Decision Tree Modelling with CART Algorithm to Predict Potential Needs of Intensive Care for COVID-19 Patients Based on Their Preconditions

Maria Khelli - 13520115[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520115@std.stei.itb.ac.id

***Abstract*—COVID-19 pandemic has led to drastic changes in human life. ICU admission became very important decision to make, especially in hospitals where the cases surge and beds are limited. Hence, the latest technologies should be able to support and facilitate the combat in the ongoing situation. Artificial intelligence (AI) is one of the solutions. This paper uses decision tree as the AI agent. The decision tree was implemented using Classification and Regression Tree (CART) algorithm which uses Gini impurity and information gain. The model performance evaluates to 61.8% F1-score with 72.8% recall and 53.6% precision score. The author found that the decision tree was not effective enough in this problem since the tree has a high bias. However, this may be due to the fact that the data are imbalanced. Therefore, making pattern recognition task harder.**

***Keywords*—covid-19, decision tree, intensive care, health preconditions**

I. Introduction

The outbreak of Coronavirus Disease (COVID-19) in December 2019 has put humans in a pandemic again since the Spanish Flu. COVID-19 is an infectious disease caused by the SARS-CoV-2 virus. As of December 2021, more than 267 million cases and about 5 million deaths have been reported [1].

According to World Health Organization (WHO), the majority of people can recover from the disease without any special treatment [2]. However, some groups are more prone to it in which they will potentially develop an illness that may require intensive care [2].

COVID-19 patients with severe symptoms are at high risk for health deterioration. Thus, intensive care unit (ICU) admission should be prioritized for them [3]. However, ICU treatments also depend on the hospital availability in medical workers and beds. Hence, ICU should be optimized and prioritized for those who need them.

While dealing with COVID-19, ICU admission is not an easy task, especially when the cases surge. Crowded patients, shortage of healthcare workers, and chaotic situations make admission much more difficult. That is because the healthcare staff should compare new arriving patients' preconditions data and decide the ICU prioritization in no time.

In this case, artificial intelligence (AI) can help the medical staff determine which patients need to have intensive care immediately. AI is commonly used to diagnose, treat, and predict outcomes in many clinical scenarios [4]. One of the AI technology in decision-making is a decision tree.

A decision tree is a predictive model which can solve both classification and regression tasks [5]. This paper aims to research the applicability of decision trees in ICU admission. The goal is to make a tree that can help medical workers distinguish who should be prioritized. The model should be capable of sorting the patient data from most prioritized to less prioritized.

The writer uses the Classification and Regression Tree (CART) algorithm. It has cheaper computation compared to other methods because it uses the Gini index rather than entropy which contains a logarithm.

Similar research had been conducted by Cotoy, et al. [6]. They use an artificial neural network (ANN) to predict ICU care requirements in COVID-19. The model successfully performs at 78% recall and 80% precision which translates to 79% F1-score.

The contrast to the existing paper is that this study uses classical machine learning (ML) approach rather than deep learning. The purpose of this research is to add more references to the existing models because the classical ML is computationally and financially cheaper than the deep learning approach [7].

II. Methodology

A. Tools

In this study, the writer uses Python programming language with its libraries, namely Pandas, NumPy, Matplotlib, SciKit Learn, Imblearn, and the writer's own decision tree classifier module.

B. Data Descriptions

The data is retrieved from Kaggle Open-Source Datasets, titled "COVID-19 Patient Pre-condition Dataset" [8]. The original data is obtained from Mexican Official Government [9]. Last update of the data is 22nd July 2020.

The dataset contains 566,602 rows and 23 columns. The columns details are:

1. Id: patient's identifier (object),
2. Sex: patient's gender (1: female, 2: male),
3. Patient type: patient's care status (1: outpatient, 2: inpatient),
4. Entry date: date arrival to the hospital (datetime),

5. Date symptoms: date when the COVID-19 symptoms began (datetime),
6. Date died: patient's die date (datetime),
7. Intubed: whether the patient is intubated or not (cyn),
8. Pneumonia: whether the patient has pneumonia or not (cyn),
9. Age: patient's age (numerical),
10. Pregnancy: whether the patient is pregnant or not (cyn),
11. Diabetes: whether the patient has diabetes or not (cyn),
12. COPD: whether the patient has chronic obstructive pulmonary disease or not (cyn),
13. Asthma: whether the patient has asthma or not (cyn),
14. Inmsupr: whether the patient has immunosuppression or not (cyn),
15. Hypertension: whether the patient has hypertension or not (cyn),
16. Other disease: whether the patient has other disease or not (cyn),
17. Cardiovascular: whether the patient has cardiovascular disease or not (cyn),
18. Obesity: whether the patient is obese or not (cyn),
19. Renal chronic: whether the patient has chronic kidney disease or not (cyn),
20. Tobacco: whether the patient does tobacco or not (cyn),
21. Contact other covid: whether the patient had contact with a person that has COVID-19 or not (cyn),
22. Covid res: patient's COVID-19 test result (1: positive, 2: negative, 3: awaiting results),
23. ICU: whether the patient needs ICU admission or not (cyn).

The *cyn* means *categorical yes or no*. The labels provided for that category are 1 for yes, 2 for no, and 97, 98, 99 for missing values. Our model's target will be the "ICU" column.

C. Data Preprocessing

Data preprocessing is done through several steps. The author only takes the necessary data which later contains 67,414 rows and 15 columns. The detailed preprocessing steps are listed and explained below.

1. Convert 97, 98, 99 labelled data as np.nan so Pandas can recognize null data through pd.DataFrame.isnull.
2. Change all "2" values to "0" as it is the more popular convention.
3. ICU preprocessing: since the data is imbalanced (or skewed), the author changes the ICU column value to prioritized "1" if the patient has a valid date died. The valid date died can be interpreted that the patient has died while the invalid date died can be thought as the patient has not died.
4. Create a new column named *days after symptoms* which is created from subtracting *date symptoms* column to *entry date* column.
5. Since we want to predict ICU from COVID-19 patients, we only take patients that has "inpatient" status and has tested positive for COVID-19. After that, drop the patient type and covid res column.
6. Column id, date died, entry date, and date symptoms are dropped because they are insignificant to the target column.
7. Column pregnancy, other disease, and contact other covid are dropped because they have many missing values.
8. The rest missing values is dropped row-wise using pd.Dataframe.dropna.
9. Cast the categorical column to object and numerical column to either integer or float. This step is needed because the tree is modelled to recognize categorical value as object and numerical value as either float or int.
10. Lastly, data are split to 80% for training and 20% for testing/validation data. The training data is resampled to a 1:1 ratio in the target column using Borderline SMOTE technique because the data is skewed. The validation data are left as they are.

D. Decision Tree Implementation

The tree used in this problem is binary decision tree since all categorical columns have only two classes. Before going into the decision tree algorithm, the readers should know these following terminologies.

1. Node: the three different types are root node, internal nodes, and leaf nodes.
2. Branch: represent edge that connects two nodes.

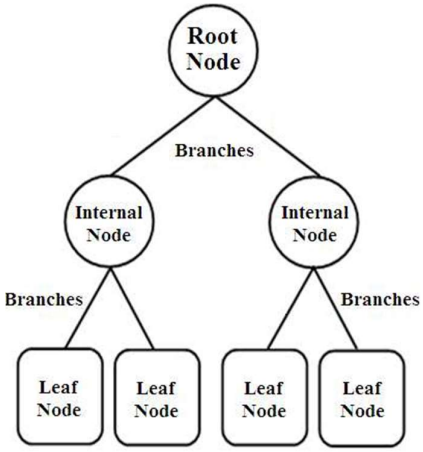

Fig. 2.1 Tree nodes visualization [10]

3. Level: tree location relative to the root node. In this paper, the root node has a level of 1.
4. Depth: maximum level of a tree.

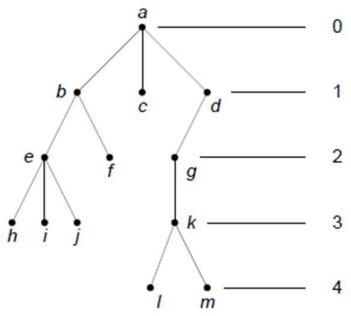

Fig. 2.2 Tree level visualization [11]

5. Parent node: node which has one lower level than certain node, e.g., in figure 2, node *d* is a parent of node *g*.
6. Child node: node which has one higher level than certain node, e.g., in figure 2, node *m* is a child of node *k*.
7. Sibling node: node which has same level as certain node, e.g., in figure 2, node *e* is a sibling of node *f*.
8. Subtree: trees are recursive. It is possible to contain a tree in a tree, this is called subtree. In figure 3, subtree is the circled area.

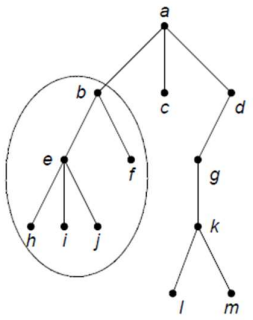

Fig. 2.3 Subtree visualization [11]

9. Pruning: reducing size of a tree by removing a subtree or several subtrees. This technique is usually used to avoid overfitting in training data.

CART Algorithm

CART stands for Classification and Regression Trees. It is one of many algorithms to build the decision tree. The pruning type used in this algorithm is pre-pruning using a single-pass algorithm [11].

The algorithm will start from the root node and will grow recursively until it fulfills the stopping criteria and produce leaf nodes. In this study, the stopping criteria is either when the information gain is less than or equal the tolerance or when the maximum depth is reached. If it is not specified, the default parameter is when the information gain is zero.

In each decision node, the data will be split based on a true or false question. To generate the question, the tree will iterate through all feature columns. Then, in each column, all unique values will be tested as threshold.

For categorical values, the question will compare with the "=" symbol and for numerical values, the question will compare with the "≥" symbol. For instance, "Is age ≥ 32?" (numerical question with threshold 32) and "Is the intubated = 1?" (categorical question with threshold 1).

The question generated will partition the data into true and false groups. The goal of this step is to filter the data until it is less mixed, i.e., maximum data impurity reduction. The impurity in each node is measured with Gini index which is defined in node *t* as

$$i(t) = 1 - \Sigma_i \, p^2(j|t) \qquad (1)$$

where *j* is the class category and $p(j|t)$ is the probability of class j in node *t*.

The tree will also weigh the question by metrics called information gain to see if the question is significant enough in reducing the impurity. Information gain at node *t* is defined as

$$\Delta i(s,t) = i(t) - p_T \cdot i(t_T) - p_F \cdot i(t_F) \qquad (2)$$

where $\Delta i(s,t)$ is the decrease of impurity with split *s*, p_T and p_F are the probabilities of data going to true partition (true child node) or false partition (false child node), $i(t_T)$ and $i(t_F)$ are Gini impurity at node t_T and t_F respectively.

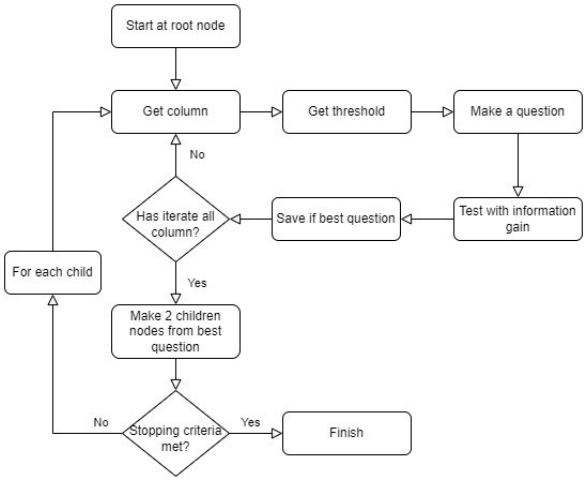

Fig. 2.4 CART algorithm flowchart

In this implementation, the root node and internal node class type is a Decision Node, while leaf node has its own class type, named Leaf. The decision node contains the type of question (categorical or numerical), the question itself, and two children. On the other side, the leaf contains the probability of each classification to appear with certain feature inputs.

To predict the target, the tree will receive several feature inputs and the tree will go through branches based on inputs until it ends in a leaf node. Because the leaf node contains the probability, it can output the probability of a class and it can also determine which class has highest probability. The probability output can be sorted descending by setting *prioritized* "1" class in *key* parameter to help in solving this ICU admission problem.

E. Decision Tree Classifier Module

This section will explain more detailed specifications of the decision tree classifier (DTC) module that the writer created. The detailed descriptions of the class are as follows.

1. DecisionTreeClassifier
   This is the main class of the module. The properties of this class are root (class type: DecisionNode), depth, and col_types to store all feature types. The methods are:
   a. grow_tree: helper function to fit the data inputs and grow the tree. This method works recursively and the splitting data process happens here.
   b. fit: to build a tree based on training data. This method will combine the features and target and transform it to np.array. This method will also extract the column types (categorical/numerical).

c. predict: this method receives feature data. Users can specify the "key" argument if they want the output is in the form of probability (floating point). Otherwise, it will output the class category that has the highest probability.
d. print_tree: this method will output the tree structure that has been fitted.

2. DecisionNode
This class will play a role as an internal node. The properties are question, information gain, true child, and false child. In the output, the true child is a left child and the false child is a right child. The methods are:
a. classify: to predict one row at a time. This method will output the class probability in Python dictionary data structure (or hash table).
b. print_node: helper function to print the tree.

3. Leaf
This class has only one property and no methods. The property is the target classes probability.

4. Question
The properties of this class are variable name, column input, threshold, and isnumeric. The method is
a. match: to pass the data into either true or false branch.

Other functions are not stated to reduce redundancy.

Limitations in The DTC Module

Below are some limitations to the DTC that was created.

1. The DTC can only split the data into two parts (binary). This may cause the model works poorly on multiclassification problem.
2. The DTC only accepts pd.DataFrame type. The user will need to transform the data to pd.DataFrame type before fitting it to the model.
3. The user needs to specify each column type explicitly, i.e., casting to object for categorical columns and casting to integer or float for numerical columns.
4. The DTC feature is not as complete as SciKit Learn's Decision Tree Classifier feature as this module only facilitate maximum tree depth and information gain tolerance as stopping criterion.

F. Evaluation Metrics

To evaluate model performance, the writer uses F1-score since target data are imbalanced or skewed. The F1-score is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{4}$$

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP+FP+FN}} \tag{5}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative. This metric will be tested in validation data after fitting to the decision tree model.

III. Results and Discussions

Before training data, the author tried several combinations of preprocessing techniques to see which combination gave the best result. The preprocessing written in II. B. are the ones that are implemented to the data.

Data preprocessing impacts on training result

The data preprocessing impact can be summarized as follows.

ICU Preprocessing	Resampling	Missing Handler	F1-score
Yes	Yes	Drop	61.8%
Yes	Yes	Impute	61.6%
Yes	No	Drop	49.6%
Yes	No	Impute	49.2%
No	No	Drop	28.9%
No	No	Impute	29.5%
No	Yes	Drop	23.6%
No	Yes	Impute	23.6%

Table 3.1 Data preprocessing impact to F1-score in test data

From table 3.1, it can be inferred that the most significant preprocessing is step three, the ICU preprocessing (refer to II. B.). This is because the preprocessing helps to increase the minority labels in the target column. Although this process manipulated the target data, it is reasonable to say that the people who already died and not admitted to the ICU should have been cared in the ICU.

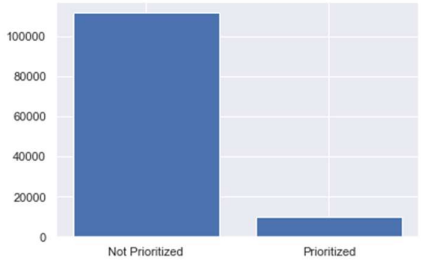

Fig. 3.1 Before ICU preprocessing

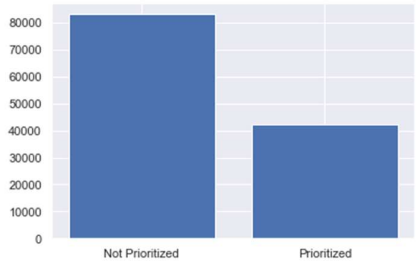

Fig. 3.2 After ICU preprocessing

Figure 3.1 and figure 3.2 show that the target ratio changed from 10:1 to 2:1. This will help the model to recognize the pattern.

The second most significant preprocess is resampling the data with Borderline SMOTE. The data are resampled so the target ratio changed from 2:1 to 1:1. The final ratio is determined automatically from default parameter in Imblearn library.

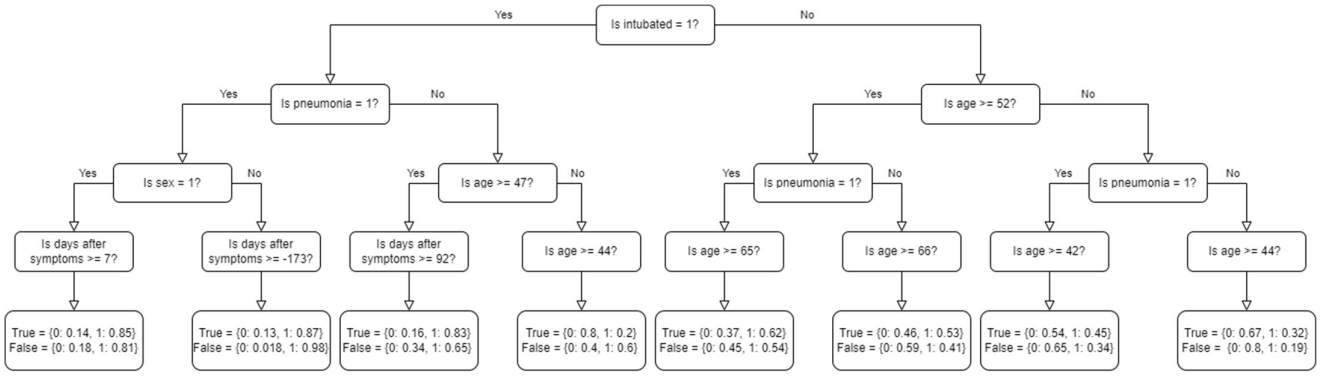

Fig 3.4 Final tree structure

Additionally, in handling missing data, the writer tried to use imputer by filling the missing data with the most frequent value. The most frequent method was chosen because the majority data are categorical and hence, discrete. However, the author found that dropping the missing values gave better result than imputing them. Therefore, after applying the best preprocessing combination, the writer successfully increased F1-score by 38.2%.

Model Evaluation

The first trial of training resulted in overfitting because the tree performed well in training data, but performed poorly in test data.

	Training Data	Validation Data
Precision	89.9%	51.6%
Recall	87.8%	49.3%
F1-Score	88.8%	54.1%

Table 3.2 Evaluation scores with default parameter

To overcome the overfitting problem, the author pruned the tree by testing several tree maximum depths to determine which one is optimum for the validation data. The results are as follows.

Maximum Tree Depth	F1-score
2	60.8%
3	60.4%
4	61.8%
5	59.8%
6	60.7%
7	60.5%
8	60.2%
9	60.2%
10	60.0%
11	59.1%
12	59.2%
13	58.8%
14	58.2%
15	58.0%

Table 3.3 Max tree depth variations and F1-score on test data

From table 3.3, we can see the best validation F1-score is 61.8% with a maximum tree depth of 4. The statistics are:

	Training Data	Validation Data
Precision	62.0%	53.6%
Recall	67.0%	72.8%
F1-Score	64.4%	61.8%

Table 3.4 Evaluation scores with max tree depth = 4

By comparing the statistics from table 3.4, it is safe to say that the tree did not overfit the training data. The reason is that the gap between F1-score training data and F1-score validation data only differs about 2.6%. It is due to the pruning done to the tree that avoids overfitting.

Having only 61.8% F1-score, the decision tree may not be a good solution to help medical staff with ICU admissions. This is because the model will tend to give errors rather than the desired outcome. To see the problem, the author also provides confusion matrix plot as follows.

Fig 3.5 Confusion matrix for validation data

Remember that the prioritized patient is labeled as 1 (positive) while non-prioritized is labeled as 0 (negative). From figure 3.5 plot, we can see that the false positive has a relatively high value (approx. 24.8% of the test data). This fact can also be seen through the low precision score.

With current performance, the decision tree can misclassify labels and prioritize people who actually do not need an ICU admission. This is not what we wanted since it will waste bed occupancies in the hospitals.

On the other side, if we look at the tree structure in figure 3.4, several questions are asked repeatedly. The author presumes that this may be one of the problems that caused the poor tree performance. The tree does not consider all features, whereas in fact, the COVID-19 patient's symptoms severity and health condition are dependent on comorbid, such as hypertension and cardiovascular disease [12]. Hence, affecting the needs of intensive care.

Furthermore, when inspecting the right subtree of node "Is age ≥ 52?", we see that the leaves tend to predict patients as the prioritized ones (higher 1 class probability). It turns out that most of the tree false positive values are coming from these. There are 3,133 rows of not intubated patients and has the age above 52, which should be predicted false, but predicted true (prioritized) by the tree. Therefore, this shows that the decision tree has a high bias to such situation.

Besides that, the decision tree root has very low information gain on the data (approximately 0.02). This means that it is hard for the tree to filter the data because the Gini impurity is high. One possible factor that caused this is target may need to be decided with several questions combined at the same time, not one question at a time like the tree did.

IV. Conclusion

To conclude, the tree has 61.8% F1-score with 72.8% recall and 53.6% precision score. Compared to the existing method from Cotoy, et al., the tree F1-score differs by 17.2% with the ANN method being the highest of the two. It shows that the decision tree may not be the best solution to implement in ICU admission task, although it depends on the performance error tolerance.

Nevertheless, further research can be done with different preprocessing, different data, or different models. In this paper, the majority of columns of the data are categorical. Therefore, more data with numerical values can be collected and tested. Specified data such as patients' blood pressure, cholesterol, or oxygen rate may correlate more to ICU admission requirements. Additionally, the target does not have to be categorical. It can be numerical, but a certain threshold should be determined.

VI. Acknowledgment

The author would like to gratitude for the help provided by Dr. Nur Ulfa Maulidevi, S.T., M. Sc. as Discrete Mathematics Instructor, Dr. Rinaldi Munir, M. T. as Discrete Mathematics Coordinator, and Stanley Yoga as mentor. The author would also like to thank friends and family who support the author throughout the study at Institut Teknologi Bandung.

References

[1] World Health Organization. (2021, December 12). *WHO Coronavirus (COVID-19) Dashboard*. Retrieved from WHO: https://covid19.who.int/. Accessed at 11th December 2021, 17.15 GMT+7.

[2] World Health Organization. (n.d.). *Coronavirus*. Retrieved from WHO: https://www.who.int/health-topics/coronavirus. Accessed at 11th December 2021, 18.05 GMT+7.

[3] Hajjar, et al. (2021). Intensive Care Management of Patients with COVID-19: A Practical Approach. *Annals of Intensive Care*.

[4] Mishra, et al. (2017). Role of Artificial Intelligence in Health Care. *Biochemistry Indian Journal*.

[5] Rokarch, L., & Maimon, O. (2008). *Data Mining with Decision Trees: Theory and Applications.* Singapore: World Scientific Publishing Co.

[6] Cotoy, et al. (2021). Predicting Intensive Care Unit (ICU) Requirement in COVID-19 With Artificial Neural Network. *Easy Chair.*

[7] Seif, G. (2018, April 4). *Deep Learning vs Classical Machine Learning*. Retrieved from Towards Data Science: https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa. Accessed at 11th December 2021, 19.47 GMT+7.

[8] Mukherjee, T. (2020, July 22). *COVID-19 Patient Precondition Dataset*. Retrieved from Kaggle: https://www.kaggle.com/tanmoyx/covid19-patient-precondition-dataset/metadata. Accessed at 11th December 2021, 08.22 GMT+7.

[9] Gobierno de Mexico. (2020). *Datos Abiertos Dirección General de Epidemiología*. Retrieved from https://www.gob.mx/salud/documentos/datos-abiertos-152127. Accessed at 11th December 2021, 08.22 GMT+7.

[10] Alberto, et al. (2011). Lightning Forecast Using Data Mining Techniques On Hourly Evolution Of The Convective Available Potential Energy. *10th Brazilian Congress on Computational Intelligence*, (p. 3). Brazil.

[11] Munir, R. (2020). *Pohon Bagian 2*. Retrieved from https://informatika.stei.itb.ac.id/~rinaldi.munir/. Accessed at 12th December 2021, 14.33 GMT+7.

[12] Song, Y.-Y., & Lu, Y. (2015). Decision Tree Methods: Applications for Classification. *Shanghai Archives of Psychiatry*, 130-135.

[13] Sanyaolu, et al. (2020). Comorbidity and Its Impact on Patients with COVID-19. *SN Comprehensive Clinical Medicine*, 1-8.

[14] Gordon, J. (2017, September 14). *Let's Write a Decision Tree Classifier from Scratch – Machine Learning Recipes #8*.

[15] Breiman, et al. (1998). *Classification and Regression Trees.* Florida: CRC Press.

Declaration

I hereby declare that this research paper is my own writing, not an adaptation, translation, or plagiarism.

Palembang, 14th December 2021

Maria Khelli
13520115