Towards an Optimized Decision Tree Model for COVID-19 Prediction

Nadia Daoudi[1], Nour Elhouda Youcefi[1], Noureddine Aribi[1] and Ramzeddine Belaoudmou[2]
[1]*Université Oran1, Lab. LITIO, 31000 Oran, Algeria*
{nadia_daoudi, youcefi.nourelhouda}@outlook.com, aribi.noureddine@univ-oran1.dz
[2]*CHU Lamine Debaghine, Alger, Algeria, consultant@appli-med.com*

***Abstract* — The coronavirus pandemic has had a dramatic impact on worldwide healthcare and economic systems. Interestingly, a lot of interest has been devoted to Machine learning technological innovations, such as Decision Trees, for promoting reliable decision aid support. In this paper, we propose a decision tree-based learning approach to predict COVID19 infections at its earlier stage and to improve the organization of care and patient follow-up. We show how this approach can be exploited in association with a cross-platform mobile application to provide an operational proof-of-concept. Experiments performed on a real COVID-19 dataset show the efficiency of our approach and its significant advantages in a healthcare context.**

Keywords— Machine Learning, Classification, Decision Tree, C4.5, Coronavirus, COVID-19, Medical diagnosis, Optimization, Pruning, Decision aid tool, Cross-platform mobile application.

I. INTRODUCTION

MACHINE learning is a technological innovation that changed the world with its ability of prediction and decision making done through a set of tools and techniques belonging to its various types. This branch of Artificial Intelligence (AI) relies on the acquisition of new knowledge or skills based on what has been experienced in the past, in order to create generalizable models that can make predictions, one of these most popular models is the decision tree (DT). Specifically, tree-based learning falls under the category of supervised learning. It can be used to solve both regression and classification problems in the form of a tree structure. These abilities of classification and prediction, made decision trees a very useful technology for many real-life areas, including fraud detection, target marketing and healthcare systems (e.g. predicting diseases, improving the decision aid support). At the beginning of 2020, the coronavirus spread across the world and became a global pandemic. This inspired us to apply machine learning with decision trees in the medical field with the aim of helping doctors in their challenging tasks. However, it is well recognized that the construction of an optimal decision tree is an NP-hard problem [12]. Interestingly, several greedy algorithms have been proposed in the literature, which generate an approximate decision tree with higher accuracy, though optimality could not be guaranteed. Among the algorithms, we cite CART [9, 11], the ID3 algorithm of Quinlan in 1986 [6], and C4.5 (then C5.0) of Quinlan in 1993 [5, 13]. From an experimental point of view, these algorithms are very efficient and allow the rapid construction of decision trees that predict with great reliability the class of new data.

In this paper, we propose a decision aid model based on optimized decision trees tailored for COVID-19 prediction in the field of medical diagnosis. For higher performance and prediction accuracy, we exploit the well-known C4.5 algorithm that uses a greedy, top-down, recursive partitioning strategy for growing a decision tree, in combination with a pruning method to remove outlier branches from the grown tree. Our predictive approach was embedded into an operational proof-of-concept, which was tested on real datasets.

This paper is organized as follows. Section II recalls preliminaries. Section III introduces an optimization technique exploiting the decision trees pruning in order to improve the prediction capabilities, while ensuring good learning accuracy. Section IV illustrates our learning approach on a running example. We discuss related work in section V. Section VI reports our experiments on a real-world dataset about Coronavirus Pandemic (COVID-19). Section VII concludes the paper.

II. INDUCTIVE DECISION TREE LEARNING ALGORITHM

The classification is a Data mining technique that aims to build a model of a function for calculating (predicting) the class attribute of a data from the other attributes. Classification follows a two-step process, learning step and prediction step. In the learning step, the model is inductively built based on given training data. In the prediction step, the model is used to predict the response for given data. Decision Tree is one of the easiest and popular classification algorithms. The decision tree is built recursively by partitioning the training dataset into subsets. Let S be the set of data from the training set, A the set of attributes (or features), and y the class attribute (*i.e.* label or target feature). Our approach is based on the C4.5 greedy[1] algorithm to build a decision tree from a set of training data. We stress that unlike the ID3 algorithm, the C4.5 attributes may be continuous or unknown.

The decision tree construction method is provided in Algorithm 1. The key steps involved in the DT learning algorithm are as follows:

[1] As it focuses only on the current node, while the other nodes are being ignored

1) Select the best attribute using the $SplitCriterion$ method to split the records into the best possible manner (line 5). This method relies on a heuristic that aims to reduce the number of tests that are needed to classify the given tuple. Most popular selection measures are *Information Gain*, *Gain Ratio*, and *Gini Index* discussed below.
2) Make that attribute a decision node and breaks the dataset into smaller subsets (lines $7-10$).
3) Starts tree building by repeating this process recursively for each child until the stopping criterion is satisfied (line 2). The obvious strategy is to stop once there are no more remaining attributes or the data set covered by the attribute node belongs to the same class. We will also consider the strategy of not continuing the partitioning to avoid a given cost, while emphasizing a certain level of performance and accuracy. The $DecisionTreePruning()$ method (line 12) relies on overfitting avoidance technique in order to produce a robust prediction in the presence of noise.

Algorithm 1: DecisionTreeGrowing $(S, A, y, SplitCriterion, StopCriterion)$

Input: S: training set; A: input feature set; y: target feature
$SplitCriterion$: a method to evaluate a split;
$StopCriterion$: the stopping criteria of the growing process
Output: T: a decision tree

1: Create a new tree T with a single (root) node;
2: **if** $StopCriterion(S)$ **then**
3: mark T as a leaf node with the most common value of y as a label;
4: **else**
5: Find $a \in A$ with the best $SplitCriterion(a_i, S)$;
6: Label t with a;
7: **foreach** outcome $v_i \in a$ **do**
8: $Subtree_i \leftarrow DecisionTreeGrowing(\sigma_{a=v_i}S, A, y)$;
9: Connect the root note of t_T to $Subtree_i$ with an edge labeled as v_i;
10: **end**
11: **end**
12: **return** $DecisionTreePruning(S, T, y)$;

Building a DT is computationally inexpensive, making it possible to handle very large training set. Furthermore, once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of $O(w)$, where w is the maximum depth of the tree.

A. Attribute selection criteria:

The main intuition behind the choice of an attribute is that we try to minimize the heterogeneity at each node (*i.e.* Impurity): the data which reaches a certain node of the decision tree must be more homogeneous than the data reaching an ancestor node. For this, we need to be able to measure the homogeneity of a dataset. Shannon [2] introduced the concept of *entropy* to measure the *impurity* of the input set. In physics and mathematics, entropy referred as the randomness or the impurity in the system. In information theory, the entropy is a measure of the amount of uncertainty in a dataset. So, the more impurity the dataset is, the greater the entropy (or uncertainty) is. The decision tree algorithms apply this theory. Given a training set S, the probability vector of the target attribute y is defined as follows [14]:

$$P_y(S) = \left(\frac{|\sigma_{y=c_1}S|}{|S|}, \ldots, \frac{|\sigma_{y=c_{dom(y)}}S|}{|S|}\right) \quad (1)$$

where $\sigma_{a_i=v_{i,j}}S$ returns the records with the attribute value $a_i = v_{i,j}, v_{i,j} \in dom(a_i)$ and $\frac{|\sigma_{y=c_i}S|}{|S|}$ is the probability of an object being classified to a particular class.

1) Information Gain. The Information gain is an impurity criterion (proposed by [8]) based on the measure of entropy. It is given by the following formula:

$$Gain(a_i, S) = Entropy(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{|\sigma_{a_i=v_{i,j}}S|}{|S|} \times Entropy\left(y, \sigma_{a_i=v_{i,j}}S\right) \quad (2)$$

where

$$Entropy(y, S) = \sum_{c_i \in dom(y)} -\frac{|\sigma_{y=c_i}S|}{|S|} \times log_2\left(\frac{|\sigma_{y=c_i}S|}{|S|}\right) \quad (3)$$

Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. Intuitively, having subsets with minimal entropy is interesting. It is natural to sum these entropies by weighting them according to the proportion of records in each subset.

2) Gini Index. The Gini index (*i.e.* gini impurity [6]) measures the divergence between the probabilities of the values of the target feature.

$$Gini(y, S) = 1 - \sum_{c_i \in dom(y)} \left(\frac{\left|\sigma_{y=c_i}S\right|}{|S|}\right)^2 \quad (4)$$

Using this measure, the attribute selection is made by taking the maximum of the following measure:

$$GiniGain(a_i, S) = Gini(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{|\sigma_{a_i=v_{i,j}}S|}{|S|} \times Gini\left(y, \sigma_{a_i=v_{i,j}}S\right) \quad (5)$$

3) Gain Ratio Information gain is biased for the attribute with many outcomes. In order to solve this problem, C4.5 algorithm, uses an extension to *information gain* known as the *gain ratio*. It handles the issue of bias by takes the number and size of branches into account when choosing an attribute. It is defined as follows:

$$GainRatio(a_i, S) = \frac{Gain(a_i, S)}{Entropy(a_i, S)} \quad (6)$$

III. Decision Tree Pruning

Pruning is a machine learning technique that removes parts of a decision tree, which obtains the least amount of information in order to reduce the size of the tree. Thus, pruning aims to reduce the rate of prediction error. It is achieved through replacing a complete sub-tree with a leaf node, and the criteria of the process depend on the course of events, which allow to estimate the true flaw in that particular sub-tree. The algorithm for pruning is as follows:

Algorithm 2: DecisionTreePruning(S, T, y)

Input: S: training set; y: target feature
1: **InOut:** T: the tree to be pruned
2: **do**
3: Select a note $t \in T$ such that pruning it maximally improve some evaluation criteria;
4: **if** $t \neq \emptyset$ **then**
5: $T \leftarrow pruned(T, t)$
6: **end**
7: **while** $t \neq \emptyset$;

There are two types of decision tree pruning [10]:

- **Pre-pruning:** if the tree stops growing before all examples of the training set are well classified.
- **Post-pruning:** if the model is built entirely, pruning the elements that contribute to excessive learning.

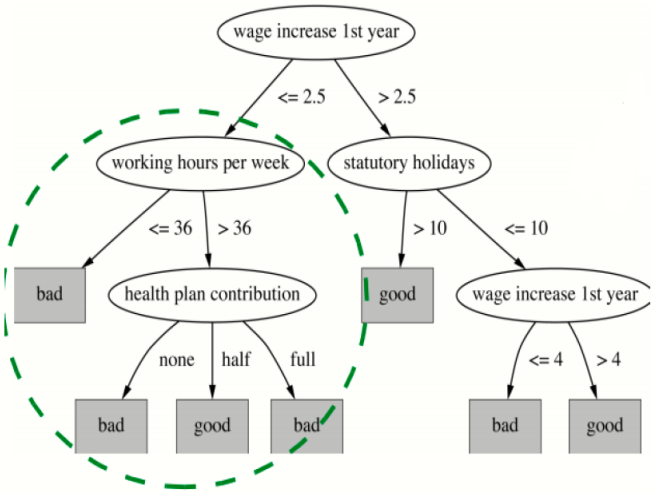


Fig. 1. Pruning the decision tree: sub tree replacement [10]

Assuming the tree shown in the figure above, after examining all of its sub trees. By observing that the gain for a leaf exceeds that of the whole sub-tree, it is replaced by the leaf "bad".

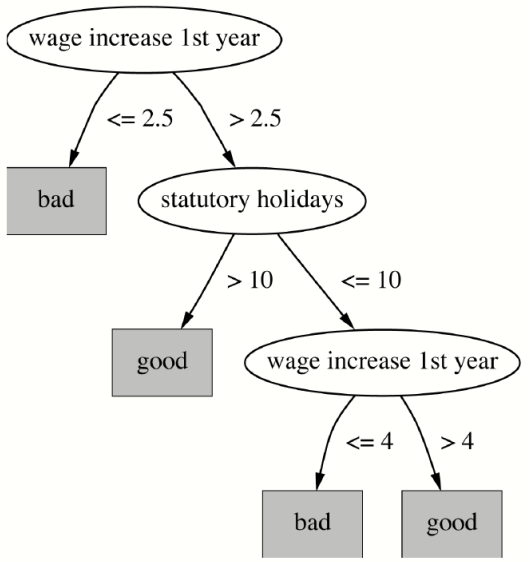


Fig. 2. Pruning the decision tree: new tree [10]

IV. Running Example

This part represents an example of a decision tree generation using the C4.5 algorithm. The example takes the dataset collected in a period of two weeks to decide if the weather is suitable for playing tennis, taking into account the following attributes: outlook, temperature, and wind, which have nominal values, as well as humidity, which has continuous values.

TABLE I
Dataset S of the playing tennis example [13]

Day	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	85	Weak	No
2	Sunny	Hot	90	Strong	No
3	Overcast	Hot	78	Weak	Yes
4	Rain	Mild	96	Weak	Yes
5	Rain	Cold	80	Weak	Yes
6	Rain	Cold	70	Strong	No
7	Overcast	Cold	65	Strong	Yes
8	Sunny	Mild	95	Weak	No
9	Sunny	Cold	70	Weak	Yes
10	Rain	Mild	80	Weak	Yes
11	Sunny	Mild	70	Strong	Yes
12	Overcast	Mild	90	Strong	Yes
13	Overcast	Hot	75	Weak	Yes
14	Rain	Mild	80	Strong	No

The present steps show the construction of the decision tree:

1) **Calculation of Entropy:**
The entropy of the dataset S is given as follows:

$$Entropy(S) = -\frac{9}{14} \times log_2\left(\frac{9}{14}\right) - \frac{5}{14} \times log_2\left(\frac{5}{14}\right)$$
$$= 0.94$$

2) **Calculation of the information gain:**
The next step is to calculate the information gain for the four attributes in order to find the attribute which

has the highest value. This attribute will be the root node.

Calculation of entropies:

$$Entropy(S_{Sunny}) = -\frac{2}{5} \times log_2\left(\frac{2}{5}\right) - \frac{3}{5} \times log_2\left(\frac{3}{5}\right) = 0.9710$$

$$Entropy(S_{Rain}) = -\frac{3}{5} \times log_2\left(\frac{3}{5}\right) - \frac{2}{5} \times log_2\left(\frac{2}{5}\right) = 0.9710$$

$$Entropy(S_{Overcast}) = -\frac{4}{4} \times log_2\left(\frac{4}{4}\right) - 0 \times log_2(0) = 0$$

Calculation of the information gain:

$$Gain(S, Outlook) = Entropy(S) - \frac{5}{14} \times Entropy(S_{Sunny}) - \frac{5}{14} \times Entropy(S_{Rain}) - \frac{4}{14} \times Entropy(S_{overcast}) = 0.94 - \frac{5}{14} \times 0.9710 - \frac{5}{14} \times 0.9710 - \frac{4}{14} \times 0$$

$$Gain(S, Outlook) = 0.2467$$

As well we find for the other discrete attributes:

$$Gain(S, Temperature) = 0.0292$$
$$Gain(S, Wind) = 0.0481$$

For an attribute of continuing value such as humidity, the C4.5 algorithm manages the calculation of information gain as follows:

1) The first step is to sort the humidity attribute values, from the smallest to the largest, and to remove duplicate values. The set of sorted values is as follows:
$$\{65, 70, 75, 78, 80, 85, 95, 96\}$$

2) As a second step, after sorting all the examples in ascending order for the attribute humidity, C4.5 proposes to perform a binary division for each of value A_i, taking all instances lower than or equal to the current value $A_j \leq A_i$, and also all instances higher than the current value $A_i > A_j$. Then, it calculates the gain and finds the value that brings the best gain which would be the threshold.

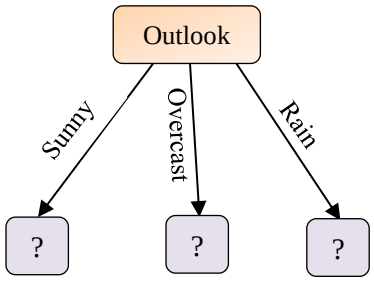
It applies formula (1) to obtain the entropy and formula (2) to get the information gain. The table II summarizes the gain calculation for the attribute continuous humidity.

Therefore, the value which brings the best gain is associated to the threshold which is equal to 80 for humidity.

$$Gain(S, Humidity) = 0.102$$

According to the comparison, the attribute "Outlook" with the highest value of information gain is the root node of the tree structure.

This root has three values: *"Sunny"*, *"Overcast"* and *"Rain"*. At this point, the tree will be divided into three sub-trees.



All the steps of the partition are applied recursively on the new subsets, and, in order to form a tree structure, all these nodes become leaves.

- **The first sub-tree associated with outlook as "Sunny":**

Day	Temperature	Humidity	Wind	Play
1	Hot	85	Weak	No
2	Hot	90	Strong	No
8	Mild	95	Weak	No
9	Cold	70	Weak	Yes
11	Mild	70	Strong	Yes

Calculation of entropy:

$$Entropy(S_{Sunny}) = -\frac{2}{5} \times log_2\left(\frac{2}{5}\right) - \frac{3}{5} \times log_2\left(\frac{3}{5}\right) = 0.9710$$

Calculation of the information gain:

$$Gain(S_{Sunny}, Temperature) = 0.5710$$
$$Gain(S_{Sunny}, Wind) = 0.0200$$
$$Gain(S_{Sunny}, Humidity) = 0.971$$

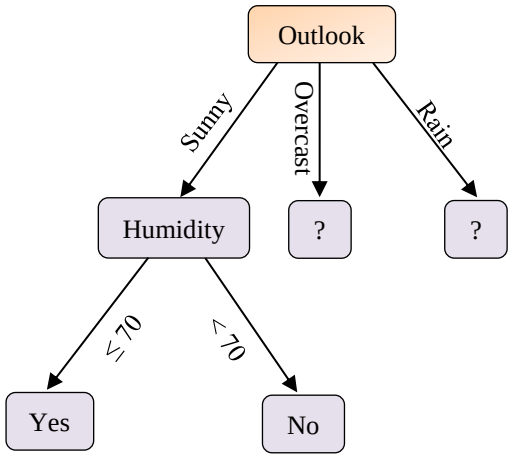
After comparison, humidity produces the best information gain, in case the outlook is sunny.

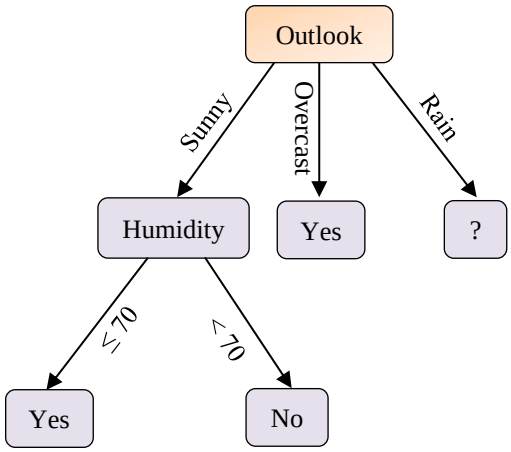
Thus, when the value of the humidity attribute is less than or equal to 70, the decision will correspond to the class "Yes".

Finally, when the value of the humidity attribute is greater than 70, the decision will correspond to the class "No".

TABLE II
GAIN CALCULATION FOR THE ATTRIBUTE CONTINUOUS HUMIDITY USING C4.5 ALGORITHM

Humidity	**65**		**70**		**75**		**78**		**80**		**85**		**90**		**95**		**96**	
Interval	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	1	8	3	6	4	5	5	4	7	2	7	2	8	1	8	1	9	0
No	0	5	1	4	1	4	1	4	2	3	3	2	4	1	5	0	5	0
Entropy	0	0.96	0.81	0.97	0.72	0.99	0.65	1	0.76	0.97	0.88	1	0.92	1	0.96	0	0.94	0
Gain	0.048		0.015		0.045		0.09		**0.102**		0.025		0.010		0.048		0	





- **The second sub-tree associated with outlook as "Overcast":**

Day	Temperature	Humidity	Wind	Play
3	Hot	78	Weak	Yes
7	Cold	65	Strong	Yes
12	Mild	90	Strong	Yes
13	Hot	75	Weak	Yes

Calculation of entropy:

$$Entropy(S_{Overcast}) = -\frac{5}{5} \times log_2\left(\frac{5}{5}\right) - \frac{0}{5} \times log_2\left(\frac{0}{5}\right)$$

$Entropy(S_{Overcast}) = 0 \Rightarrow$ So this sub-tree is perfectly classified

As a result, when the "Outlook" attribute is "Overcast", all examples belong to the positive class, so it returns the leaf node "Yes".

- **The second sub-tree associated with outlook as "Rain":**

Day	Temperature	Humidity	Wind	Play
4	Mild	96	Weak	Yes
5	Cold	80	Weak	Yes
6	Cold	70	Strong	No
10	Mild	80	Weak	Yes
14	Mild	80	Strong	No

Calculation of entropy:

$$Entropy(S_{Rain}) = -\frac{3}{5} \times log_2\left(\frac{3}{5}\right) - \frac{2}{5} \times log_2\left(\frac{2}{5}\right)$$

$$= 0.9710$$

Calculation of the information gain:

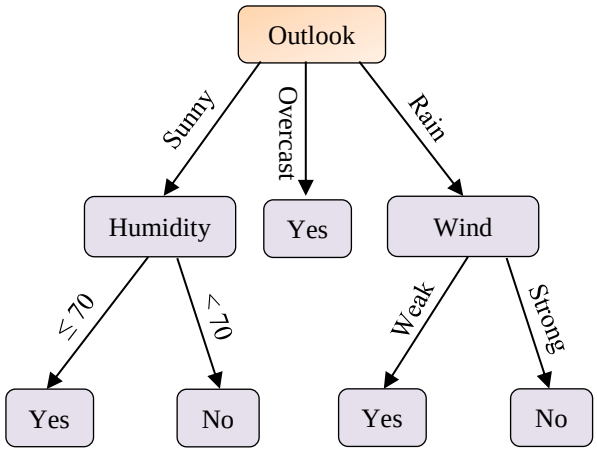
$$Gain(S_{Rain}, Temperature) = 0.0200$$
$$Gain(S_{Rain}, Wind) = 0.9710$$
$$Gain(S_{Rain}, Humidity) = 0.322$$

After comparison, wind produces the best information gain, in case the outlook is rain.

Thus, when the value of the humidity attribute is weak, the decision will correspond to the class "Yes".

Finally, when the value of the wind attribute is strong, the decision will correspond to the class "No".

The DT procedure (see Algorithm 1) lasts until the following decision tree is generated:



The decision tree can also be expressed in the form of decision rules:

If *Outlook* = *sunny and Humidity* < 70 ***Then*** *Play* = *no*

If *Outlook* = *sunny and Humidity* ≤ 70 ***Then*** *Play* = *yes*

If *Outlook* = *overcast* ***Then*** *Play* = *yes*

If *Outlook* = *rain and Wind* = *weak* ***Then*** *Play* = *yes*

If *Outlook* = *rain and Wind* = *strong* ***Then*** *Play* = *no*

V. Related Work

There are few studies that focused on predicting the presence of COVID-19 for a given human being. However, in this section, we concentrate on the research works related involving the use of a decision tree algorithm to predict positive Covid-19 cases. Riana *et al.* [4] introduced a supervised approach to make a diagnosis of surveillance classification using the C4.5 algorithm. The results indicated that the COVID-19 surveillance diagnostic was successfully modeled into a decision tree classification using the C4.5 algorithm. The accuracy rate of the testing process in a confusion matrix with three classes is equal to 92.86%. Another C4.5 machine learning classifier is also used in the work of Nanda *et al.* [3]. According to the results, the algorithm helps in obtaining a 75% accuracy rate. Our work is different in the sense that it consists to build a robust prediction model using an optimized decision tree algorithm. The optimization step allows having a classifier with the best compromise between the learning accuracy and prediction generalization for the unseen tuples.

VI. Experimental Evaluation

In this section, we experimentally evaluate our fast learning framework for predicting COVID-19 infections.

A. Experimental protocol:

Our experiments were conducted using real datasets and according to the following protocol:

1) **Hardware environment.** Our DT model was trained and tested on a PC with an Intel Core i5 processor, 2.70 GHz and 8GB memory and with Windows 10, 64-bit operating system. Besides, our mobile application (which embeds the resulting DT prediction model in the form of decision rules) was deployed and tested on a mobile phone Samsung J4 having ARMv7-A CPU, 2GB memory, and 32GB storage capacity.
2) **Software environment.** Our Decision Tree-based classifier was implemented using the well-known Scikit-Learn python library[2] using the Gini impurity criterion [6], whereas the cross-platform mobile application was implemented using a couple of web development frameworks (Ionic.v5, Angular.v8, NodeJs, Cordova, SQLite) and programming languages, namely, TypeScript, HTML5, CSS, SQL. The full source code is publicly available here [1], whereas the dataset can be provided upon request.
3) **Benchmark dataset.** We have considered a dataset about real 270 PCR (Polymerase Chain Reaction) tests for COVID-19 provided by the university hospital center (CHU) of Algiers (the capital of Algeria). Several information were collected from each patient, such as *AGE*, *FIEVRE*, *ECOULEMENT*, *CEPHALEES*, *IRRITABILI*, ASTHENIE, *DIARRHEE*, *TABAC*, *DIABETE*, etc. The class label is given by the PCR result that we which to learn inductively.

   Before building our model, we went through an essential step in the success of any machine learning project, which is data *cleaning* and *preprocessing*. The cleaning step consisted to eliminate some lines (patients) having some missed values. In the preprocessing step, we verified that there is not some classes that dominate the dataset, as this may most likely lead to biased trees. So, it is highly recommended to balance the dataset prior to fitting with the decision tree. Next, we proceed with a cross-validation procedure and divide the data set into 80% for the training dataset and 20% for testing the model.
4) **Decision tree optimization.** There are two kinds of misclassification: ***(i) Training error***, which is the number of misclassification errors committed on training records, and the ***(ii) Generalization error***, which is the expected error of the model on previously unseen records. So, a good model must have low training error as well as low generalization error so as to avoid undesirable situations, namely model (under or) overfitting. We have used the built-in Scikit-learn pre-pruning optimization to produce a DT with the best trade-off between both training and generalization errors.

[2] https://scikit-learn.org

5) **Model accuracy evaluation.** To assess the effectiveness of our classification model we use a confusion matrix, which is a contingency table that measures the performance of a machine learning model, typically a supervised learning algorithm. It allows analyzing the frequency with which the results of predictions are accurate compared to the reality in classification problems. We also recall the four main terminologies used to define quality metrics related to the prediction capability of an actual COVID-19 infection, which correspond to:

- TP (*True Positive*): the number of patients correctly classified as carriers of the COVID-19 virus.
- TN (*True Negative*): the number of correctly classified patients who are in good health.
- FP (*False Positive*): the number of misclassified patients who are in good health.
- FN (*False Negative*): the number of misclassified patients carrying the COVID-19 virus.

B. Experiment results

Results given by our optimized decision tree. In order to qualitatively assess our approach, we summarize the prediction results of our decision tree classifier (see figure 5) using 30 new patients, and then compare them to the expected values using the following confusion table:

	ACTUALLY: +(1)	ACTUALLY: –(0)
PREDICTED: +(1)	TP = 21	FP = 3
PREDICTED: –(0)	FN = 1	TN = 5

Based on this table, we can extract enough information to calculate various performance statistics including **precision** and **recall** metrics.

- **Precision** is the frequency in which the number of the predicted positive class actually belongs to the positive class.

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

- **Recall** which is sometimes referred to as ``sensitivity", is a metric that quantifies the number of correctly identified positive predictions among all of the positive predictions that could have been made.

$$Recall = \frac{TP}{TP+FN} \quad (8)$$

For our approach, we obtained the following results:

- Precision = 0.87
- Recall = 0.95

Additionally, we use the **harmonic mean** which combines both precision and recall into a single score. Formally, it is defined as follows:

$$F1\ Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (9)$$

In our case the value of *F1 Score* is equal to 0.91.

Overall, we found that the evaluation results are quite good in terms of being able to better distinguish patients carrying the COVID-19 virus, from those who are healthy by reaching the value of the *F1 Score* of 0.91, where the ideal value for *F1 Score* is 1 and the worst value is 0.

Results using a classical decision tree. The main goal of this section is to show to what extent the optimization step allows us to avoid the model overfitting problem, while maximizing the prediction generalization of the proposed classifier. To this end, we have performed the same experiments on the preprocessed COVID-19 dataset, and according to the same cross-validation procedure, except that we do not consider the optimization step when producing the decision tree classifier. In this case, we obtained the following results: (i) A **precision** value equals to 0.63 instead of **0.87** for the optimize DT; (ii) A **recall** value of 0.54, whereas the optimized classifier provides a value of **0.95**, which is quite superior compared to the classical version; and finally (iii) The F1 Score is equal to 0.56, which remains a poor result compared to our optimized decision tree, where the *F1 Score* was equal to **0.91**.

These results can be explained by the fact that the classical decision tree classifier tries to be as perfect as possible, by increasing the decision tree depth of required tests in order to minimize the classification error (in the training phase). However, this may lead to very complex decision trees and most probably ends with overfitting (see figure 6), with a severe lack in terms of prediction capabilities (do not generalize the data well). Unlike the classical model, the optimized classifier tends to reduce the required information to classify the tuples (i.e. the number of tests that are needed to classify a given tuple) using a subset of attributes having the highest information gain.

C. Proof-Of-Concept: A decision Aid Mobile Application for COVID-19 prediction.

In the context of a healthcare emergency, we opted for optimized decision trees (with pruning) in order to effectively tackle the COVID-19 pandemic (in Algeria country), using the Sci-kit Learn python library and a handy mobile application. As explained above, we have used, in our experiments, real COVID-19 datasets provided by the CHU (university hospital) of Algiers (the capital city of Algeria). The resulting DT is provided and illustrated in [1].

The first page of our mobile application represents the graphical interface of the home page, it gives the user an idea of the different functionalities of the application, see figure 3. Our cross-platform mobile application exploits the resulting decision tree classifier described in sections II and III.

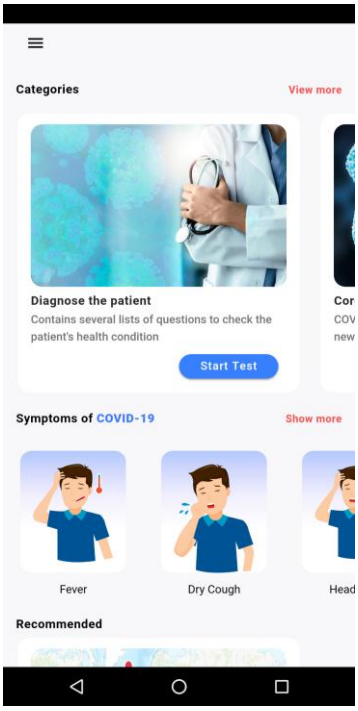


Fig. 3. Home page of the COVID-19 application.

We have designed and developed jointly with a medical expert a form page to collect the data needed from a patient to analyze and predict infection by COVID-19, see figure 4.

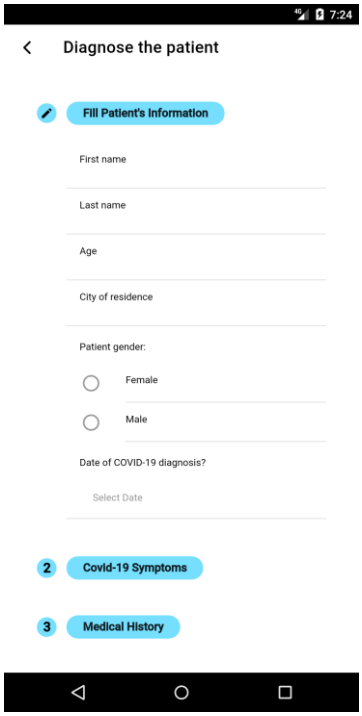


Fig. 4. A form page to gather data from a patient

- First, the user (doctor) of our mobile application will fill in the information related to a patient. In the meantime, we display error messages whenever the doctor makes a mistake in this step.
- The second step involves a set of questions about the most common symptoms of COVID-19 (e.g. '*FIEVRE*', '*ECOULEMENT*', '*CEPHALEES*', '*IRRITABILI*', 'ASTHENIE', '*DIARRHEE*', '*TABAC*', '*DIABETE*', etc.).
- The third step focuses on a patient's medical history, which includes illnesses (chronic, serious), as well as other complementary questions in order to define a specific risk factor and make a more fine-grained decision.

VII. Conclusion

In this paper, we have proposed a fast operational machine learning approach based on optimized decision trees in order to mitigate the effects and the spread of the COVID-19 pandemic crossing international boundaries, while realizing a good compromise between prediction accuracy and generalization. Experimental results on a real dataset have demonstrated that our DT-based approach is well suited to predict potential COVID-19 infection with high accuracy. The proposed approach is packaged as a proof-of-concept cross-platform mobile application for predicting a possible COVID-19 infection, thereby minimizing the potential burden of the pandemic on our society.

References

[1] N. Youcefi, N. Daoudi, and N. Aribi. Source code: https://drive.google.com/drive/folders/1NFXw3trX43kX2IOB8VJ27Bi6FuSQbJbi?usp=sharing, 2022. Accessed: 2022-06-15.
[2] C. E. Shannon. A mathematical theory of communication. Bell *System Technical Journal (juillet et octobre 1948)*, 27:379–423/623–656, 1948.
[3] C. R. Panigrahi, B. Pat, M. Rath, T. Weng and S. Nanda. Covid-19 risk assessment using the c4.5 algorithm. 2021.
[4] D. Riana and W. Wiguna, Diagnosis Of Coronavirus Disease 2019 (COVID-19) Surveillance Using C4.5 Algorithm. 2020.
[5] J. Fürnkranz. *Decision Tree*, pages 263–267. Springer US, Boston, MA, 2010.
[6] J. R. Quinlan. Induction of decision trees. Machine Learning, 1:81–106, 1986.
[7] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
[8] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.
[9] K. Sotiris. (2007). Supervised Machine Learning: A Review of Classification Techniques. Informatica (Ljubljana). 31.
[10] M. Fakir and H. Ezzikouri. Algorithmes de classification : Id3 & c4.5. page 36.
[11] N. Patil, R. Lathi and V. Chitre. (2012). Comparison of C5. 0 & CART classification algorithms using pruning technique. Int. J. Eng. Res. Technol.. 1. 1-5.
[12] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
[13] P. Preux. Notes de cours de fouille de données. Technical report, Support de cours, Université de Lille 3, 2006.
[14] Y. Lebbah. (2021, Febriary). Support de cours: Fouille de Données Orientée Motifs. Université Oran1. [Online]. Available: https://sites.google.com/site/ylebbah/teach?authuser=0#h.xeqj7nrmwu9q

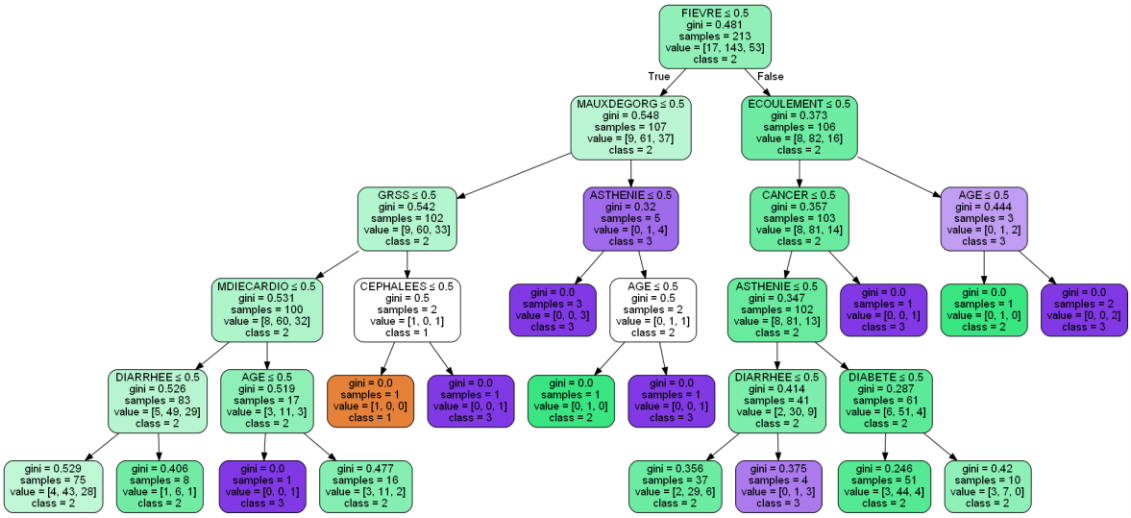


Fig. 5. Optimized decision tree of the COVID-19 classifier

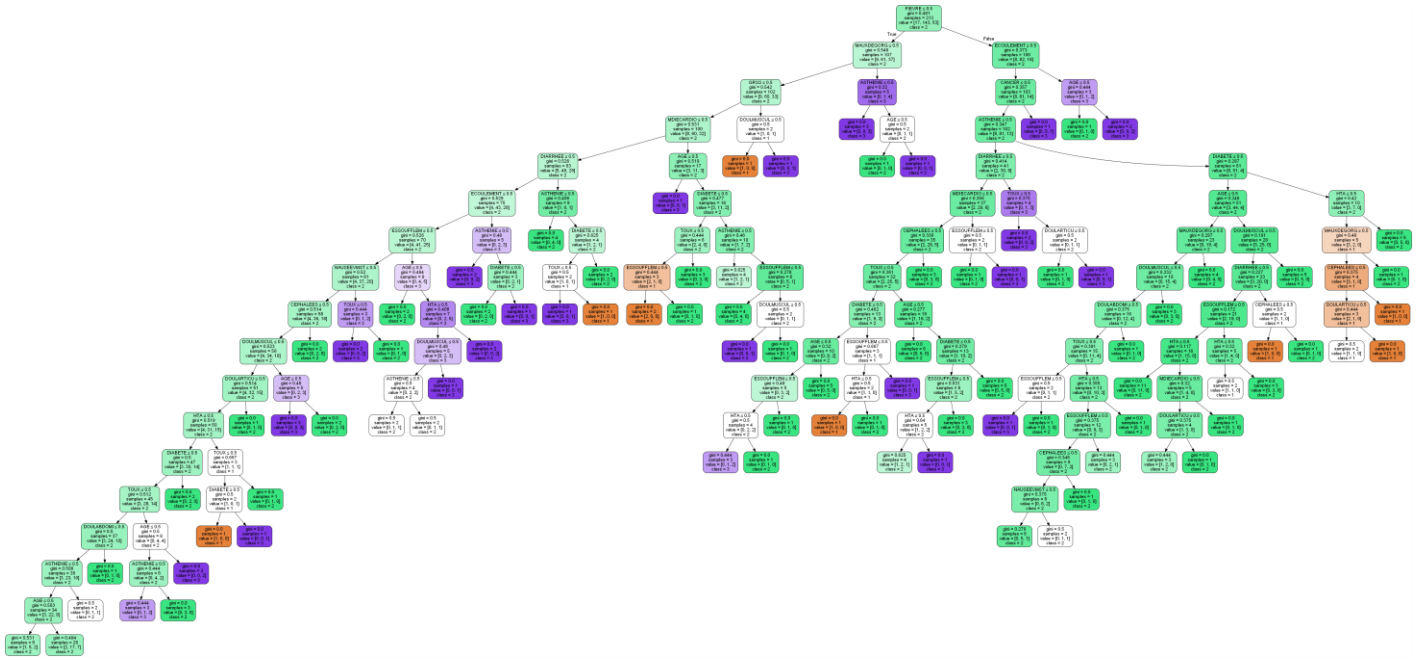


Fig. 6. Classical decision tree of the COVID-19 classifier