CSI5155: Report of Assignment 3

Adrien Heymans

# Question 1: Display/visualised the resultant model created by the decision tree.

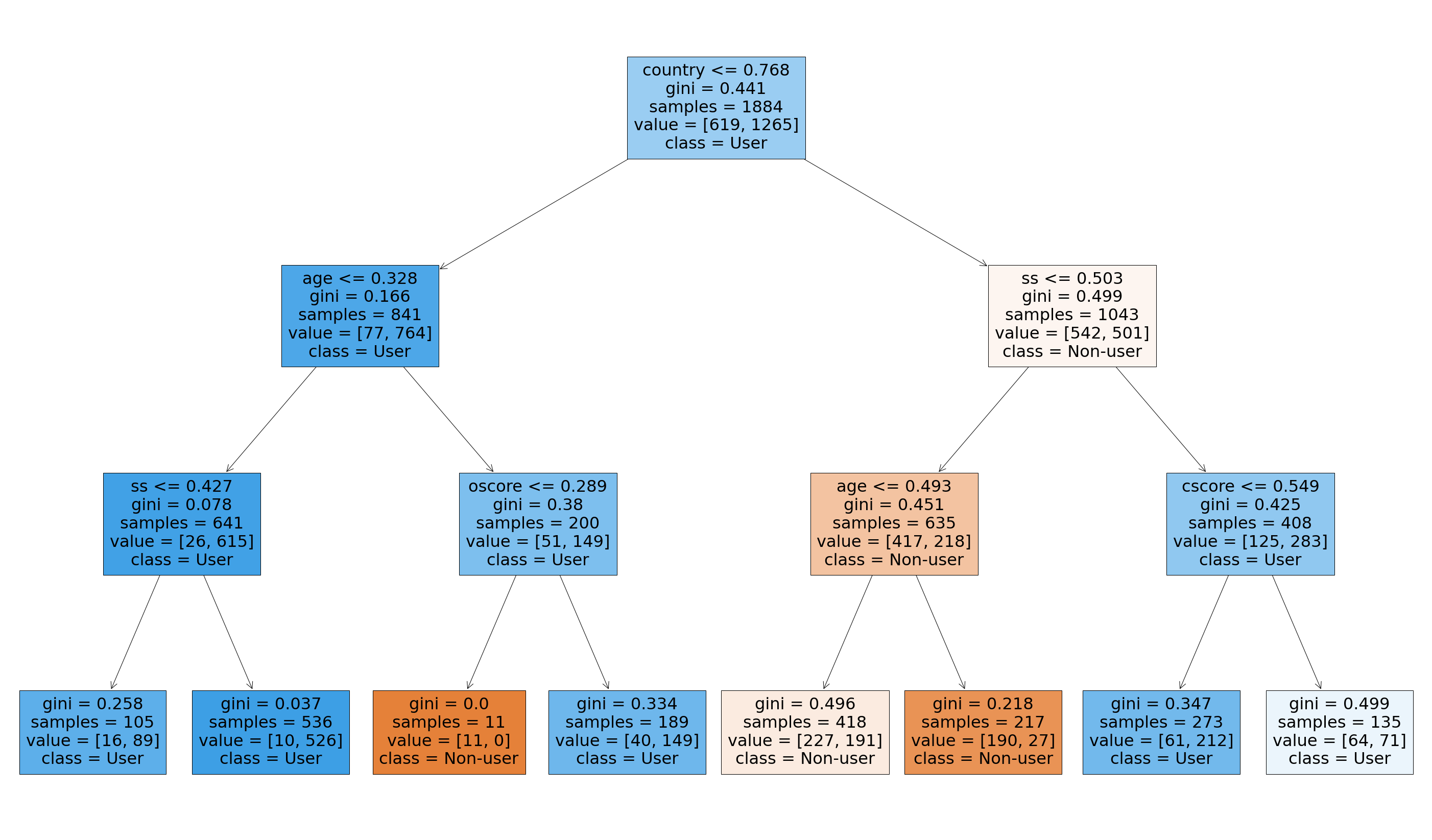


Figure 1 Visual representation of the decision tree

Estimator: Decision Tree Classifier

Balanced accuracy: 0.7840312957334235

Parameters: {'criterion': 'gini', 'max\_depth': 4, 'splitter': ‘best}

# Question 2: Explain how, and why, the algorithm made a specific decision.

The first thing that we can explain is why the algorithm decided to split on certain features and not other at the nodes. We can answer this question by analysing the Gini index at each split.

The algorithm used the Gini index to determine where the best split on the features should be at each level, depending on the data. The Gini index quantifies the purity of the node, similarly to entropy, it is the expected error if we labelled the elements of the leaf randomly. If the Gini score is greater than zero it means that the samples contained within that node belong to different classes. A Gini index close to 0.5, indicates an equal distribution of elements over the classes, we want to avoid this. If we look at the Figure 1, we can see that one of the leaves has a Gini index equals to 0, this means that the tree perfectly classified all the instances in that node.

Let’s explain a specific decision that the algorithm took. As shown on Figure 1, we see that the first split was done on “Country”. One of the parameters of the model is the “criterion”, in our case, it is set to “Gini”, as explained in Question 2. When the algorithm started, it calculated the Gini index of all the different features, compared them and decided to split on the feature with the smallest Gini index. We can see that the split is done on “Country<0.768”, the country feature was normalized, which explains why it is a number between 0 and 1. We also need to answer the following question, why 0.768? We are working here with a numerical feature, when calculating the Gini index, we had to find the midpoints in data and calculate the Gini index for each of them. The split is then done on the midpoints with the lowest Gini index [1].

The different split will lead us to find something closely related to the Gini Index, it is the “feature importance”. By using some python libraries [2], we can visualize the feature importance (see Figure 2). The “Country”, “Age”, “SS” and “Oscore” feature have the most importance. It can be calculated by the decrease in node impurity weighted by the probability of reach that node. The Figure 2 has feature importance of zero for some of the feature, because they have not been used for any splits.

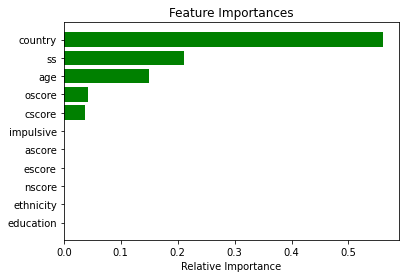


Figure 2 Graph of the relative importance of the features

By understanding these two concepts, we can understand why the tree made some decisions for the classification of the elements.

# Question 3: Explain why the algorithm didn’t do something else.

To answer this, we talked in Question 2 about the split done at each node, and we explained why the algorithm took those decisions, more specifically, the split done at the root of the tree (see Figure 3). Let’s try to understand why the algorithm did not split on a feature other that country.

Text

Description automatically generated

Figure 3 Root of the decision tree

If the algorithm had done the split on another feature, the Gini index would have been higher. This means that there would be more impurity in the instances of the node, after the split, this would lead to little to no progress in our classification task. This explains why the algorithm did not chose to split on another feature for the root.

# Question 4: Discuss when the algorithm succeeded and when it failed.

The algorithm successfully classified almost all the elements in two of the leaves (see Figure 3). They are the purest leaf of the tree with one of them correctly classifying 100% of the instances. Besides these perfect leaves, most of the other leaf have Gini indexs around 0.3, meaning that the algorithm did decently but those are not optimal results.

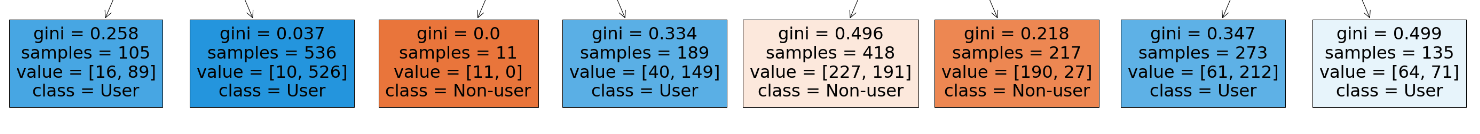


Figure 4 Illustration of the purest leaves from the tree

The algorithm failed to classify the instances in 2 of the leaves (see Figure 4). They have Gini indexes close to 0.5, meaning that there is an equal distribution of elements over the classes.

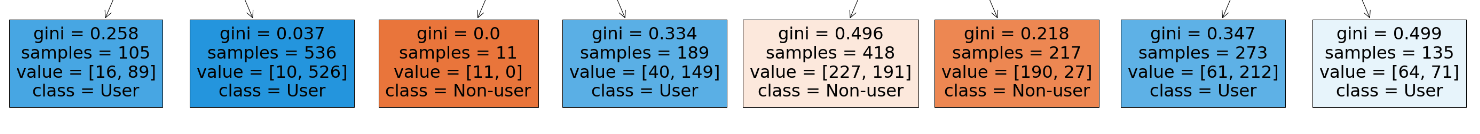


Figure 5 Illustration of the least pure leaves in the tree

# Question 5: Explain how you would decide if the resultant model can be trusted.

We could decide whether the resultant model could be trusted by looking at the precision, recall, the f1-score, the accuracy, or the confusion matrix on the predicted values.

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion matrix** | | **Predicted** | |
| Non-user | User |
| **Actual** | Non-user | 102 | 79 |
| User | 38 | 347 |

Table 1 Confusion matrix of the decision tree

|  |  |
| --- | --- |
| **Measure** | **Score** |
| Accuracy | 0.793286 |
| Precision | 0.814554 |
| Recall | 0.901299 |
| F1 score | 0.855734 |

Table 2 Scores of the decision tree on the dataset

We can decide whether to trust the model by looking at the scores in Table 1 and Table 2, but it is also important to keep in mind the context. For example, let’s say that the purpose of the model is to detect breast cancer, in that case we should be looking how good the model is performing on the positive class, the false negatives should be very low as we do not want to misclassify someone who has breast cancer. In our case, we are looking at detecting whether someone is using Cannabis or not. We do not know what the data will be used for later, it is better not to make any assumption on which class is more important than the other.

If we were to look at the positive class, the recall (score measuring how well the model is performing to predict the positive class) we have a score of 90%, which is a good score, and we will therefore trust the model.

If we are now interested specifically in the negative class, we see from Table 1 that out of the 181 instances, the model correctly classified only 102 of them, giving us an accuracy score of only 56% for the negative class and we would not trust the model.

Finally, if we look at the overall accuracy of the model, we have a score of 79%, we will not trust the model as many other algorithms (Gradient Boosting, Multi-layer perceptron, etc.) have performed much better on this data.

# Question 6: Explain how the algorithm could potentially improve its predictions

One of the ways we could improve the prediction of the algorithm is to use hyper-tuning and cross-validation. In hyper-tuning, our goal is to find the right combination of hyper-parameter for the model. This technique could help us achieve higher results. The downside to this method is that the training time can take longer and consumes a lot of resources (depending on the search space).

A second way we could investigate to improve our results is to use cross-validation on the dataset. By dividing the later into multiple subsets and test the model on each subset one by one. This method can reduce bias as we are using most of the data for fitting, and significantly reduces variance as most of the data is also being used in validation set.

# Bibliography

|  |  |
| --- | --- |
| [1] | T. Ujhelyi, "Coding a Decision Tree in Python Using Scikit-learn, Part #2: Classification Trees and Gini Impurity," 16 02 2022. [Online]. Available: https://data36.com/coding-a-decision-tree-in-python-classification-tree-gini-impurity/. [Accessed 02 11 2022]. |
| [2] | J. Brownlee, "How to Calculate Feature Importance With Python," 30 March 2020. [Online]. Available: https://machinelearningmastery.com/calculate-feature-importance-with-python/. [Accessed 2 Novembre 2022]. |