 1.284 ± 0.26 | **0.916** ± 0.16 | * |
| **PL** | 0.450 ± 0.02 | 0.961 ± 0.02 | 0.575 ± 0.05 | **0.407** ± 0.03 | 0.734 ± 0.06 |
| **PO** | **0.838** ± 0.43 | 0.855 ± 0.24 | 0.934 ± 0.39 | 1.005 ± 0.65 | 1.013 ± 0.51 |
| **RE** | **1.014** ± 0.02 | 1.045 ± 0.04 | 1.380 ± 0.12 | 1.042 ± 0.06 | 1.311 ± 0.10 |
| **SC** | 0.672 ± 0.63 | 0.582 ± 0.16 | 0.386 ± 0.18 | **0.305** ± 0.26 | 0.391 ± 0.14 |
| **SE** | 1.036 ± 0.05 | 0.835 ± 0.15 | **0.471** ± 0.24 | 0.798 ± 0.28 | 0.641 ± 0.14 |
| **ST** | 1.104 ± 1.21 | 1.188 ± 0.79 | 0.914 ± 1.28 | 0.817 ± 0.58 | **0.756** ± 0.60 |
| **TE** | **2.222** ± 1.59 | 3.241 ± 3.60 | 5.572 ± 9.31 | 5.614 ± 11.3 | 2.709 ± 2.94 |
| **UN** | **0.385** ± 0.03 | 0.757 ± 0.03 | 0.557 ± 0.06 | 0.394 ± 0.04 | 0.906 ± 0.15 |
| **VL** | **0.930** ± 0.08 | 1.050 ± 0.15 | 1.454 ± 0.25 | 1.257 ± 0.45 | 1.307 ± 0.16 |
| **Mean** | **0.914** | 1.126 | 1.104 | 1.141 | 0.967 |

## References


[1] **Breiman, L, Friedman, J H, Olshen, R A and Stone, C J** '*Classification and Regression Trees*' Wadsworth, Belmont, California (1984)

[2] **Friedman, J H** 'Local Learning Based on Recursive Covering' Department of Statistics, Stanford University

[3] **Weiss, S and Indurkhya, N** ' Rule-based Machine Learning Methods for Functional Prediction' *Journal of Artificial Intelligence Research* Vol 3 (1995) pp 383-403

[4] **Aha, D, Kibler, D and Albert, M** 'Instance-based Learning Algorithms' *Machine Learning* Vol 6 (1991) pp 37 – 66

[5] **Quinlan, J R** 'Learning with Continuous Classes' *Proceedings AI'92* Adams and Sterling (Eds) Singapore (1992) pp 343-348

[6] **Bratko, I and Karalic A** 'First Order Regression' *Machine Learning* Vol 26 (1997) pp 147-176

[7] **Karalic, A** 'Employing Linear Regression in Regression Tree Leaves' *Proceedings of ECAI'92* Vienna, Austria, Bernd Newmann (Ed.) (1992) pp 440-441

[8] **Friedman, J H** 'Multivariate Adaptive Regression Splines' *The Annals of Statistics* Vol 19 No 1 (1991) pp 1-141

[9] **Breiman, L** 'Stacked Regressions' *Machine Learning* Vol 24 (1996) pp 49-64

[10] **Kibler, D, Aha D W and Albert, M K** 'Instance-based Prediction of Real-valued Attributes' *Comput. Intell.* Vol 5 (1989) pp 51-57

[11] **Weiss, S and Indurkhya, N** 'Optimized Rule Induction' *IEEE Expert* Vol 8 No 6 (1993) pp 61-69

[12] **Friedman, J H**, Local Learning Based on Recursive Covering, 1996.

[13] **Graybill, F, Iyer, H and Burdick, R** 'Applied Statistics' Upper Saddle River, NJ (1998)