Prediction of COVID-19 severity in Madrid region with Model-leaf trees

Miguel A. Riaño-Sánchez, Concha Bielza, Pedro Larrañaga
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
Campus de Montegancedo, Boadilla del Monte, Spain
ma.riano@alumnos.upm.es, {mcbielza, pedro.larranaga}@fi.upm.es

Abstract—The global pandemic caused by the Coronavirus Disease 2019 has affected society at severe levels. Among the major problems brought by it, hospital overcrowding was one of the most dangerous. This paper presents Model-leaf trees, a new machine learning classification algorithm that combines decision trees with other classification models at its leaves to create an enhanced solution. The model will be tested using COVID-19 data from 1632 patients from hospitals belonging to Fundación Jiménez Díaz, and one of the generated models will be interpreted and presented as a viable solution that could be used in a real scenario.

Keywords—Coronavirus; COVID-19; Predictive analytics; Machine learning; Supervised machine learning algorithm; Madrid; Tree-based model; Supervised classification

I. INTRODUCTION

The Coronavirus disease (COVID-19) [1] has severely impacted society over the past year. According to the World Health Organization Coronavirus Dashboard [2], 167492769 cases of COVID-19 have been reported and it has caused 3482907 deaths by May 26th, 2021.

In a study made by Roth et al. [3] it was discovered that one of the causes of the high mortality rate between March and May 2020 was hospital overcrowding due to lack of resources. This issue concerned hospitals at the beginning of the pandemic, and doctors voiced their complaints about it several times. For this reason, tools that could help them make decisions about which patients need admission and which ones do not are highly valued.

However, these tools must justify each suggestion, as the doctor’s decision is key to the patient’s life. Among the supervised machine learning algorithms, one stands out from the rest due to its simplicity, interpretability and understandability [4]: decision trees.

Decision trees have edges and nodes that generate a structure in a tree shape: a single root node at the top is divided into several nodes. Each node might also be divided into other nodes and so on, until each node is not divided anymore. Those final nodes are called leaves. Each internal node, that is, each node that is not a leaf and thus is divided, has an internal test over an attribute and it decides which branch to follow from that node. Leaf nodes contain class labels instead of a test [5]. This recursive partition of the instance space into smaller spaces [6] to solve the whole problem makes decision trees a divide-and-conquer algorithm.

The simplicity in its structure is what makes it highly interpretable, as each of these internal nodes can be translated into an if $x$ then $y$, else $z$ rule, which is intuitive for humans to understand, and the path an instance follows in the tree generates a set of rules that would explain why the tree made the final decision in its classification. This characteristic fits perfectly with the requirements of tools for doctors.

Examples of decision tree based proposals that were released during the past year are [7] in which the authors predict the severity of patients, or [8] in which the researchers predict their mortality rate.

However, decision trees suffer from the fragmentation problem: as each split is made, the data is divided based on the test and after several levels there is not much data to make decisions [9]. One of the solutions to this problem was a hybrid decision tree algorithm called: NBTree [9]. His creator noticed that the naïve Bayes classifier algorithm could be used at the leaves to perform the final classification, solving the problem. The result was a new algorithm that is able to improve the performance of decision trees.

The addition of classification models to decision trees has not always been approached by replacing the tree leaves with them. Gama and Brazdil [10] joined a supervised classification algorithm called Cascade Generalization with decision trees, naming the result CGTree. They thought about the possibilities that divide-and-conquer strategies could bring to Cascade Generalization, as the relations that could not be captured in the data at global level might be discovered by dividing the instance space. The result was another algorithm that improved the performance of Cascade Generalization by itself.

Two questions could be risen from these two works: could other classification models improve the performance of decision trees just as naïve Bayes does in NBTree? And, could other classification models be improved by applying a divide-and-conquer strategy to them? In this study we present Model-leaf tree (MLTree), a new supervised machine learning algorithm that combines both ideas to elaborate an enhanced...
solution. The idea is not only to improve any possible supervised classification model by dividing the whole problem into subproblems, but also to improve the decision trees themselves by solving the \textit{fragmentation problem} presented before. We will then apply this new algorithm to COVID-19 datasets in order to generate models that will help doctors in the admission of patients.

This research paper is divided into four additional sections: Section II reviews related solutions to this new algorithm. Section III presents MLI Tree and includes the algorithm explanation, its characteristics and functionalities. The experiments with the COVID-19 datasets and their results are shown in Section IV. Finally, Section V contains some conclusions about this research.

II. RELATED WORK

There have been many different proposals that were aimed at combining decision trees with other models in order to design an enhanced solution.

The NBTree [9], as mentioned above, is a fusion of decision trees and naive Bayes classifiers that combines the advantages of both models by replacing leaf nodes with naive Bayes classifiers.

A similar solution that uses this same idea of substituting leaf nodes with other models, is feedforward neural networks due to its powerful nonlinear processing ability, is called Hybrid Decision Trees (HDT) [11]. The algorithm uses the categorical features in the dataset to grow the tree and, in those cases in which a node cannot be split but the numerical features have diversity in their values, it is transformed into a feed forward neural network trained with the instances at the node.

In another proposal called Logistic Model Trees [12], authors substituted tree leaves with logistic regressors. However, instead of generating one model per leaf, they generated a group of them during the tree growth: in the root, they start training logistic regression models as long as it improves the fit to the data. Then, a split is made using the same criteria as the C4.5 algorithm. In each child they continue adding more logistic regression models, using the instances in the child node as data. This continues until the leaf, that will contain a committee of linear regression models that were trained on increasingly smaller subsets of data.

Decision trees were also used in Gil-Begue, Larrañaga and Bielza [13] to design a new algorithm called Multi-dimensional Bayesian Network Classifier Trees (MBCTrees), that has multi-dimensional Bayesian network classifiers in the leaves. It is a similar approach as NBTrees, but the authors recommend using a wrapper approach to generate them so that, when doing the split, the chosen candidate will be the one that best divides the data and the one that achieves the highest global accuracy.

As stated in the previous section, the addition of new models to decision trees has not always been approached by replacing the leaves with them. CGTree [10] incorporates a divide-and-conquer strategy to Cascade Generalization using decision trees so that relations that could not be found in each problem might be found in the subproblems.

Finally, in another research [14] on combining multiple classifiers that involves the usage of decision trees, the authors presented a new kind of decision tree algorithm called Meta Decision Trees (MDT), that are able to predict, for a dataset and a list of classification models, which algorithm should predict each sample.

III. THE ALGORITHM

The algorithm proposed in this paper is built on top of an implementation in Python of the decision tree CART algorithm [15] created by Joachim Valente [16]. Both include only the classification functionality, and they work with numerical and Boolean features (True or False).

The CART algorithm generates a binary decision tree. It is binary because each node is divided only into two nodes. For each feature, all the midpoints between adjacent training samples with different class value are possible splitting candidates. The algorithm performs a greedy splitting search while growing the tree, so that for each splitting candidate it chooses the one that minimizes the weighted sum of the node impurities in the two child nodes that would be generated. As impurity function, CART [15] uses the generalization of the binomial variance, or Gini index. This is repeated until a stopping criteria is met.

Two stopping criteria for the growing process are implemented in [16]: a hyperparameter that determines the maximum depth that the tree can reach and, when looking for the best split at a node, the weighted impurity of the children must be lower than the impurity in the current node. These two conditions would work as an early stopping (also called pre-pruning) method to prevent the tree from overfitting [17].

As the objective of this paper is to elaborate a decision tree algorithm enhanced with other classification models in some leaves, it is necessary to modify the growing process. The final algorithm can be found in Figure 1. Some important remarks will be explained below.

A. Operating options

The innovation of this algorithm is the possibility of choosing what type of classification model (\textit{extra model}) to put in the leaves. Thus, it offers three operating options:

1. One \textit{extra model}: the user can choose a classification model to enhance the decision tree.

2. Voting majority in model list: the user can choose a list of classification algorithms to enhance the decision tree. This list will be interpreted as a single \textit{extra model}, and the resulting prediction will be the majority vote of each individual classification model.

3. Best model in list: the user can choose a list of classification algorithms and, each time MLTree decides that a leaf should be turned into a model, it will choose the one that maximizes the accuracy.
with the instances that reached that leaf, and it will be considered the extra model in that node.

B. Calculation of the best splitting candidate

The calculation of the best splitting candidate in this new algorithm is the same as in the CART algorithm with one addition taken from the NBTree study: instead of only making sure that the average of the Gini index has to be lower than the impurity on the current node, in this new approach the reduction in the impurity should be at least of a 5%.

C. Calculation of the model accuracy

This new term, model accuracy, in the current node is just measuring the 5-fold cross-validation accuracy of the extra model trained in the node with the instances it has. The model accuracy in each child node is, nevertheless, calculated in a different way:

- If all the instances in the child node belong to the same class, the model accuracy in the child is 1.
- If there are fewer than 5 instances of each class label in the child node, its model accuracy is calculated as the number of instances that belong to the majority class label divided by the total number of instances.
- If the child node does not meet the previous two points, then the model accuracy is calculated as it was described above, with the 5-fold cross-validation accuracy.

After the model accuracy in each child node has been found, the weighted model accuracy in the children is calculated as the sum of the model accuracy of each one multiplied by the proportion of instances from the current node each of them has.

D. Stopping criteria

As the CART algorithm, this new approach has also some stopping conditions to improve the quality of the resultant tree by avoiding the node to be split. These conditions can be divided into two groups:

Group 1: Stopping conditions that generate a leaf
- All the instances belong to the same class label.
- In the node, there are fewer than 5 instances per class label.

As the model accuracy is calculated using the 5-fold cross-validation, the programming function that is being used requires at least one instance per class label in each fold, so at least there should be 5 instances per class label.

These stopping conditions are checked at the beginning of each iteration of the algorithm, before making calculations to generate the split. If any of them are met, the node is transformed into a traditional decision tree leaf, in which the predicted class label is the most repeated one among the instances in the node.

Group 2: Stopping conditions that generate an extra classification model
- The tree has reached the maximum depth.
- The best threshold to make the split was not found.
- The node has fewer than 30 instances.
- The model accuracy in the node is better than the weighted model accuracy in the children.

The first two conditions were already in the CART algorithm, as explained before. The last two, however, were taken from the NBTree [9]. The author considered that the split should not be done in the node if it has less than 30 instances. This is mainly due to the fact that the resulting nodes would have fewer instances and the possible extra models that could be trained might have low quality. Moreover, he also stated that the model accuracy in the current node should be lower than the weighted model accuracy in the children. This way, when a split is done, it is not only assuring that the resulting nodes will have less impurity in their instances, but also that a possible model trained with them will be better than a model trained in the current node with all the samples in it, which differences this new algorithm from other proposals like [11].

E. Class weights

Last, but not least, this new algorithm offers the user the possibility of assigning weights to each class label as an argument. Each weight must be a positive integer.

In a two-class label classification problem, the algorithm will prioritize the correct classification of the instances with the class label that has the highest weight. Consequently, instances with the other class label will be classified worse. This is achieved in this implementation by:

- Changing the way accuracies are calculated: the only accuracy that is not calculated using a Sci-kit learn [18] function is the one when a child node
has less than 5 instances per class label. In that case, when counting the number of instances for each class label when performing the calculation, each sample will count as their weight. That is, for each class label \( L \) with its weight \( W_L \), each instance with class label \( L \) will count as \( W_L \) instead of 1.

- Changing the training of the extra model: the weights will be passed as arguments to the Sci-kit learn [18] functions that generate each extra model, so that they are taken into consideration.

IV. EXPERIMENTS AND RESULTS

If it were possible to predict if a COVID-19 patient would evolve unfavorably or not, we could develop a tool that would help doctors in the emergency department decide whether a patient has to be admitted or not. In this section, the MLTree algorithm will be used to generate and to study models capable of achieving this.

A. Dataset description and preprocessing

To get a model that is able to distinguish which COVID-19 patients will evolve favorably and thus need admission and which ones will not, medical data on patients with this disease are needed. For this paper, we are grateful to have such dataset from hospitals belonging to Fundación Jiménez Díaz. It contains data from COVID-19 positive patients that were registered in the hospital at some point for any reason, like a consultation or admission. Therefore, it needed to be preprocessed in order to keep only the most useful data to elaborate the model that is needed.

When a patient arrives at the emergency department, doctors have only their personal information, their comorbidities and the measures taken for the triage. Then, if the patient is admitted, he/she usually undergoes a blood test. At the beginning of this pandemic, testing laboratories were as overcrowded as hospitals. Because of these facts, it was planned to generate two datasets: emergency, that includes only the available measures when a patient arrives at the hospital to generate models that could help doctors decide whether it is worth admitting the patient and ordering a blood analysis or not, and analysis, with the same attributes as emergency plus all possible blood analysis measures. The decided list of features to be extracted from the raw dataset included: personal information (age, sex and body mass index), comorbidities (like smoker and diabetes, among others), measures taken for the triage (first oxygen saturation and first body temperature) and several measures from blood analysis (like albumin or urea). To achieve this goal, some operations were performed over the data.

First, typical transformations in data analysis were made on the raw dataset: variable rename, generating useful columns (admission time to indicate how much time the patient stayed at the hospital and death and alive, to mark which patients passed away). Missing values were imputed using an iterative imputation with linear regressions. Afterwards, three important filtering steps would generate the final datasets.

The model needs to distinguish patients whose disease will evolve unfavorably from those that will not. For this paper, a patient is considered critical (the disease will evolve unfavorably) if he/she dies within 72 hours after their last test. A patient is considered non-critical (the disease will evolve favorably) if he/she does not die before 120 hours after their last test. So, the first filter only keeps patients that were admitted in the hospital and that have had at least one blood analysis and that meet the requirements above. Then the second filter kept only the list of features described before. This step left 69 attributes, of which 56 were blood test measures, which is a high number. A meta estimator that generates 100 random decision trees was fitted using the filtered dataset. A third filter kept only the blood analysis features that were used by the majority of models. This reduced the number from 56 to 23.

In order to fit classification models, the last step was to balance the class labels (critical and non-critical). After all this preprocessing, emergency has 13 features and analysis 36, and they both have instances from 1632 patients.

B. Performance evaluation of the algorithm

In this paper both datasets, emergency and analysis, will be used to test the MLTree algorithm performance. The 5-fold cross-validation accuracy will be used as metric for the evaluation.

For this analysis, MLTree will use four models taken from the Python module Sci-kit learn [18]: SVC (a C-support vector classification algorithm), KNN (a k-nearest neighbors vote classifier), LR (a logistic regression classifier that uses the solver liblinear) and BAG (a bagging meta-estimator that fits 20 SVC classifiers using a 75% of samples and features). The algorithm will also be evaluated with its second and third operation option using the first three classifiers presented before as a list of models.

The accuracies resulting of this performance evaluation can be observed in Table I. Each one is followed by its standard deviation (the number after the ± sign).

C. Comparison with other models

In order to assess how well the algorithm performs, a comparison with other models will be made:

- A comparison with the implementation of the CART algorithm in the Python module Sci-kit learn [18]. This way it can be assessed if the addition of extra classification models on the leaves improves its performance.

- A comparison with each extra model. This way, it will be checked if the usage of a divide-and-conquer algorithm to separate the instances for the extra model gets a better result than having the model alone. This last comparison will be made only with the results obtained using the first operating option of MLTree.

To compare the performance evaluations, the 5-fold cross-validation accuracies of the CART algorithm [18] and each classifier that was used in the previous section will be used.
against their correspondent result from the performance evaluation of MLTree in a hypothesis test. Due to its low Type I error, Dietterich [19] proposed the usage of the McNemar’s test for this situation. Its null hypothesis is that “the two models have the same error rate”. The accuracies resulting of the performance evaluation using each model alone can be observed in Table II. Each one is followed by its standard deviation (the number after the ± sign). The p-values resulting from the comparison between the CART models and the MLTree models can be found in Table III. The ones resulting from the comparison between each classification model alone and its usage in MLTree can be found in Table IV.

Using a significance level of 0.05, the null hypothesis is rejected in almost all the cases when comparing MLTree with CART. So, excepting the case when KNN is used with analysis dataset in MLTree, it can be stated that the CART algorithm generates significantly different models than MLTree, being better the results by the new algorithm according to the accuracies obtained. The only case where the null hypothesis cannot be rejected might have happened due to an overfit of the data, which caused the CART algorithm to outperform the MLTree algorithm.

The results when comparing the performance evaluation of

<table>
<thead>
<tr>
<th>Table I. MLTree Performance Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dataset name</strong></td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td><strong>Extra model: SVC</strong></td>
</tr>
<tr>
<td><strong>Extra model: KNN</strong></td>
</tr>
<tr>
<td><strong>Extra model: LR</strong></td>
</tr>
<tr>
<td><strong>Extra model: BAG</strong></td>
</tr>
<tr>
<td><strong>Dataset name</strong></td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td><strong>Extra model: SVC</strong></td>
</tr>
<tr>
<td><strong>Extra model: KNN</strong></td>
</tr>
<tr>
<td><strong>Extra model: LR</strong></td>
</tr>
<tr>
<td><strong>Extra model: BAG</strong></td>
</tr>
</tbody>
</table>

**D. Interpretation of a MLTree model**

The ultimate goal of the design of this algorithm is to find a better solution that can help doctors make decisions about COVID-19 patients that arrive to the emergency department. We will make an interpretation of a MLTree model trained with logistic regression as extra model and using the dataset emergency to study if it could be used in this situation. Its visual representation is found in Figure 2.

If the features the model has chosen to divide the dataset are studied, it can be assessed that they match reality. This model is saying that patients that are below 54 years old (feature Age) and that have an oncologic disease (variable Onco) can be classified as critical (class label True) and they should be admitted, or at least it is worthwhile for the doctor to order a blood test. Patients that are in the same age range and do not have any oncological diseases are classified using a LR model, whose feature coefficients can also be interpreted (a higher coefficient would mean that that feature collaborates in increasing the probability for a patient to be critical). This first logistic regression model has assigned high positive coefficients to two comorbidities: Smoker and Diabetes.

Going down the other branch from the root, if the patient is 54 years old or more, then another LR model will classify the instances. In this case, it has given high coefficients to the variables Cardio (that indicates if the patient has any cardiovascular disease) and Onco. It is interesting to remark that this logistic regressor has identified the variable Onco as important. The decision tree algorithm could identify it as important as well in another branch, but not in this one, showing the improvements that MLTree brings by adding other classification models to tree leaves.

The age and comorbidities values chosen by MLTree are conditions recognized as risk factors in COVID-19 patients by institutions like the Centers for Disease Control and Prevention [20][21]. This exhibits that MLTree has extracted from the data
information that matches reality and it demonstrates that the model could be used by doctors in their hospitals.

V. CONCLUSION

The main objective of this paper was to contribute to the fight against the pandemic caused by the COVID-19 disease. We presented MLTree, a new machine learning algorithm that generates a decision tree extended with other classification algorithms in some of its leaves to improve the accuracies obtained using each algorithm by itself. We showed that MLTree is able to generate good solutions that can help doctors in their process of deciding which patients that come to the emergency department need an admission and which ones can stay at home. It was also demonstrated that MLTree generates models that are significantly different from the ones a common decision tree would create, and the accuracies calculated showed that it is possible that they are outperforming them. Moreover, in the majority of the experiments performed, MLTree generates models that are at least equally and sometimes even better than the ones each base classification algorithm would generate by itself. Finally, we proposed an MLTree model with one of the configurations used in the experiments as a viable solution, and an interpretation of it was given in which we assessed that it matched reality and thus it could be used in a real scenario. Moreover, this interpretation demonstrates that, as long as MLTree uses other interpretable classification algorithms, it keeps the high understandability that is brought by decision trees.

This new tool to help in the fight against COVID-19 considers a hypothetical situation in which the cost of making an error in the prediction is the same for all class labels. However, the cost of sending an ill patient home might be higher than admitting a healthy one. A proper continuation of this work could be the discussion of a cost function to assign weights to each label: critical and non-critical, in order to make the model to classify with less error one of them.

As MLTree is based on the Classification and Regression algorithm (CART) [15], possible future research could also be aimed at adding the functionality to make it able to solve regression problems.

Finally, even though hospital overcrowding isn’t a major issue anymore, we highly believe that the algorithm and the approach presented in this paper could be used as an inspiration to solve similar situations in the future that may occur with other diseases.

ACKNOWLEDGMENTS

This work has been partially supported by the project funded by Fundación BBVA: “Outcome prediction and treatment efficiency in patients hospitalized with COVID-19 in Madrid: A Bayesian network approach”, in which the dataset preprocessing was performed, by Fundación Jiménez Díaz, that provided us with the necessary data for this study, and by “Red Temática sobre Inteligencia Artificial en Biomedicina (RED2018-102312-T)” of the Ministry of Science and Innovation.

REFERENCES