CSEC 620 | Assignment 4 report

Team 4 - Nick Mangerian, Jacob Ruud, Hugo Tessier Due on 11-14-2021

This study has been done using resources (dataset and initial code) from an article about the classification of IoT devices in smart environments (Reference 1).

1. Hyperparameter tuning

a. *Explain your hyperparameter tuning process.*

The samples used for the computation of the performance metrics during the hyperparameter tuning aren't in the train set. Thus, as the performance metrics are calculated on separated test samples they aren't biased by training samples. Let's focus on the hyperparameters for the different models implemented:

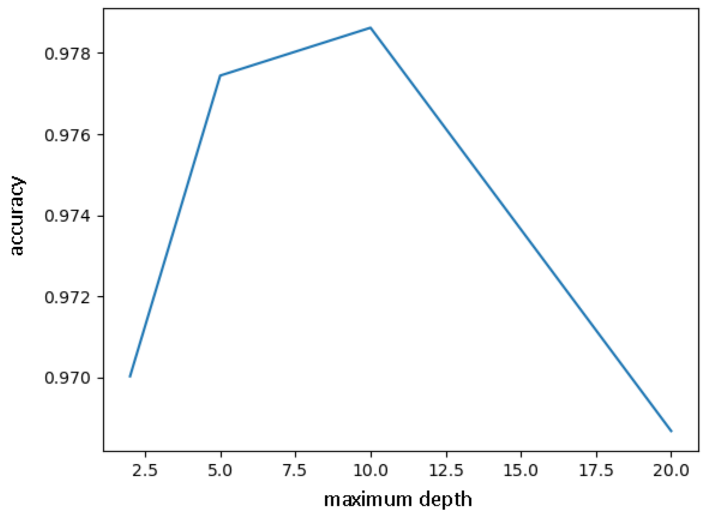
Decision Tree hyperparameters
- **max_depth:** corresponds to the maximum number of levels of a tree
- **min_node:** corresponds to the minimum number of samples in a node in a tree

Random Forest hyperparameters
- **n_trees:** this parameter adjusts the number of trees in the forest
- **data_frac:** this parameter is linked with the number of bootstrapped samples to use during the creation of each tree.
- **features_subcount:** this parameter corresponds to the number of features to consider when splitting a node.

The hyperparameter tuning process consisted in gathering and visualizing a performance metric for a range of each of these hyperparameters. We choose the best value of the parameter and reiterate the process until for the range of all the other hyperparameters.
As we're considering a multiclass classification, we computed the accuracy as a performance metric. False positives and false negatives aren't as problematic as in malware classification in this case of study so accuracy is an appropriate metric.

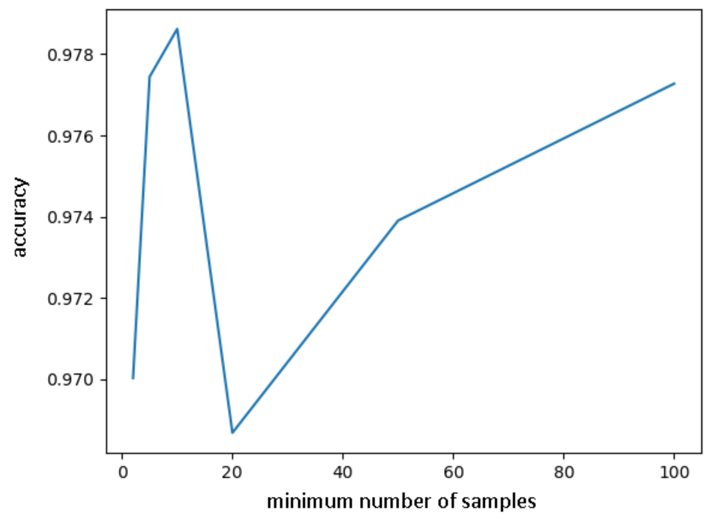
b. *Identify which parameters had the largest impact on performance for your Random Forest model. Use plots to visualize your results, but avoid using too many and over-explaining.*

Maximum tree depth study



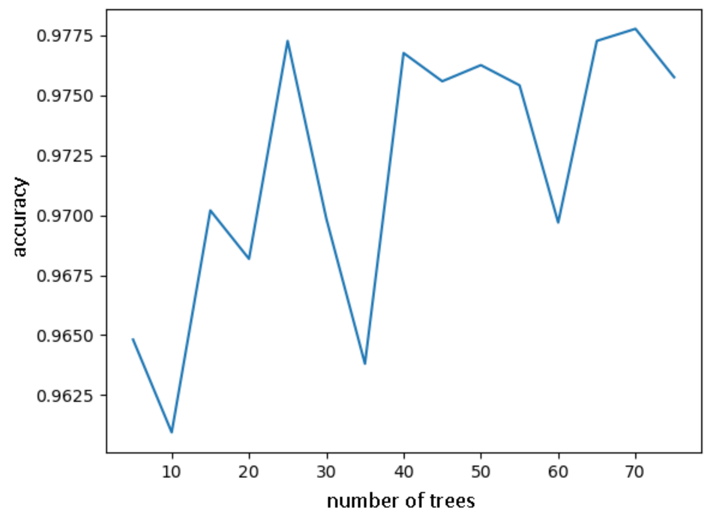
The more depth a tree has, the more it will tend to overfit as it can isolate small sets of samples. However, it takes more execution time to build the tree. As we want to have trees that overfit as much as possible we allow a high depth. We choose max_depth = 10.

Minimum number of samples per nodes study



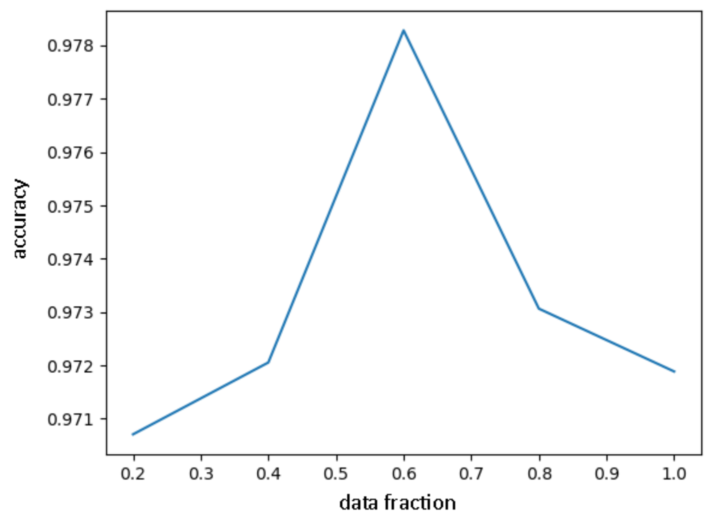
The fewer number of nodes a tree allows, the more it will create few element nodes. Thus, choosing a small number of nodes will help to overfit the trees. We choose min_node = 10.

Number of trees study



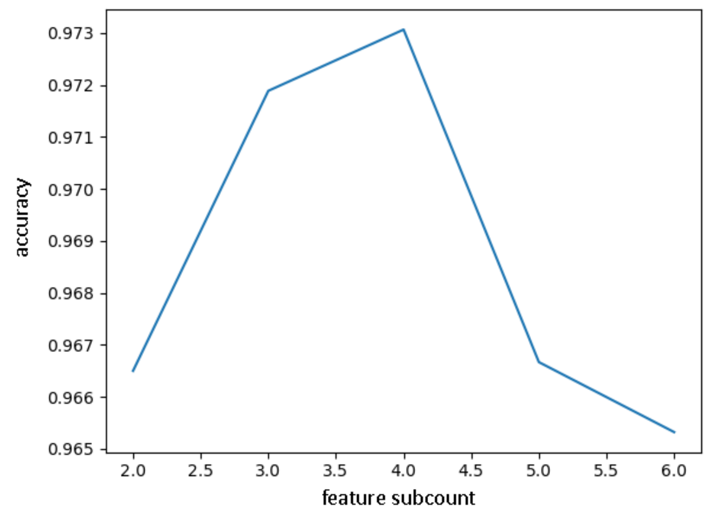
The more trees there are in the forest the more aggregation is made, if the trees aren't correlated the performance will still increase. However, it won't increase a lot after a certain amount of trees and the more trees, the more computation time. We choose n_trees=40.

Data fraction study



As we want to have trees correlated with others as little as possible, choosing different samples to build each tree is a good practice. Thus, we want this hyperparameter to be relatively low. Moreover, it would make the creation of the trees faster. We choose data_frac = 0.6.

Features_subcount study



Choosing different features at each node split helps to decrease the correlation between the trees of the forest. Thus, this hyperparameter is chosen small. As there are only 6 features in the data, we choose feature_subcount = 4.

Conclusion

The hyperparameters that increase the accuracy the most are the number of trees, the data fraction, the maximum depth and the minimum number of samples.

2. **Performance**
    a. ***a. Evaluate the performance of your Decision Tree model (non-ensemble) to your Random Forest model against the data.***
        i. ***How do their execution time and performance compare?***

Decision Tree
max_depth=10, min_node=5

Training time: 7.75695013999939
Accuracy: 0.9561369757599076

Random Forest
n_trees=40, data_frac=0.6, feature_subcount=4, max_depth=10, min_node=10

Training time: 69.34619879722595s
Accuracy: 0.9850523873624759

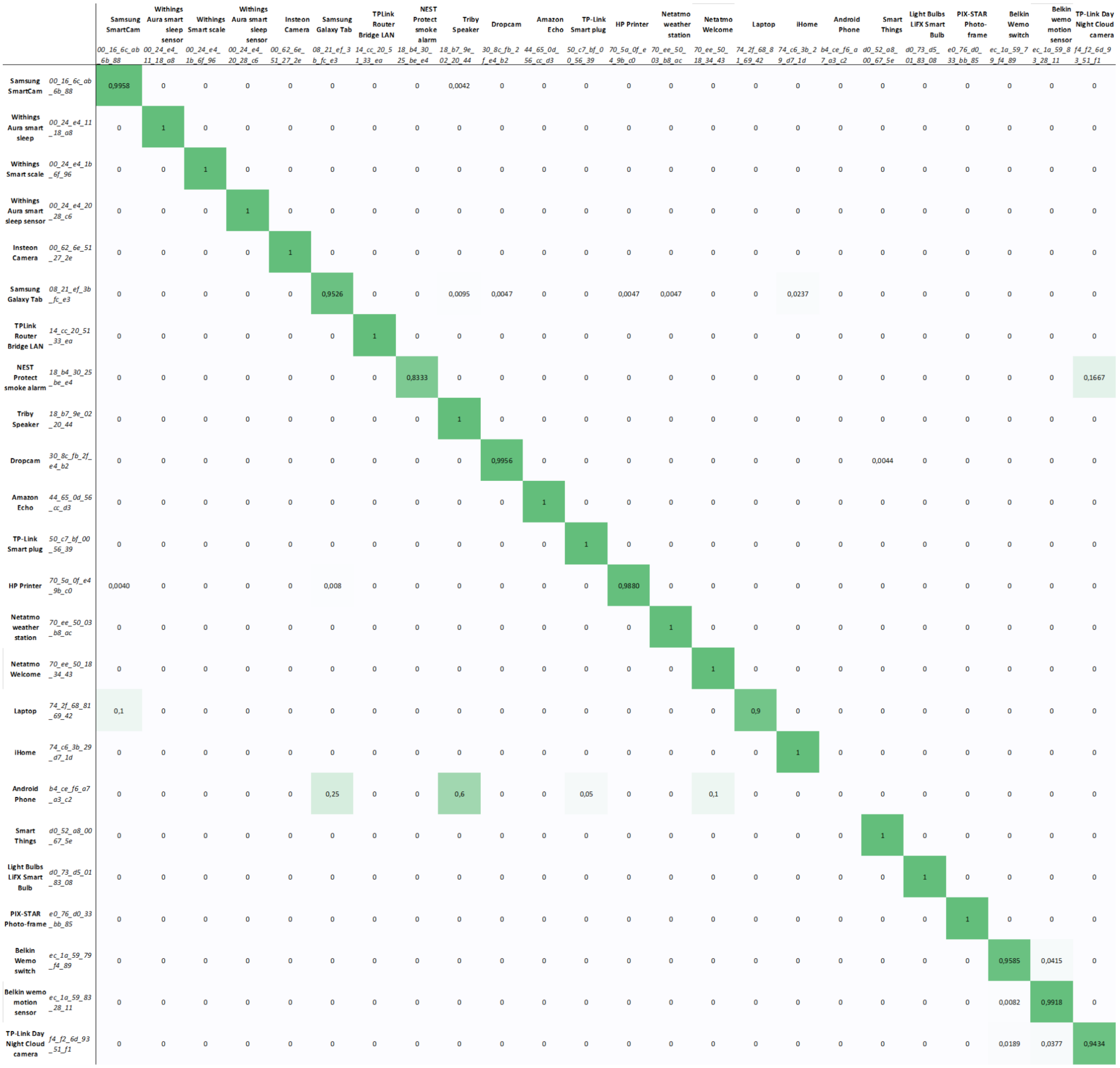
ii. ***Explain why you think you get these results.***

The Random Forest is generating a lot of trees, even if it considers a subset of the samples and only a limited number of variables at each node split, the training time is understandingly higher for this model.

However, Decision Trees tend to overfit. As the samples used to compute the performance are different from the ones used to train the tree, the accuracy score is affected by the overfitting of the tree. With appropriate hyperparameters, Random Forest tends to alleviate this overfitting issue. Then, it is understandable that Random Forest has better performance.

b. *Include a normalized confusion matrix of your results to demonstrate the accuracy of your model on a class-by-class basis.*

In this normalized confusion matrix we associated every MAC address to the corresponding device (a higher quality version is available as attachment to the script files).



		Samsung SmartCam 00_16_6c_ab_6b_88	Withings Aura smart sleep sensor 00_24_e4_11_18_a8	Withings Smart scale 00_24_e4_1b_6f_96	Withings Aura smart sleep sensor 00_24_e4_20_28_c6	Insteon Camera 00_62_6e_51_27_2e	Samsung Galaxy Tab 08_21_ef_3b_fc_e3	TPLink Router Bridge LAN 14_cc_20_51_33_ea	NEST Protect smoke alarm 18_b4_30_25_be_e4	Triby Speaker 18_b7_9e_02_20_44	Dropcam 30_8c_fb_2f_e4_b2	Amazon Echo 44_65_0d_56_cc_d3	TP-Link Smart plug 50_c7_bf_00_56_39	HP Printer 70_5a_0f_e4_9b_c0	Netatmo weather station 70_ee_50_03_b8_ac	Netatmo Welcome 70_ee_50_18_34_43	Laptop 74_2f_68_81_69_42	iHome 74_c6_3b_29_d7_1d	Android Phone b4_ce_f6_a7_a3_c2	Smart Things d0_52_a8_00_67_5e	Light Bulbs LiFX Smart Bulb d0_73_d5_01_83_08	PIX-STAR Photo-frame e0_76_d0_33_bb_85	Belkin Wemo switch ec_1a_59_79_f4_89	Belkin wemo motion sensor ec_1a_59_83_28_11	TP-Link Day Night Cloud camera f4_f2_6d_93_51_f1
Samsung SmartCam	00_16_6c_ab_6b_88	0,9958	0	0	0	0	0	0	0	0,0042	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Withings Aura smart sleep	00_24_e4_11_18_a8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Withings Smart scale	00_24_e4_1b_6f_96	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Withings Aura smart sleep sensor	00_24_e4_20_28_c6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Insteon Camera	00_62_6e_51_27_2e	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Samsung Galaxy Tab	08_21_ef_3b_fc_e3	0	0	0	0	0	0,9526	0	0	0,0095	0,0047	0	0	0,0047	0,0047	0	0	0,0237	0	0	0	0	0	0	0
TPLink Router Bridge LAN	14_cc_20_51_33_ea	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NEST Protect smoke alarm	18_b4_30_25_be_e4	0	0	0	0	0	0	0	0,8333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,1667
Triby Speaker	18_b7_9e_02_20_44	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dropcam	30_8c_fb_2f_e4_b2	0	0	0	0	0	0	0	0	0	0,9956	0	0	0	0	0	0	0	0	0,0044	0	0	0	0	0
Amazon Echo	44_65_0d_56_cc_d3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
TP-Link Smart plug	50_c7_bf_00_56_39	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
HP Printer	70_5a_0f_e4_9b_c0	0,0040	0	0	0	0	0,008	0	0	0	0	0	0	0,9880	0	0	0	0	0	0	0	0	0	0	0
Netatmo weather station	70_ee_50_03_b8_ac	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Netatmo Welcome	70_ee_50_18_34_43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Laptop	74_2f_68_81_69_42	0,1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,9	0	0	0	0	0	0	0	0
iHome	74_c6_3b_29_d7_1d	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Android Phone	b4_ce_f6_a7_a3_c2	0	0	0	0	0	0,25	0	0	0,6	0	0	0,05	0	0	0,1	0	0	0	0	0	0	0	0	0
Smart Things	d0_52_a8_00_67_5e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Light Bulbs LiFX Smart Bulb	d0_73_d5_01_83_08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
PIX-STAR Photo-frame	e0_76_d0_33_bb_85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Belkin Wemo switch	ec_1a_59_79_f4_89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,9585	0,0415	0
Belkin wemo motion sensor	ec_1a_59_83_28_11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,0082	0,9918	0
TP-Link Day Night Cloud camera	f4_f2_6d_93_51_f1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,0189	0,0377	0,9434

c. Discuss the results in your confusion matrix.

The accuracy of the model was close to one, it is then understandable that the confusion matrix is close to the identity matrix (with 1 on the diagonal). We can notice that some devices are confused with others, we'll study it in the next question.

d. Are there any classes that are commonly confused with one another? Why or why not may that be the case? (Hint: Use the list_of_devices.txt to translate MAC address labels into device types to help explain the classifier confusion).

As we can see in the confusion matrix, the MAC addresses are linked with the device type. Most of the devices are almost perfectly classified and the True Positives are dominating. It is less the case for the Laptop category that is sometimes confused with the Samsung SmartCam. This can be due to similar features. Moreover, the Android Phone is not well predicted, there must be few samples of this category as its bad accuracy doesn't shift the accuracy much. It is most of the time confused with Triby Speaker. As one can notice these two last objects are not real IoT devices and thus may be harder for our model to classify them. It is also likely that the devices misclassified as the android phone are running some sort of android OS under the hood as android is open source and easily customizable for IOT applications.

3. How useful would it be to normalize the data? Explain your answer.

Normalizing the data isn't useful for Decision Trees or Random Forests. It is due to the split of each node which is done on only one variable. These machines structurally don't directly compare a feature with another by computing distances.

4. Select one non-trivial node (i.e. not a leaf) in one decision tree in your final forest. Print out all the information used at that node to compute Gini impurity. Show how to use this information to compute the node importance for that node, and do the computation.

To compute the node importance, we first extracted the labels of the samples of a node of one of the trees of the generated forest and the labels of its child nodes (left and right).

Labels of each node

Parent node labels:

```
[21 22 22 22 21 21 22 21 22 21 22 22 22 22 22 22 21 21 22 22 22 22 22 22 21 21 21 22 22 21 22
21 22 21 22 22 22 21 22 22 22 21 21 22 22 21 21 21 22 21 22 22 22 22 21 22 22 22 21 21 22 22
22 22 22 22 22 22 21 21 22 22 22 22 21 21 21 21 22 22 22 21 21]
```

Left node labels:

```
[22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 21 22
22 22 22 22 22 22 22 21 22 22 22 22 22 22 22 22 22 22 22 22 22 22]
```

Right node labels:

```
[21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21  21 21 21 21 21]
```

We then compute the node importance based on the Gini impurity computation.

Gini Impurity computation (Gini impurity formula: Σ p(i) * (1-p(i)))
Gini impurity for parent node = Σ p(i) * (1-p(i))

= (p(21) * (1 - p(21))) + (p(22) * (1 - p(22)))

$= (\frac{31}{83}*(1-\frac{31}{83})) + (\frac{52}{83}*(1-\frac{52}{83}))$

$= (\frac{31}{83}*\frac{52}{83}) + (\frac{52}{83}*\frac{31}{83})$

= 0.46799245173

Gini impurity for parent node = **0.46799245173**

Gini impurity for left child = Σ p(i) * (1-p(i))

= (p(21) * (1 - p(21))) + (p(22) * (1 - p(22)))

$= (\frac{2}{54}*(1-\frac{2}{54})) + (\frac{52}{54}*(1-\frac{52}{54}))$

$= (\frac{2}{54}*\frac{52}{54}) + (\frac{52}{54}*\frac{2}{54})$

= 0.07133058984

Gini impurity for left child = **0.07133058984**

Gini impurity for left child =Σp(i) * (1-p(i))

= (p(21) * (1 - p(21))) + (p(22) * (1 - p(22)))

= (1*(1-1)) + (0*(1-0))

= (1*0) + (0*(-1))

= 0 + 0

Gini impurity for right child = **0**

Children weighted impurity =

$$= \frac{(number\ of\ samples\ of\ left\ child) *(\ Gini\ impurity\ of\ left\ child) + (number\ of\ samples\ of\ right\ child) *(\ Gini\ impurity\ of\ right\ child)}{total\ number\ of\ samples\ for\ both\ children}$$

$= \frac{(54* 0.07133058984) + (29 * 0)}{83}$

$= \frac{3.85185185136}{83}$

= 0.04640785363

Children weighted impurity = **0.04640785363**

Node importance computation
Node Importance = Parent node impurity - Children weighted impurity

= 0.46799245173 - 0.04640785363

= **0.4215845981**

This node has a relatively large node importance, and therefore it can be said that it is an important node for the algorithm as it led to a huge decrease of impurity.

References

1. A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," in IEEE Transactions on Mobile Computing, vol. 18, no. 8, pp. 1745-1759, 1 Aug. 2019, doi: 10.1109/TMC.2018.2866249.