Revista Brasileira de Computação Aplicada, July, 2020

DOI: 10.5335/rbca.v12i2.10565
Vol. 12, Nº 2, pp. 134–143
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ORIGINAL PAPER

Missing data analysis using machine learning methods to predict the performance of technical students

Gilberto de Melo Júnior [1], Symone G. Soares Alcalá [2], Geovanne Pereira Furriel [1], Sílvio L. Vieira [1]

[1]Electrical and Computer Engineering, Federal University of Goiás, [2]Faculty of Science and Technology, Federal University of Goiás

*gilberto.melo@outlook.com

Received: 2020-01-31. **Revised:** 2020-07-06. **Accepted:** 2020-07-15.

Abstract

Machine learning (ML) has become an emerging technology able to solve problems in many areas, including education, medicine, robotic and aerospace. ML is a specific field of artificial intelligence which designs computational models able to learn from data. However, to develop a ML model, it is necessary to ensure data quality, since real-world data is incomplete, noisy and inconsistent. This paper evaluates state-of-the-art missing data treatment methods using ML algorithms to classify the performance of technical high school students at the Federal Institute of Goiás in Brazil. The aim is to provide an efficient computational tool to aid educational performance that allows the educators to verify the student's tendency to fail. The results indicate that ignoring and discarding method outperforms other missing data treatment methods. Moreover, the tests reveal that Sequential Minimal Optimization, Neural Networks and Bagging outperform the other ML algorithms, such as Naive Bayes and Decision tree, in terms of classification accuracy.

Keywords: Missing Data Treatment Methods; Machine Learning; Evaluation of algorithms.

Resumo

O aprendizado de máquina (ML) tornou-se uma tecnologia emergente capaz de resolver problemas em muitas áreas, incluindo educação, medicina, robótica e aeroespacial. O ML é um campo específico de inteligência artificial que projeta modelos computacionais capazes de aprender com os dados. No entanto, para desenvolver um modelo de ML, é necessário garantir a qualidade dos dados, pois os dados do mundo real são incompletos, ruídosos e inconsistentes. Este artigo avalia métodos avançados de tratamento de dados ausentes usando algoritmos ML para classificar o desempenho de estudantes do ensino médio do Instituto Federal de Goiânia como no Brasil. O objetivo é fornecer uma ferramenta computacional eficiente para auxiliar o desempenho educacional que permite aos educadores verificar a tendência do aluno a reprovar. Os resultados indicam que o método de ignorar e descartar supera outros métodos de tratamento de dados ausentes. Além disso, os testes revelam que a Otimização Mínima Sequencial, Redes Neurais e Bagging superam os outros algoritmos de ML, como Naive Bayes e Árvore de Decisão, em termos de precisão de classificação.

Palavras-Chave: Métodos de tratamento de dados ausentes; Aprendizado de Máquina; Avaliação de algoritmos.

1 Introduction

Machine Learning (ML) is concerned with the question of how to build computer programs that learn and improve automatically through experience (Jordan and Mitchell, 2015). Nowadays, ML has become ubiquitous and indispensable for solving complex problems in most science areas (Obermeyer and

Emanuel, 2016, de Miranda et al., 2016). For example, Jean et al. (2016) combined satellite imagery and ML algorithms to predict poverty in Africa. Reviews of ML applications to analyze genome sequencing data and to support diagnosis of diseases are conducted in references Libbrecht and Noble (2015), Tagaris et al. (2018), respectively. Ahmad et al. (2018) developed a tutorial covers the definitions, nuances, challenges, and requirements for the design of interpretative and explainable machine learning models and systems in healthcare.

ML has also been used as a tool for decision making, prediction and optimization in the area of education.ML algorithms are proposed to predict the educational performance using the database of an education institution by de Melo et al. (2017). The proposed algorithms allow the education professional, even in the first months of the school year, to verify the student's tendency to fail.

Similar jobs applied in education using Machine Learning can be seen at: Minaei-Bidgoli et al. (2003) present an approach to classifying students in order to predict their final grade based on features extracted from logged data in an education Web-based system; Kolo et al. (2015) use of a decision tree approach for predicting student' academic performance; Yukselturk et al. (2014) use 4 ML algorithms to classify students who dropped out of school; Ayinde et al. (2013) find out interesting patterns in the educational data that could contribute to predicting student performance; Kumar et al. (2011) use ML algorithms to predict the performance of students in their final exam.

The ML algorithm development involves some difficulties. For example, the performance of the ML algorithms depends on the data quality employed during the algorithm development. Additionally, in real data sets, noisy, missing and unreliable samples are common. For this reason, pre-processing techniques, such as, outlier removal, missing data treatment and others, are necessary to handle these problems. Usually, missing values occur in data being forgotten or lost; certain values are not applicable for a given variable; or, the designer of the data does not care about the values (Soares, 2015). Missing data values occur in several applications, so that several missing data treatment techniques have been proposed in literature (Zhu et al., 2018). The most common technique is the *ignoring and discarding approach*, which discards all samples with missing values. However, this technique is not viable when the data set is small, therefore other techniques (such as, mean or median substitution, and linear interpolation) are preferred.

This paper evaluates a number of missing data treatment techniques using state-of-the-art ML algorithms in order to predict students' performance in technical high school education at the Federal Institute of Goiás in Brazil. The main objective is to provide an efficient and valuable computational tool to aid educational performance that allows educators to verify the student's tendency to fail. Since the available data set to predict the students' performance contains missing data values, this paper investigates and evaluates missing data treatment techniques to design ML algorithms. The experimental results reveal that, for this case study, the ignoring and discarding approach outperforms other missing data treatment techniques when applied in most ML algorithms.

The main contributions of this paper are: (*i*) proposal and evaluation of state-of-the-art missing data treatment methods using a case study to predict student performance; and (*ii*) evaluation of state-of-the-art ML algorithms (including an ensemble learning algorithm) using state-of-the-art missing data treatment methods and this real case study.

The paper is organized as follows: Section 2 and Section 3 present background on the state-of-the-art missing data treatment methods and ML algorithms, respectively; Section 4 presents methods and materials applied in this paper; Section 5 presents and discusses the experimental results using the missing data treatment methods and ML algorithms to predict student performance; finally, Section 6 presents concluding remarks.

2 Missing Data Treatment Methods

This section presents popular methods for missing data treatment in the ML scope, since missing values can affect the accuracy of ML algorithms.

As described previously, in literature, several missing data treatment methods have been proposed (Zahin et al., 2018). The most popular missing data treatment methods include discarding the samples with missing values (know as *ignoring and discarding*, and *listwise deletion*), and *imputation approaches*. The first approach reduces the data set size by eliminating all the samples with missing values; on the other hand, *imputation approaches* aim to keep the data set size by replacing missing values in a data set by some plausible values. According to Gao et al. (2018), *imputation approaches* outperform *ignoring and discarding approaches*, as they produce complete data sets and make use of the samples that deletion techniques would remove.

Taking this into account, this work evaluates *ignoring and discarding approach*, and six *imputation approaches* to deal with missing values:

- **Ignoring and discarding.** This approach removes all samples that present missing values. The main advantage is convenience; however, it reduces the number of samples.
- **Mean imputation.** The missing value for a given attribute in a sample is replaced by the mean value of all the sample values for that attribute. If the replaced value is not conditioned on the values of other attributes in the record, this approach is called imputing unconditional mean. Despite this approach is simple to be implemented, its disadvantage is that the variance of the replaced attribute and its co-variance with other attributes are systematically underestimated (Lakshminarayan et al., 1999).
- **Median imputation.** In this technique, each missing value in each attribute is replaced by the median

value of all non-missing values of that attribute. It should be employed when the distribution of the underlying attribute is not symmetric. As the mean imputation approach, this technique is simple to implement.

- **Last Observation Carried Forward (LOCF).** In this approach, a missing value in an attribute is replaced by the last measured value before the missing one. This approach is easy to understand and implement; but it assumes that the value of the attribute remains unchanged (Kang, 2013).
- **Linear interpolation.** The missing value is computed by the linear interpolation of the known values of which the missing value is located. The usual motivation for linear interpolation is simplicity, and linear functions are the easiest to determine (Pownuk and Kreinovich, 2017).
- **Spline interpolation.** The missing value is replaced by piece-wise cubic spline interpolation of the non-missing values of that attribute. A spline function consists of polynomial pieces on sub-intervals joined together with certain continuity conditions (De Boor et al., 1978).
- **Piecewise Cubic Hermite Interpolating Polynomial (PCHIP).** This method replaces the missing value by the shape-preserving piece-wise cubic spline interpolation non-missing values of that attribute.

3 Machine Learning Algorithms

ML approaches are computer programs used to solve problems using data or past experience (Rudolph and Martinez, 2015). They are employed in a wide range of applications, including forecasting problems (Zhang, Teng and Chen, 2018) and image classification problems (Yuan et al., 2019). This work compares six state-of-the-art ML algorithms able to automatically classify the performance of students in a technical high school education. To do so, five single learning algorithms (i.e. Naive Bayes, Sequential Minimal Optimization, Decision Tree, Decision Rule and Neural Network) and one ensemble learning algorithm (i.e. Bagging) are considered. The next subsections detail each ML algorithm.

3.1 Naive Bayes

Naive Bayes is a Bayesian probabilistic algorithm based on the Bayes's Theorem. It is simple ML algorithm, with clear semantics, to represent, use, and learn probabilistic knowledge (Witten et al., 2016). The term "naive" comes from the hypothesis that the attribute values of a sample are independent of its class.

To design a Naive Bayes model, consider a data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with N samples of attributes $\mathbf{x}_i$ and labels y_i associated to a supervised classification task, where $\mathbf{x}_i = \{x_{i,1}, \dots, x_{i,m}\}$; m is the number of attributes; $x_{i,j}$ is the j-th attribute value of $\mathbf{x}_i$; $y_i \in C = \{c_1, \dots, c_K\}$ is a K-class (with $K = 2$ in this study). Moreover, consider that $P(x_{i,j}|c_k)$ denotes the conditional probability distribution of attribute $x_{i,j}$ belonging to class $c_k \in C$ (Faceli et al., 2011), and $P(c_k)$ is the prior probability of class c_k in the data set D. Then, for a given test instance $\mathbf{x}_t$, its output value (estimated class) by the Naive Bayes model can be mathematically obtained in Eq. (1) (Wu et al., 2015):

$$c(\mathbf{x}_t) = \underset{c_k \in C}{\operatorname{argmax}} P(c_k) \prod_{j=1}^{m} P(x_{t,j}|c_k) \tag{1}$$

The Naive Bayes learning algorithm involves a learning step procedure in which the values of $P(c_k)$ and $P(x_{i,j}|c_k)$ are calculated. According to Shanahan (Shanahan, 2012), one difference between the Naive Bayes algorithm and other ML algorithms is that there is no explicit search through the space of possible models; instead, the model is obtained without searching by calculating the frequency of various data combinations within the training samples.

3.2 Sequential Minimal Optimization

Support Vector Machines (SVMs) are algorithms based on the statistical learning theory. To train a SVM model, it is required the solution of a large Quadratic Programming (QP) involving an optimization problem. According to Platt (1999), the Sequential Minimal Optimization (SMO) algorithm breaks the QP problem into a series of small possible problems to ensure convergence. Thus, they are solved analytically, avoiding the use of the time consuming numerical optimization. SMO is an efficient learning algorithm to handle with large training data sets, because the amount of memory required for the SMO is linear to the size of the training set. The SMO algorithm is detailed in paper (Zhang, Wang, Lu, Wang and Ma, 2018).

3.3 J48 - Decision Tree

J48 is a decision tree learning model based on the C4.5 algorithm, which builds a decision tree using a divide and conquer strategy (Ruggieri, 2002). The goal of the J48 learning algorithm is to create a binary tree that includes: a root node, which consists of all input data; internal nodes, which are associated with a decision function; and leaf nodes, which show the output of a given input. The J48 model outperforms other decision tree models in terms of classification accuracy (Pham et al., 2017).

Moreover, the J48 algorithm has other attracting features. For example, it is available as an open source in the WEKA project, is easy to understand, makes use of categorical and continuous values, handles missing values, and provides a tree pruning process (Aljawarneh et al., 2017).

In the J48 algorithm, a model is built in two main stages as follows (Bharti et al., 2010) and (Tien Bui et al., 2014): in the first step, a classification tree is designed; and in the second step, the classification tree is pruned. Specifically, in the first step, the input data with the highest gain rate is determined. This is done in the root node of the tree classification; and then, a division process is implemented, using the training data set, to create sub-nodes based on the values in the node root. In the second step, the gain rate value is generated individually for all sub-nodes; and then, the classification variables (slip or non-landslide) are determined based on each gain rate value of each sub-node.

3.4 OneR - Decision Rules

OneR is a simple classification algorithm, which presents high degree of precision. It generates a rule for each predictor in the data, and then selects the rule with the lowest total error, being called a "single rule". To create a rule for a predictor, it constructs a frequency table for each predictor and the target. OneR produces rules that are only slightly less accurate than the last generation sorting algorithms, while it generates rules that are simple for humans to interpret (Witten et al., 2016). Therefore, the main features of the OneR algorithm include: simplicity, high degree of accuracy and easy interpretation of rules.

The main steps of the OneR algorithm are (Nasa and Suman, 2012): (1) for each attribute j and for each value v of that attribute, create a rule; (2) calculate how often each class appears; (3) find the most frequent class $c^{'}$; (4) make a "single rule"; (5) calculate the error rate of this rule; and (6) select the attribute whose rules produce the lowest error rate.

3.5 Neural Networks

Neural Networks (NNs) are ML algorithms inspired by the biological neurons. The NN model has processing elements (neurons), connections between them (weights) and a learning/training algorithm. The main features of the NN model are generalization, robustness, massive parallelism, learning and adaptation (Kasabov, 1996). There are many NN architectures, but Multilayer Perceptrons (MLPs) are the most popular and efficient NN architecture (Soares, 2015). It contains one input layer, one or multiple hidden layers and one output layer. Fig. 1 shows a generic MLP architecture, where the input layer has x neurons, the hidden layer has h neurons, and the output layer has y neurons.

In literature, many learning NN algorithms can be found to obtain the NN parameters (weights and biases), but the most popular is the back-propagation algorithm. It employs iteratively a gradient descent method to select the NN parameters; and its main advantages include reverse propagation capability, good performance for problems in which no relationship is found between output and inputs, flexibility, and great learning ability (Saduf and Wani, 2013).

3.6 Bagging Ensemble

According to Soares (2015), ensemble learning models are sets of learning algorithms that combine in some way their decisions, or their learning algorithms, or different data to obtain accurate predictions. This is because, in most case, an ensemble learning model is more accurate than any single model used separately. The effectiveness of ensemble learning models has been proved in different applications (Tamvakis et al., 2018).

The most popular ensemble learning model is Bagging (Breiman, 1996). It promotes diversity between the individual models by creating a different training data set for each model using bootstrap (Dinakaran and Thangaiah, 2017). Bootstrap is a resampling approach which can produce a new training data set by randomly drawing with replacement from the original training data set. Statistically, each new training data set contains on average 63.2% of samples from the original training set. In the Bagging algorithm, after training each model with a different training data set, the aggregation step combines all the models' outputs/classifications using a simple voting method (i.e. the models have the same contribution on the final classification) (Soares et al., 2012).

4 Problem description: Student performance prediction

This paper aims to develop a predictive tool, using ML algorithms, able to predict whether a student will be "approved" or "disapproved" in the initial bimesters of each technical high school year based on historical data. To do so, it will be performed a comparison between state-of-the-art ML algorithms and missing data strategies to create a powerful predictive tool to estimate student performance. This tool will help education professionals in actions to improve

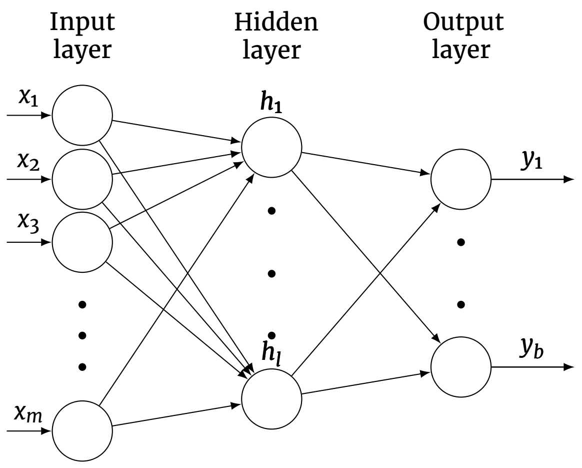

Figure 1: A multilayer perceptron neural network architecture.

Brazilian education and the student performance. For example, Stimpson and Cummings (2014) proposed ML algorithms to help education professionals. Their results revealed that early information for education professionals provides the development of targeted intervention methods, as it allows accurate estimations to be made earlier in the course.

In this work, data were collected from a technical high school of a campus of the Federal Institute of Education, Science and Technology of Goiás, in Brazil, where each technical course is divided in four years and each year is divided in four *bimesters* (a bimester corresponds to two months). In each bimester, assessments and attendance checks are performed for each student. A student is approved whether: the average of the bimesters' grades is equal or greater than 6.0; and the attendance is equal or greater than 75% in the classes. Data were obtained from students of three technical high school courses (Electronics, Electrotechnology and Informatics) between the years 2008 and 2013.

The original data set contains 15788 samples (where each sample is associated to a student), and it was divided in three data sets:

- *data set 1*: grade and attendance of the first bimester;
- *data set 2*: grade and attendance of the first and second bimesters;
- *data set 3*: grade and attendance of the first, second and third bimesters;

where data sets 1, 2 and 3 have 7, 9 and 11 attributes, respectively; and each sample contains the label **Approved** or **Disapproved**. Therefore, it is binary classification task (two classes problem). Table 1 illustrates the attributes for all the data sets. It should be observed that, *data set 1* uses data (course's ID; and student's data, grade and number of missed classes) from the first bimester to predict an approval or a disapproval at a scholar year; while *data set 2* employs data from the first and second bimesters to predict an approval or a disapproval at a scholar year; and *data set 3* applies data from the first, second and third bimester to estimate an approval or a disapproval at a scholar year. Additionally, it should be pointed that, data from the fourth bimester is not used, because, in this period, the final student performance (approved or disapproved) is obtained. Further details of each attribute can be found in paper (de Melo et al., 2017).

The data sets present a large number of missing values. For example, *data set 1*, *data set 2* and *data set 3* have 14.54%, 27.28% and 31.84% presented missing data in the attributes, respectively. To deal with these missing values, seven missing data treatment methods are proposed in this paper (as described in Section 3). For each data set (*data set 1*, *data set 2* and *data set 3*), seven missing data treatment methods were applied to build seven different pre-processed data sets (cases); and then, each pre-processed data set is employed to design ML algorithms. Table 2 shows this procedure and the identification of each pre-processed data set. Therefore, the number of pre-processed data sets is 21.

To evaluate each ML learning algorithm (described in Section 4) trained with a pre-processed data set, *cross-validation* method is applied. In cross-validation, the data set is randomly divided into k folds of equal size. At each cross-validation iteration, one fold is used for testing (to obtain the classification accuracy) and the other $k - 1$ folds are used for training the ML learning algorithm. The test values (classification accuracy values) are calculated and averaged over all the k folds; and then, this average value corresponds to the final ML learning algorithm's accuracy (Bouckaert et al., 2008).

In this work, the number of folds is set to 10. Moreover, in order to get statistically meaningful results, each ML learning algorithm is evaluated in 20 runs. That is, for each ML learning algorithm, 10-fold cross-validation is applied in 20 runs. Therefore, in the experimental results below, the classification accuracy corresponds to the average of the 10-fold cross-validation values in 20 runs. The algorithm with the best performance (i.e. the highest percentage of correctly predicted samples) will be used in the ensemble learning model (Bagging).

The ML algorithms were implemented using Weka (Waikato Environment for Knowledge Learning). It is a software developed by students, from the University of Waikato in New Zealand, with an initial purpose of identifying information coming from raw data in agricultural applications (Vaithiyanathan et al., 2013).

The Weka software contains many tools for data pre-processing, classification, clustering, association, regression and feature selection. This paper uses six classification algorithms (Naive Bayes, SMO, J48, OneR, NN and Bagging (MLP mode)) from the Weka software,

Table 1: Description of the attributes in all the data sets.

Attributes	Data set 1	Data set 2	Data set 3
Course's ID	x	x	x
Student's year of birth	x	x	x
Student's gender	x	x	x
Student's birth location	x	x	x
Student's academic year	x	x	x
Student's grade in the first bimester	x	x	x
Number of missed classes in the first bimester	x	x	x
Student's grade in the second bimester		x	x
Number of missed classes in the second bimester		x	x
Student's grade in the third bimester			x
Number of missed classes in the third bimester			x

Table 2: Description of the pre-processed data sets using the missing data treatment methods.

Missing data treatment technique	Data set 1	Data set 2	Data set 3
Mean imputation	Case 1	Case 8	Case 15
Median imputation	Case 2	Case 9	Case 16
Last Observation Carried Forward (LOCF)	Case 3	Case 10	Case 17
Linear interpolation	Case 4	Case 11	Case 18
Spline interpolation	Case 5	Case 12	Case 19
Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)	Case 6	Case 13	Case 20
Ignoring and discarding	Case 7	Case 14	Case 21

where the ML learning algorithms' parameters are set to default (in case of NN with six perceptrons).

5 Experimental Results and Discussions

In this section, missing data treatment techniques and state-of-the-art ML algorithms are evaluated and discussed to predict students' approval and disapproval in a technical high school education. The experiments have been done on the Weka software, running on a PC equipped with an Intel(R) Core(TM) i5-7200U 2.5 GHz-2.71 GHz processor of 4 cores and 4 GB of RAM.

Experimental results using all the missing data treatment techniques and the single ML algorithms (Naive Bayes, SMO, J48, OneR and NN) are presented in Table 3. As described previously, for each missing data treatment and for each ML algorithm, the simulation was conducted in 20 runs. The average and standard deviation of the percentage of correct instances (classification accuracy) using the 10-fold cross-validation on 20 runs is reported. The underlined values highlight the ML algorithm with the best performance in a case (pre-processed data set); highlighted in blue values the best missing data treatment technique for a single ML algorithm; and the circle (◦) highlights results with statistically significant improvement, it generated automatically by Weka.

From the results presented in Table 3, some conclusions can be drawn, for example:

- the JR8 algorithm outperforms the other single ML algorithms in most cases from data set 1 (i.e. it achieves best performance in five cases from data set 1);
- the NN algorithm outperforms the other single ML algorithms in all the cases from data set 2 (i.e. it achieves best performance in all the cases from data set 2);
- the SMO algorithm outperforms the other single ML algorithms in most cases from data set 3 (i.e. it achieves best performance in six cases from data set 3);
- the NN algorithm presents the best average of classification accuracy in all the cases;
- the best classification accuracy (i.e., 96.0574 ± 0.5261) was achieved by SMO algorithm in case 21, which uses *ignoring and discarding method* in the data set 3;
- the results from data set 3 are better when compared to the data sets, since it uses more attributes;
- in most results, the best missing data treatment technique for a ML algorithm is the *ignoring and discarding method*;
- in most results, the missing data treatment techniques with the worst performance are *median imputation* and *spline interpolation* methods.

Therefore, according to Table 3, for all cases, the single ML algorithms that present best performance are SMO, J48 and NN. But, it is noted that NN presents best classification accuracy when compared to the other ML algorithms. Thus, a more detailed implementation of NN algorithm is proposed by tuning the number of hidden neurons using case 21 (data set 3 with *ignoring and discarding method*).

Table 4 indicated the NN performance when the number of hidden neurons in the hidden layer varies from 1 to 20, using the pre-processed data from case 21. As it can be seen, the NN accuracy tends to decrease when the number of hidden neurons increases.

Other experiments are performed to analyze an ensemble system (Bagging) performance using NN as ML algorithm. Table 5 presents the Bagging performance when the number of hidden neurons in NN algorithm varies from 1 to 20. As it can be seen, the best accuracy of Bagging with NN is achieved when the number of hidden neurons is 4 (highlighted in blue on the table). In contrast, Fig. 2 compares the performance of Bagging with NN to a single (one) NN, when the number of hidden neurons varies. The results show that, for a ll number of hidden neurons, the Bagging algorithm outperforms a single NN algorithm, in terms of average of the classification accuracy.

Additionally, other tests with case 21 are performed to compare the performance of SMO, NN (with one hidden neuron) and Bagging (with four hidden neuron) using other evaluation metrics, as shown in Table 6. The evaluation metrics are the standard classification accuracy (%), kappa statistic, Mean Absolute Error (MAE), precision, recall, F-Measure, Matthews correlation coefficient (MCC), and Receiving Characteristics of Operation (ROC) area (Faceli et al., 2011). The algorithms present good performance, almost equal from the "ideal algorithm" (where "ideal algorithm" represents the best/ideal performance of a ML algorithm). SMO demonstrates MAE, Recall and F-Measure nearest from the ideal algorithm. Moreover, NN presents the best Precision and ROC-area. Finally, Bagging has classification accuracy, Kappa statistics, F-Measure and MCC almost the ideal algorithm.

Table 3: Accuracy results of the single ML algorithms using different missing data treatment methods.

Case	Naive Bayes	SMO	J48 ○	OneR	NN ○
1	**89.7390 ± 0.5372**	89.5471 ± 0.3247	90.6711 ± 0.5547	89.9781 ± 0.4875	90.3699 ± 0.5865
2	89.2504 ± 0.5555	89.5351 ± 0.3113	90.5460 ± 0.5300	89.8768 ± 0.4744 ○	90.3613 ± 0.5818
3	89.5097 ± 0.5259	89.8160 ± 0.3287	90.5900 ± 0.5095	90.0117 ± 0.4707 ○	90.4548 ± 0.5676
4	89.4660 ± 0.5339	89.8626 ± 0.3433 ○	90.6027 ± 0.5097	89.9699 ± 0.4774 ○	90.5064 ± 0.5600
5	89.4632 ± 0.5465	89.9199 ± 0.3439 ○	90.5336 ± 0.5324	89.9867 ± 0.4829 ○	90.5501 ± 0.5328
6	89.3542 ± 0.5632	89.8622 ± 0.3400 ○	90.5875 ± 0.4914	89.9446 ± 0.4763 ○	90.5054 ± 0.5866
7	89.7258 ± 0.6666	**89.9366 ± 0.3658**	**90.7082 ± 0.5265**	**90.2320 ± 0.4777**	**90.7835 ± 0.5864**
8	90.6099 ± 0.5881	92.0164 ± 0.4302 ○	92.1275 ± 0.5839	91.1819 ± 0.4780 ○	92.3207 ± 0.5405
9	90.1799 ± 0.6171	91.9157 ± 0.4140 ○	91.9955 ± 0.5203	91.1819 ± 0.4780 ○	92.3404 ± 0.5367
10	89.9683 ± 0.6604	91.9306 ± 0.4310 ○	91.8742 ± 0.5455	90.6647 ± 0.5639 ○	92.0595 ± 0.5470
11	90.1973 ± 0.6192	92.1561 ± 0.4273 ○	91.9056 ± 0.5670	90.7468 ± 0.5322 ○	92.2400 ± 0.5010
12	89.7717 ± 0.6444	91.2655 ± 0.4378 ○	91.8913 ± 0.5662	90.7173 ± 0.5388 ○	92.0484 ± 0.5867
13	90.0583 ± 0.6398	92.0927 ± 0.4547 ○	91.8213 ± 0.5437	90.8956 ± 0.5315 ○	92.1656 ± 0.5303
14	**91.8021 ± 0.6907**	**93.6151 ± 0.5467** ○	**93.2759 ± 0.5516**	**92.8778 ± 0.5340**	**93.6142 ± 0.5887**
15	92.4262 ± 0.5542	94.4920 ± 0.4570 ○	94.1572 ± 0.5138	93.2113 ± 0.4453 ○	94.4128 ± 0.5116
16	91.9885 ± 0.5770	94.4543 ± 0.4495 ○	93.9549 ± 0.5045	93.2534 ± 0.4604 ○	94.3656 ± 0.5510
17	92.1624 ± 0.6150	94.4679 ± 0.4849 ○	94.0372 ± 0.4947	93.4498 ± 0.4465 ○	94.3866 ± 0.5025
18	92.3426 ± 0.5911	94.5271 ± 0.4945 ○	94.0065 ± 0.4643	93.4494 ± 0.4514 ○	94.4765 ± 0.5159
19	92.1029 ± 0.6285	94.2903 ± 0.4818 ○	93.9048 ± 0.5309	93.4526 ± 0.4489 ○	94.3742 ± 0.5082
20	92.2704 ± 0.5912	94.5259 ± 0.4908 ○	93.9574 ± 0.4923	93.4453 ± 0.4524 ○	94.4356 ± 0.4847
21	**93.9012 ± 0.6856**	**96.0574 ± 0.5261** ○	**95.3633 ± 0.5598**	**94.6985 ± 0.4955** ○	**95.8702 ± 0.5947**
Mean	90.7757	92.2041	92.3100	91.5822	92.5067

The underlined values highlight the ML algorithm with the best performance in a case; highlighted in blue values the best missing data treatment technique for a single ML algorithm, and the circle (○) highlights results with statistically significant improvement.

6 Conclusions

This work evaluates seven missing data treatment methods and six ML algorithms to estimate students' performance in technical high school education at the Federal Institute of Goiás in Brazil. The aim is to propose an efficient computational tool to aid educational performance that allows the education professional to verify the student's performance tendency in a technical course.

According to the reported results, for this case study,

Table 4: Accuracy of the NN algorithm, when the number of hidden neurons varies, using data from case 21.

Number of hidden neurons in the NN algorithms	Percentage of correct classifications
1	95.9747 ± 0.6196
2	95.8878 ± 0.5698
3	95.8781 ± 0.5535
4	95.9143 ± 0.5932
5	95.8525 ± 0.5938
6	95.8702 ± 0.5947
7	95.8432 ± 0.6103
8	95.8409 ± 0.6268
9	95.7680 ± 0.6133
10	95.7833 ± 0.5781
11	95.7596 ± 0.6160
12	95.7243 ± 0.5870
13	95.7164 ± 0.5937
14	95.6960 ± 0.6103
15	95.6992 ± 0.6035
16	95.6588 ± 0.5779
17	95.7085 ± 0.6343
18	95.6876 ± 0.6177
19	95.6151 ± 0.6570
20	95.6537 ± 0.6124

Table 5: Accuracy of Bagging and a single NN, when the number of hidden neurons varies, using data from the case 21.

Number of hidden neurons	Percentage of correct classifications
1	96.0672 ± 0.5547
2	95.9924 ± 0.5472
3	96.0695 ± 0.5330
4	**96.0714 ± 0.5429**
5	96.0639 ± 0.5478
6	96.0598 ± 0.5496
7	96.0296 ± 0.5564
8	96.0454 ± 0.5533
9	96.0347 ± 0.5614
10	96.0416 ± 0.5606
11	96.0565 ± 0.5638
12	96.0351 ± 0.5806
13	96.0384 ± 0.5628
14	96.0402 ± 0.5823
15	96.0319 ± 0.5759
16	96.0342 ± 0.5859
17	96.0644 ± 0.5798
18	96.0379 ± 0.5538
19	96.0249 ± 0.5619
20	96.0537 ± 0.5292

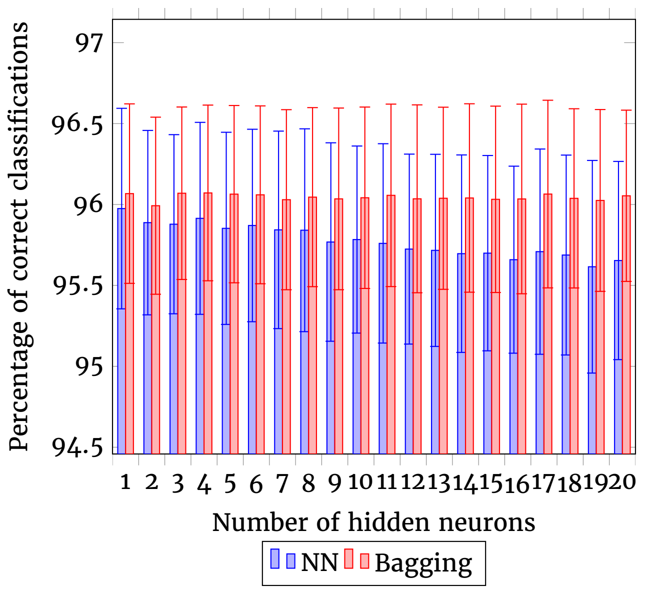

Figure 2: Comparison between a single NN and Bagging with NNs when the number of hidden neurons varies.

Table 6: Evaluation metrics for the best ML algorithms: SMO, NN and Bagging Metrics.

	Ideal algorithm	SMO	NN	Bagging
Cl. accuracy %	100	96.0574	95.9747	96.0714
Kappa Statistic	1	0.7599	0.7684	0.7693
MAE	0	0.0394	0.0718	0.0545
Precision	1	0.9632	0.9688	0.9671
Recall	1	0.9940	0.9869	0.9899
F-Measure	1	0.9783	0.9777	0.9783
MCC	1	0.7721	0.7736	0.7757
ROC-Area	1	0.8342	0.9707	0.9701

the missing data treatment method with the best performance is *ignoring and discarding* method; while *median imputation* and *spline interpolation* methods have the worst performance. Regarding the ML algorithms, three out of the six achieved the best classification accuracy: SMO (96.0574%), NN (95.9747%) and Bagging (96.0714%). Therefore, it can be concluded that the ML algorithms that used the *ignoring and discarding* method can be used in the classification of student's performance.

Seeking to improve the tools used, it is still possible to think of applications of such tools in undergraduate courses. For this purpose, the same analyzes and procedures can be used, which can bring significant improvements to the pedagogical assistance of educational institutions.

Future works will be devoted to analyze other missing data treatment methods and optimize the ML algorithm's parameters. Moreover, future efforts will be done to propose other ML algorithms, such deep learning algorithm, in this case study.

Acknowledgment

The authors would like to thank Coordination for the Improvement of Higher Education Personnel (CAPES) and the National Council for Scientific and Technological Development (CNPq) for financial support research and scholarships.

References

Ahmad, M. A., Teredesai, A. and Eckert, C. (2018). Interpretable machine learning in healthcare, *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 447–447. https://doi.org/10.1109/ICHI.2018.00095.

Aljawarneh, S., Yassein, M. B. and Aljundi, M. (2017). An enhanced j48 classification algorithm for the anomaly intrusion detection systems, *Cluster Computing* **20**. https://doi.org/10.1007/s10586-017-1109-8.

Ayinde, A., Adetunji, A., Bello, M. and Odeniyi, O. (2013). Performance evaluation of naive bayes and decision stump algorithms in mining students' educational data, *International Journal of Computer Science Issues (IJCSI)* **10**(4): 147.

Bharti, K., Jain, S. and Shukla, S. (2010). Fuzzy k-mean clustering via j48 for intrusion detection system, *International Journal of Computer Science and Information Technologies* **1**(4): 315–318.

Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A. and Scuse, D. (2008). Weka manual for version 3-6-0, *University of Waikato, Hamilton, New Zealand* **2**.

Breiman, L. (1996). Bagging predictors, *Machine learning* **24**(2): 123–140. https://doi.org/10.1023/A:1018054314350.

De Boor, C., De Boor, C., Mathématicien, E.-U., De Boor, C. and De Boor, C. (1978). *A practical guide to splines*, Vol. 27, Springer-Verlag New York.

de Melo, G., Oliveira, S. M., Ferreira, C. C., Vasconcelos Filho, E. P., Calixto, W. P. and Furriel, G. P. (2017). Evaluation techniques of machine learning in task of reprovation prediction of technical high school students, *Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2017 CHILEAN Conference on*, IEEE, pp. 1–7. https://doi.org/10.1109/CHILECON.2017.8229739.

de Miranda, A. R., de A.B̃arbosa, T. M. G., Araújo, R. and Alcalá, S. G. S. (2016). An on-line extreme learning machine with adaptive architecture for soft sensor design, *IEEE Sensors Conference*, pp. 1–3. https://doi.org/10.1109/ICSENS.2016.7808721.

Dinakaran, S. and Thangaiah, P. R. J. (2017). Ensemble method of effective adaboost algorithm for decision tree classifiers, *International Journal on Artificial*

Intelligence Tools **26**(03): 1750007. https://doi.org/10.1142/S0218213017500075.

Faceli, K., Lorena, A. K., Gama, J. and Carvalho, A. C. P. L. F. D. (2011). *Artificial Intelligence: Machine Learning Approach*, LTC.

Gao, Y., Merz, C., Lischeid, G. and Schneider, M. (2018). A review on missing hydrological data processing, *Environmental earth sciences* **77**(2): 47. https://doi.org/10.1007/s12665-018-7228-6.

Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B. and Ermon, S. (2016). Combining satellite imagery and machine learning to predict poverty, *Science* **353**(6301): 790–794. https://doi.org/10.1126/science.aaf7894.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects, *Science* **349**(6245): 255–260. https://doi.org/10.1126/science.aaa8415.

Kang, H. (2013). The prevention and handling of the missing data, *Korean journal of anesthesiology* **64**(5): 402–406. https://doi.org/10.4097/kjae.2013.64.5.402.

Kasabov, N. K. (1996). *Foundations of neural networks, fuzzy systems, and knowledge engineering*, Marcel Alencar.

Kolo, K. D., Adepoju, S. A. and Alhassan, J. K. (2015). A decision tree approach for predicting students academic performance, *International Journal of Education and Management Engineering* **5**(5): 12. https://doi.org/10.5815/ijeme.2015.05.02.

Kumar, S. A. et al. (2011). Efficiency of decision trees in predicting student's academic performance. https://doi.org/10.5121/csit.2011.1230.

Lakshminarayan, K., Harp, S. A. and Samad, T. (1999). Imputation of missing data in industrial databases, *Applied intelligence* **11**(3): 259–275. https://doi.org/10.1023/A:1008334909089.

Libbrecht, M. W. and Noble, W. S. (2015). Machine learning applications in genetics and genomics, *Nature Reviews Genetics* **16**(6): 321. https://doi.org/10.1038/nrg3920.

Minaei-Bidgoli, B., Kashy, D. A., Kortemeyer, G. and Punch, W. F. (2003). Predicting student performance: an application of data mining methods with an educational web-based system, *33rd Annual Frontiers in Education, 2003. FIE 2003.*, Vol. 1, pp. T2A–13. https://doi.org/10.1109/FIE.2003.1263284.

Nasa, C. and Suman, S. (2012). Evaluation of different classification techniques for web data, *International journal of computer applications* **52**(9): 34–40. https://doi.org/10.5120/8233-1389.

Obermeyer, Z. and Emanuel, E. J. (2016). Predicting the future—big data, machine learning, and clinical medicine, *The New England journal of medicine* **375**(13): 1216. https://doi.org/10.1056/NEJMp1606181.

Pham, B. T., Bui, D. T. and Prakash, I. (2017). Landslide susceptibility assessment using bagging ensemble based alternating decision trees, logistic regression and j48 decision trees methods: a comparative study, *Geotechnical and Geological Engineering* **35**(6): 2597–2611. https://doi.org/10.1007/s10706-017-0264-2.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization, *Advances in kernel methods*, MIT press, pp. 185–208.

Pownuk, A. and Kreinovich, V. (2017). Why linear interpolation?, *Mathematical structures and modeling* **43**(3). http://digitalcommons.utep.edu/cs_techrep/1098.

Rudolph, G. and Martinez, T. (2015). Finding the real differences between learning algorithms, *International Journal on Artificial Intelligence Tools* **24**(03): 1550001. https://doi.org/10.1142/s0218213015500013.

Ruggieri, S. (2002). Efficient c4. 5 [classification algorithm], *IEEE transactions on knowledge and data engineering* **14**(2): 438–444. https://doi.org/10.1109/69.991727.

Saduf, M. A. W. and Wani, A. (2013). Comparative study of back propagation learning algorithms for neural networks, *International Journal of Advanced Research in Computer Science and Software Engineering* **3**(12). https://doi.org/10.5121/ijcsit.2013.5407.

Shanahan, J. G. (2012). *Soft computing for knowledge discovery: introducing Cartesian granule features*, Vol. 570, Springer Science & Business Media. https://doi.org/10.1007/978-1-4615-4335-0.

Soares, S., Antunes, C. and Araújo, R. (2012). A genetic algorithm for designing neural network ensembles, *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, ACM, pp. 681–688. https://doi.org/10.1145/2330163.2330259.

Soares, S. G. (2015). *Ensemble learning methodologies for soft sensor development in industrial processes*, PhD thesis, University of Coimbra. http://hdl.handle.net/10316/28313.

Stimpson, A. J. and Cummings, M. L. (2014). Assessing intervention timing in computer-based education using machine learning algorithms, *IEEE Access* **2**: 78–87. https://doi.org/10.1109/ACCESS.2014.2303071.

Tagaris, A., Kollias, D., Stafylopatis, A., Tagaris, G. and Kollias, S. (2018). Machine learning for neurodegenerative disorder diagnosis—survey of practices and launch of benchmark dataset, *International Journal on Artificial Intelligence Tools* **27**(03): 1850011. https://doi.org/10.1142/S0218213018500112.

Tamvakis, A., Anagnostopoulos, C.-N., Tsirtsis, G., Niros, A. D. and Spatharis, S. (2018). Optimized classification predictions with a new index combining machine learning algorithms, *International Journal on Artificial Intelligence Tools* **27**(03): 1850012. https://doi.org/10.1142/s0218213018500124.

Tien Bui, D., Ho, T. C., Revhaug, I., Pradhan, B. and Nguyen, D. B. (2014). *Landslide Susceptibility Mapping Along the National Road 32 of Vietnam Using GIS-Based J48 Decision Tree Classifier and Its Ensembles*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 303–317. https://doi.org/10.1007/978-3-642-32618-9_22.

Vaithiyanathan, V., Rajeswari, K., Tajane, K. and Pitale, R. (2013). Comparison of different classification techniques using different datasets, *International Journal of Advances in Engineering & Technology* **6**(2): 764.

Witten, I. H., Frank, E., Hall, M. A. and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann.

Wu, J., Pan, S., Zhu, X., Cai, Z., Zhang, P. and Zhang, C. (2015). Self-adaptive attribute weighting for naive bayes classification, *Expert Systems with Applications* **42**(3): 1487–1502. https://doi.org/10.1016/j.eswa.2014.09.019.

Yuan, C., Wu, Y., Qin, X., Qiao, S., Pan, Y., Huang, P., Liu, D. and Han, N. (2019). An effective image classification method for shallow densely connected convolution networks through squeezing and splitting techniques, *Applied Intelligence* **49**: 1–17. https://doi.org/10.1007/s10489-019-01468-7.

Yukselturk, E., Ozekes, S. and Türel, Y. K. (2014). Predicting dropout student: An application of data mining methods in an online education program, *European Journal of Open, Distance and E-Learning* **17**(1). https://doi.org/10.2478/eurodl-2014-0008.

Zahin, S. A., Ahmed, C. F. and Alam, T. (2018). An effective method for classification with missing values, *Applied Intelligence* **48**(10): 3209–3230. https://doi.org/10.1007/s10489-018-1139-9.

Zhang, J., Teng, Y.-F. and Chen, W. (2018). Support vector regression with modified firefly algorithm for stock price forecasting, *Applied Intelligence* **49**: 1–17. https://doi.org/10.1007/s10489-018-1351-7.

Zhang, Q., Wang, J., Lu, A., Wang, S. and Ma, J. (2018). An improved smo algorithm for financial credit risk assessment–evidence from china's banking, *Neurocomputing* **272**: 314–325. https://doi.org/10.1016/j.neucom.2017.07.002.

Zhu, J., Ge, Z., Song, Z. and Gao, F. (2018). Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data, *Annual Reviews in Control* **26**. https://doi.org/10.1016/j.arcontrol.2018.09.003.