Edwin Thomas (Student Number: 300278402)

CSI 5155: Machine Learning

Assignment 3: Explainable AI (XAI), interpretability, and trustworthiness

Part 1: Display the resultant model created by the decision tree

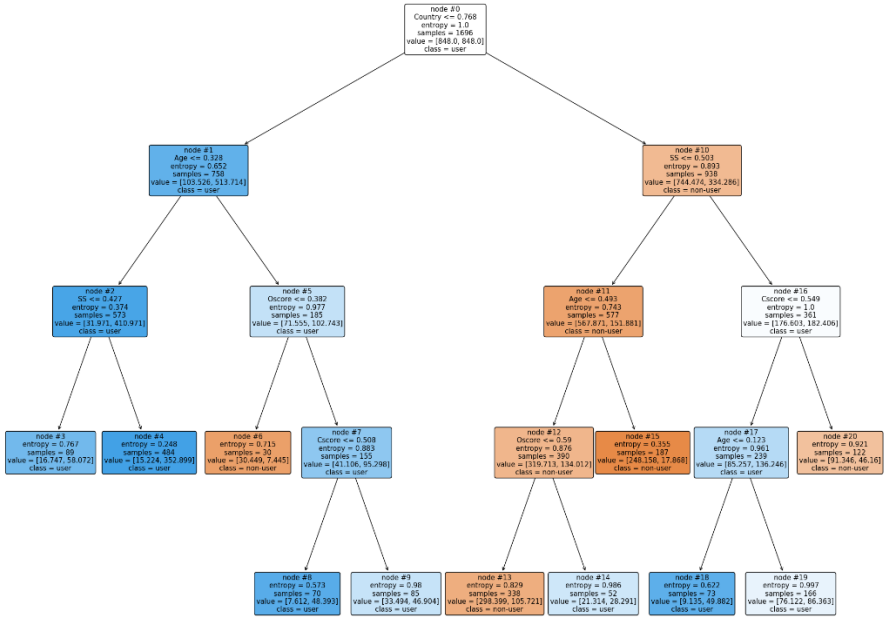


Fig 1. A visualization of the Decision Tree Classifier trained on the Drug Consumption Dataset (Cannabis).

Edwin Thomas (Student Number: 300278402)

The Decision Tree shown in Fig. 1 is trained on fold-8 (fold with the best performance) of the Cannabis Drug Consumption dataset in a stratified nested k-fold manner. The best hyperparameters obtained for the fold are as shown below (Table 1):

Parameter Name	**Parameter Value**
Ccp_alpha	0.005
Class_weight	Balanced
Criterion	Entropy
Max_depth	4
Max_features	None (use all features)
Min_samples_leaf	27
Splitter	best

Table 1. Parameter values for best DT obtained in training fold 8.

The best generalization on the fold was observed for a depth of 4 with a pruning factor of 0.005. The criterion of the decision split is Entropy and the samples were weighted by their class proportions in each split as indicated in Table 1.

The tree in Fig 1. is divided into 3 components: The root node, internal nodes, and the leaf nodes.

In each node, the distribution of samples for each class [user, non-user] is specified under the **values attribute**. It is to be noted that the specified values are floating points, as they represent weighted class samples (Class_weight: balanced in Table 1). The weight is computed using the below formulation:

```
n_samples/(n_classes*np.bin count(y))  → (1)
```

The **entropy** of a given node is given by (2), where pi is the frequentist probability of class i. Entropy measures the amount of randomness in the system. A higher entropy indicates that the distribution of samples belonging to the user and non-user classes are almost the same, and a lower score indicates how close the node is to be homogenous (having samples from only one class).

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \qquad \rightarrow (2)$$

The **class** attribute indicates the majority class within the given node which can be either user or non-user and **samples** indicate the total number of user and non-user samples within the node.

Edwin Thomas (Student Number: 300278402)

Constructing the Tree

Initially, the root node with all the samples has an impurity of 1.0 as there is an equal number of samples from both classes (balanced).

The entropy scores = -((848/1696)*log2(848/1696)) -((848/1696)*log2(848/1696)) = -0.5(-1)-0.5(-1) = 1 (Eq. 2)

To decide the feature for splitting the root node, all the entropies of all possible split points of all the features are computed and the feature that results in the least entropy of the resultant left and right splits are selected.

For a given feature the best achievable entropy is computed using the below methodology:

1. Compute unique values of the feature
2. Find all mid-points between these values (they are the potential split point candidates)
3. For each mid-point split the data based on each of these mid-points, such that values **lesser than or equal to the mid-point** go to the **left sub-tree** and values **greater than this mid-point** go to the **right sub-tree**.
4. Compute the entropy of the left and right split using Eq 1.
5. Then compute the combined entropy of the resultant split by (ratio of left samples)*Entropy_left + (ratio or right split samples)*Entropy_right.
6. Choose the split value as the one that resulted in the least combined entropy value.

Table 2. shows the computations for the Country Feature on all possible split points. As noted in Fig. 1, the split point chosen is 0.76 as it resulted in the least entropy value of 0.74.

Mid-point	Entropy (left)	Entropy (right)	Combined Entropy
0.033209006	0.287677647	0.99083579	0.784780603
0.126258239	0.285550153	0.991431913	0.782497571
0.247349616	0.361626587	0.998081095	0.769167622
0.409498272	0.371634507	0.999758762	0.756434685
0.522790367	0.386321948	0.999943949	0.755725287
0.767592478	0.4355003	0.998687829	**0.746980195***

Table 2. Candidate split points for the Country feature at the root node and their entropies.

Repeating the other features yields results in Table 3-7.

Mid-point	Entropy (left)	Entropy (right)	Combined Entropy
0.033209006	0	0.912483412	0.91140737
0.126258239	0.918295834	0.911430793	0.911467223
0.247349616	0.918295834	0.897821462	0.898654441
0.409498272	0.987102598	0.830649674	0.864504758
0.522790367	0.999005194	0.717534911	**0.847150649***
0.767592478	0.928762832	0.371232327	0.896547036

Table 3. Candidate split points for the Oscore feature at the root node and their entropies.

Mid-point	Entropy (left)	Entropy (right)	Combined Entropy
0.033209	0.852405	0.908886	0.908286
0.126258	0.852405	0.908886	0.908286
0.24735	0.989844	0.861793	0.882178
0.409498	0.999754	0.814694	**0.873835***
0.52279	0.974863	0.690894	0.874402
0.767592	0.924639	0.606111	0.90567

Table 4. Candidate split points for Impulsive features at the root node and their entropies.

Mid-point	Entropy (left)	Entropy (right)	Combined Entropy
0.033209	0.974489	0.902209	0.904937
0.126258	0.974489	0.902209	0.904937
0.24735	0.955798	0.845891	0.863258
0.409498	0.98513	0.70737	0.80678
0.52279	0.999308	0.630962	**0.80623***
0.767592	0.954304	0.428378	0.869337

Table 5. Candidate split points for SS feature at the root node and their entropies.

Edwin Thomas (Student Number: 300278402)

Mid-point	Entropy (left)	Entropy (right)	Combined Entropy
0.033209	0	0.913504	0.912965
0.126258	0.863121	0.913486	0.913278
0.24735	0.494183	0.923348	0.904623
0.409498	0.634105	0.9666	0.874655
0.52279	0.770629	0.995957	**0.86828***
0.767592	0.909126	0.995378	0.911669

Table 6. Candidate split points for CScore feature at the root node and their entropies.

Mid-point	Entropy (left)	Entropy (right)	Combined Entropy
0.033209	0.476097	0.991267	0.816
0.126258	0.476097	0.991267	0.816
0.24735	0.676163	0.992901	**0.804277***
0.409498	0.80559	0.949452	0.836211
0.52279	0.80559	0.949452	0.836211
0.767592	0.892174	0.940286	0.894954

Table 7. Candidate split points for the Age feature at the root node and their entropies.

For each feature, the best obtained splits points and their corresponding entropy scores are highlighted in the Combined Entropy column (Tables 3-7). As we can see Country is the best feature to split on as it obtains the lowest entropy score of 0.7469 (which is the least among all features).

After forming the left and right splits, the same algorithm is applied to the individual sub-trees on all features until the resultant splits are homogenous (containing samples from only one class) or meet the maximum depth or minimum samples per leaf criterion. This explains the generation of the tree in Fig. 1. Fig 2. shows an alternate representation where the relative class distributions for the selected features in each internal node and the distributions in the leaf nodes are seen.

PS: Another perspective to look at the problem is to choose features based on maximum information gain, which is the difference between the entropy of the parent and the combined entropy due to the split. As the entropy of the parent node is constant, we can simply look at the minimum combined entropy of the split (which is the same as maximizing its difference with the parent node's entropy).

Edwin Thomas (Student Number: 300278402)

Part 2: Explain how and why the algorithm made a specific decision.

The decision tree classifier algorithm works by following the divide-and-conquer approach on dataset features to classify data. It essentially partitions the instance space into segments where each segment corresponds to the left and right sub-trees of the node under consideration. This implies that after every split the partitioned segments are further subdivided into finer segments. This is done until the instance space segments are pure (samples belong to only 1 class) or until the impurity is below an allowable maximum threshold value.

The split points of a decision tree are decided by using a suitable metric such as Entropy or the GINI index. The idea is to choose the best split point among all the possible candidate features such that the resultant split reduces the impurity of the current node by the largest margin. For instance, in the below figure, the dataset has 6 features (Country, Age, SS, Cscore, Oscore, Impulse). For splitting the root node, the feature Country is chosen, as the resultant split has the least Entropy score and maximum information gain when compared to the remaining 5 features (as explained in Part 1: Constructing the tree).

In Fig. 1, the split point of 0.77 for Country was chosen as it best divides the users from the non-users (almost all non-users are on the other side of the split point). Similarly, in the right leaf, SS is chosen as the feature with the highest information gain and the split point of 0.50 almost evenly splits the users and the non-users that are assigned to this node. This process continues till the specified maximum depth is reached or the minimum samples per leaf criteria hold. Specific examples of the DT classification on instances are discussed in Part 4.

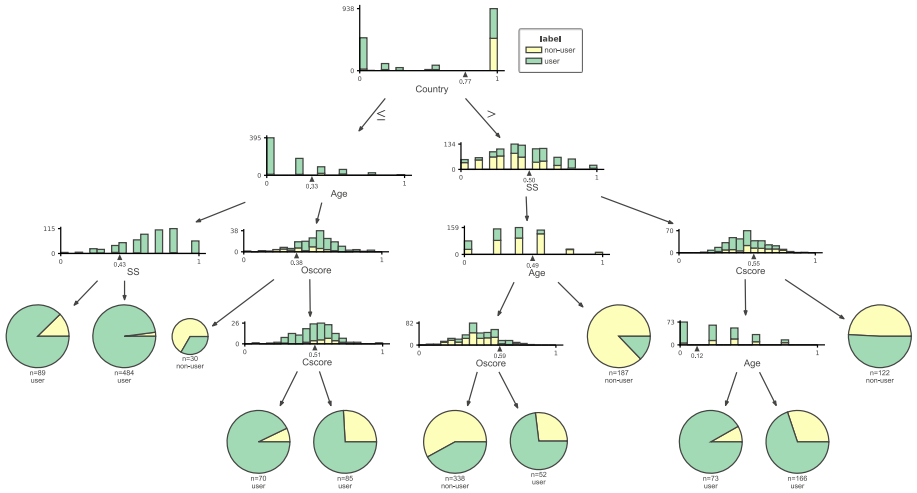


Fig 2. A visualization of the Decision Tree classifier showing class-wise distributions achieved in each split.

Edwin Thomas (Student Number: 300278402)

Part 3: Explain why the algorithm didn't do something else?

The decision tree grows based on the structure defined in Fig.1 and not any other alternative due to the relative feature importance of the given dataset. The importance of a given feature f is computed as the scaled sum of the entropy minimizations achieved using this feature in all possible decision paths from root nodes to the set of all leaves. The computed feature importance is as shown below:

Feature	Scaled Score	Rank
Age	0.0733398012347298	2
Country	0.1945151965472212	1
Oscore	0.02123058893167583	4
Cscore	0.017058741095277274	5
Impulsive	0.0	6
SS	0.050793755151587176	3

Table 1. Feature importance is determined by the relative decrease in entropy of the Decision Tree

Country has the highest feature importance followed by Age, SS, Oscore and Cscore. Impulsive has a relative importance of 0 which indicates that there are no split points that use this feature for dividing the instance space. This also indicates that, in all the split points, there was always another feature that had a higher information gain score than the Impulsive feature with the given depth and minimum samples per leaves constraints. The feature correlation matrix (Fig. 3) shows that the Impulsive and SS features have the highest correlation value. Therefore, the SS feature was given precedence by the Decision Tree whenever both were suitable candidates for achieving the split. In addition, SS has a higher correlation to the target variable as opposed to Impulsive. This reveals an interesting property of the decision tree; if two or more features have a very high correlation, the feature whose split produces the least entropy would be given precedence in most cases, thus ignoring the redundant features.

Country has the highest feature importance even though it is used only once for splitting the root node. This is because the resultant impurity decrease is the highest when computed on the whole dataset. The other feature splits then work on the reduced subset of the instance space and hence the expected reduction in entropy is comparatively lesser. This implies that they must be used more than once to have a higher feature importance. (Age which is used once in the left sub-tree and twice in the right sub-tree is ranked second in Table 1 and likewise for the other features).

As seen in Fig. 1 the best split point obtained after selecting a feature tries to separate the users (green bar) from non-users(yellow bar) such that in the resultant splits one of them always dominates. This is the idea behind entropy reduction and maximum information gain. Specific examples of the DT classification on instances are discussed in Part 4.

Edwin Thomas (Student Number: 300278402)

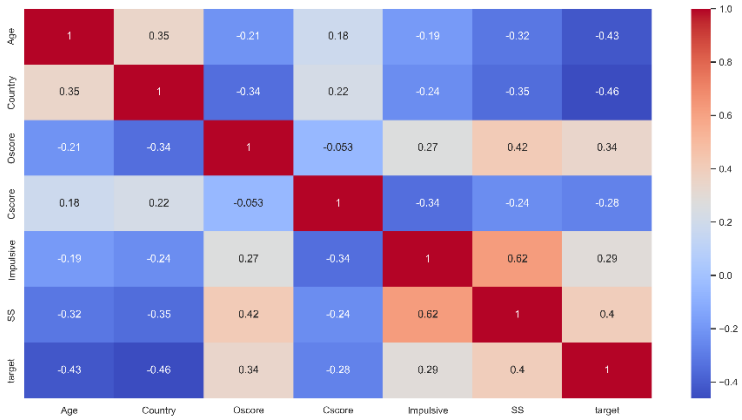


Fig. 3 Heatmap showing the correlation between features.

Part 3: Discuss when the algorithm succeeded and when it failed.

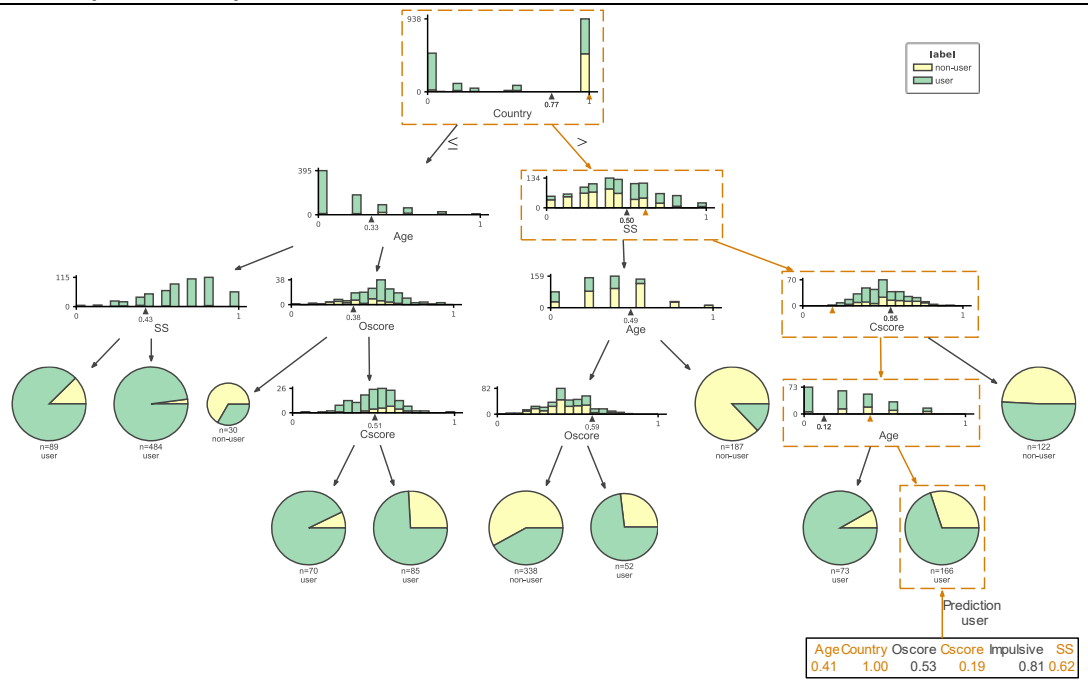
The algorithm succeeds when the input test case fits within the majority class of the leaf node obtained by traversing the Decision Tree. On the other hand, the algorithm fails under the following two scenarios:

1. All the test cases follow a feature distribution similar to the train and occupy the minority class in the leaf nodes. The prediction fails as the majority class is wrongly predicted for these samples in the non-homogenous leaf nodes.
2. An out-of-distribution test sample gets wrongly assigned to the majority class of a leaf node.

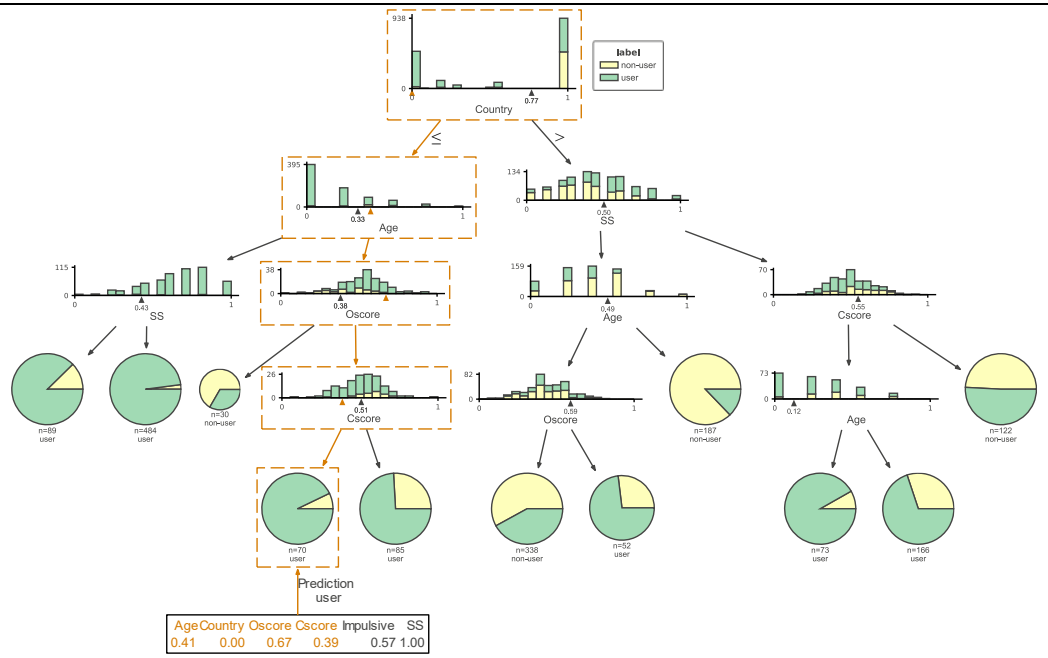
In both cases, samples are divided into False positives and False negatives depending on whether the majority class is ‘users’ or ‘non-users’.

Edwin Thomas (Student Number: 300278402)

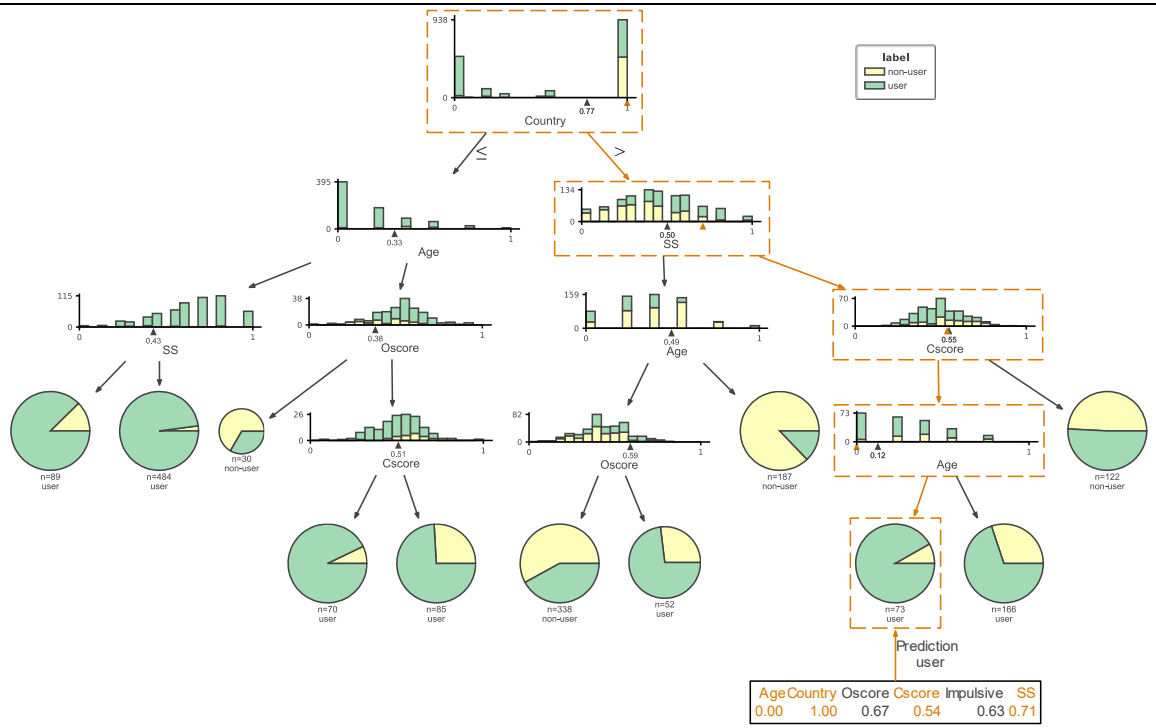
Shown below are success and failure cases of the DT classifier for both the classes:

Class	Predictions	Traced path of the prediction
Users	Correct (For this instance, Country=1.0 which is greater than 0.77 so we move to the right sub-branch, then SS score (0.62) also lies on the right branch, Cscore (0.19) lies on the left branch and Age (0.41)>0.12 , so the resultant prediction is user class as this the class with highest proportion of samples in the leaf node)	

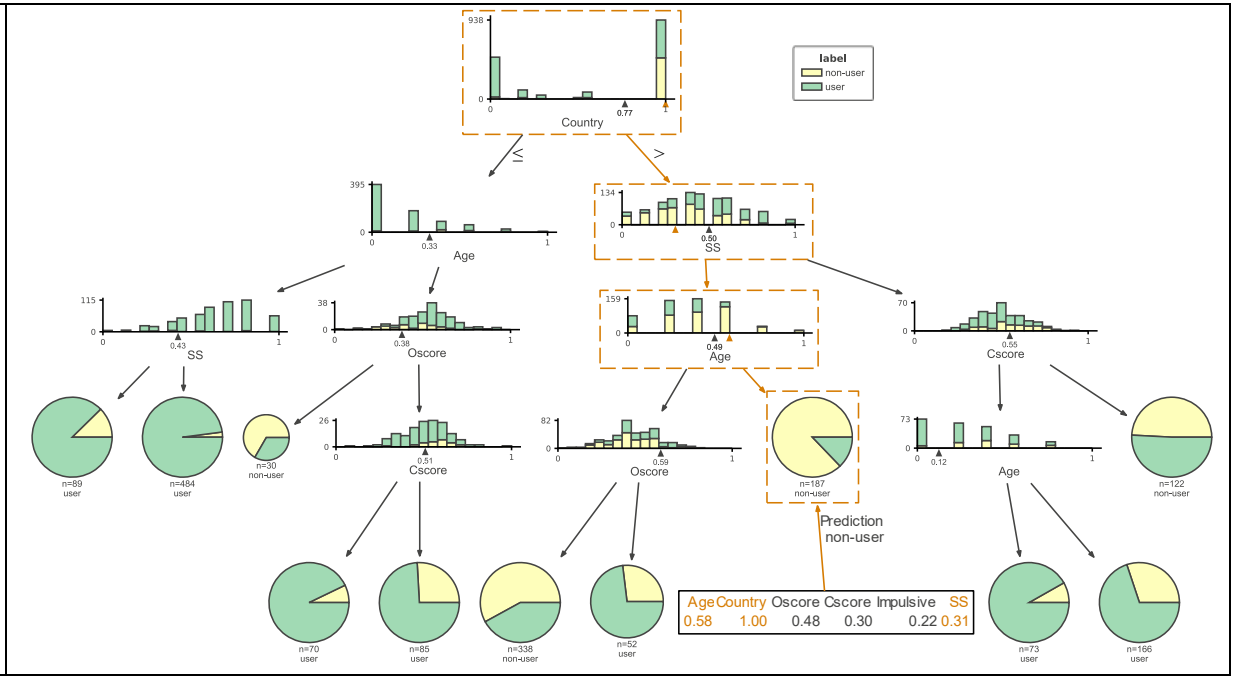
Edwin Thomas (Student Number: 300278402)

	Correct (In a similar fashion the values of the instance are used to make decisions on which sub-tree branch to move to, until we reach a leaf node. The majority class of the leaf node is given as the prediction, which in this case is again user class.)	

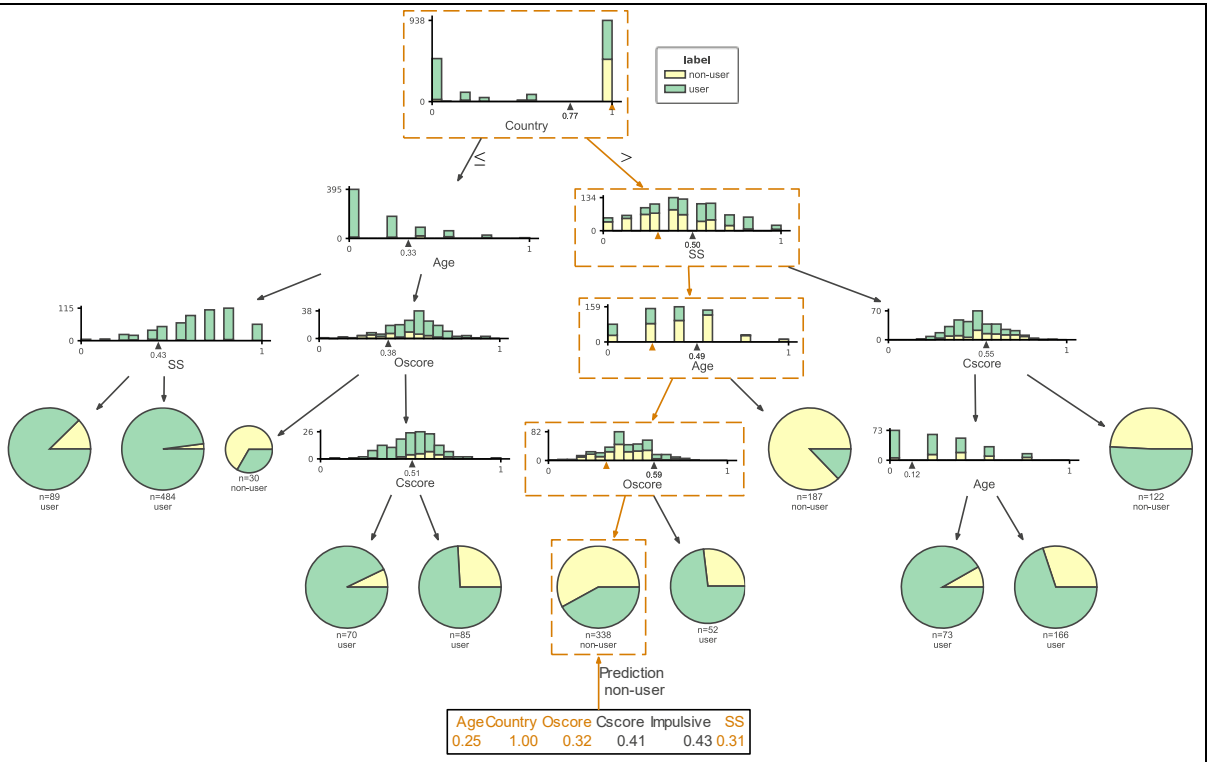
Edwin Thomas (Student Number: 300278402)

	Correct	

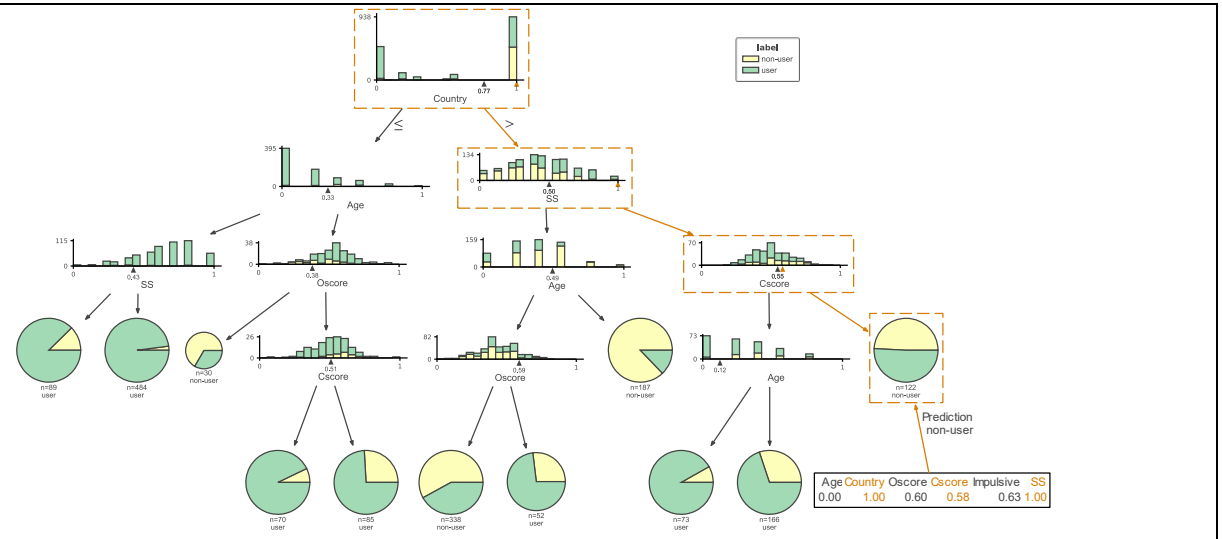
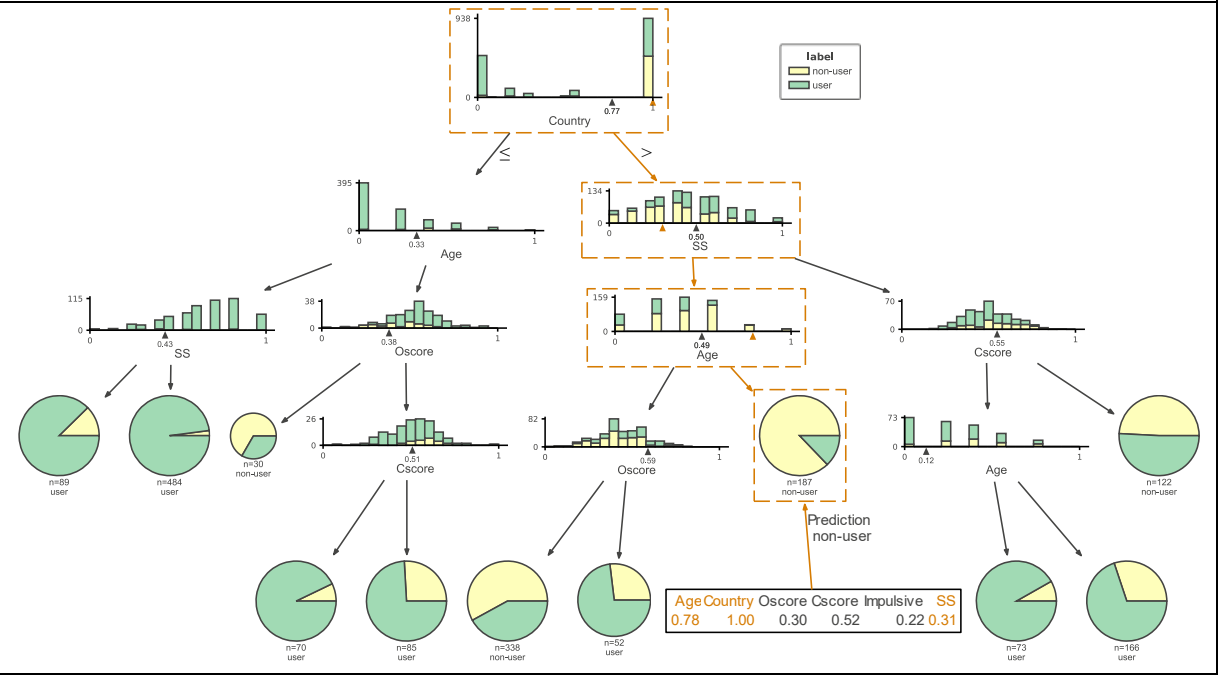
Edwin Thomas (Student Number: 300278402)

Users	In-correct (Only 3/6 features are used to make the decision. The test sample belongs to the minority class of the decision tree leaf. Clearly in this case if country value was less than 0.77, the user would have been classified correctly (Country (left)-> Age(right)->Oscore(right)->Cscore(left)	

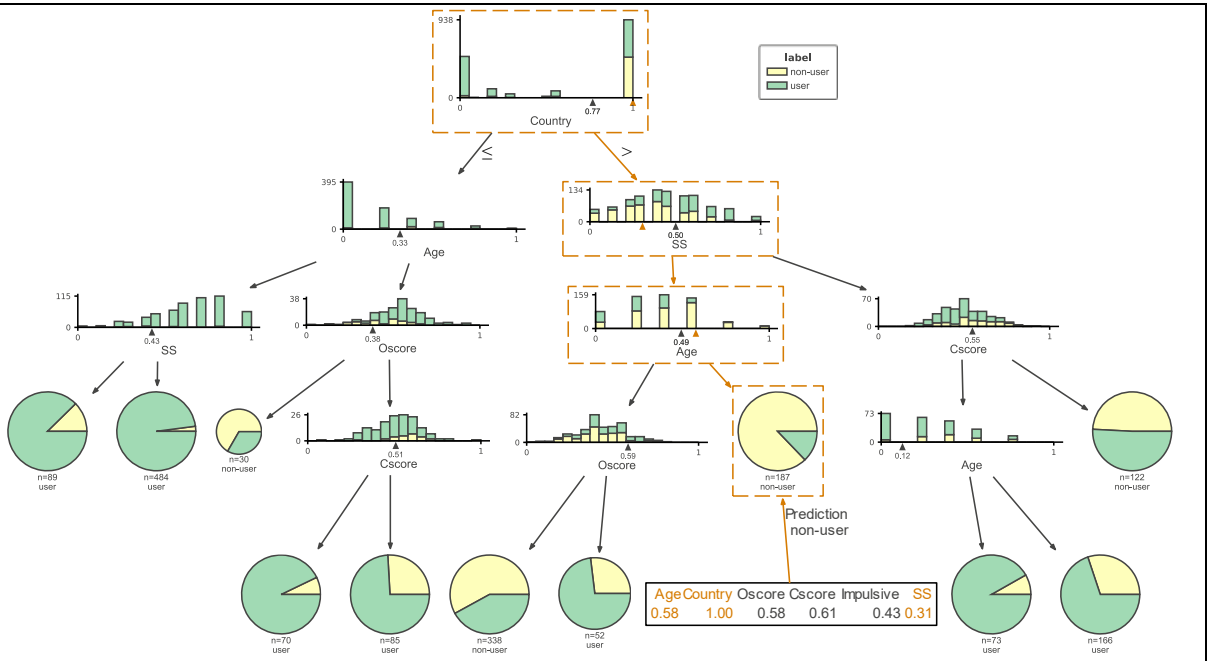
Edwin Thomas (Student Number: 300278402)

	In-correct (In this case again the mis-classified example belongs to the minority class. If Oscore>0.59 it would have been classified as a user correctly.)	

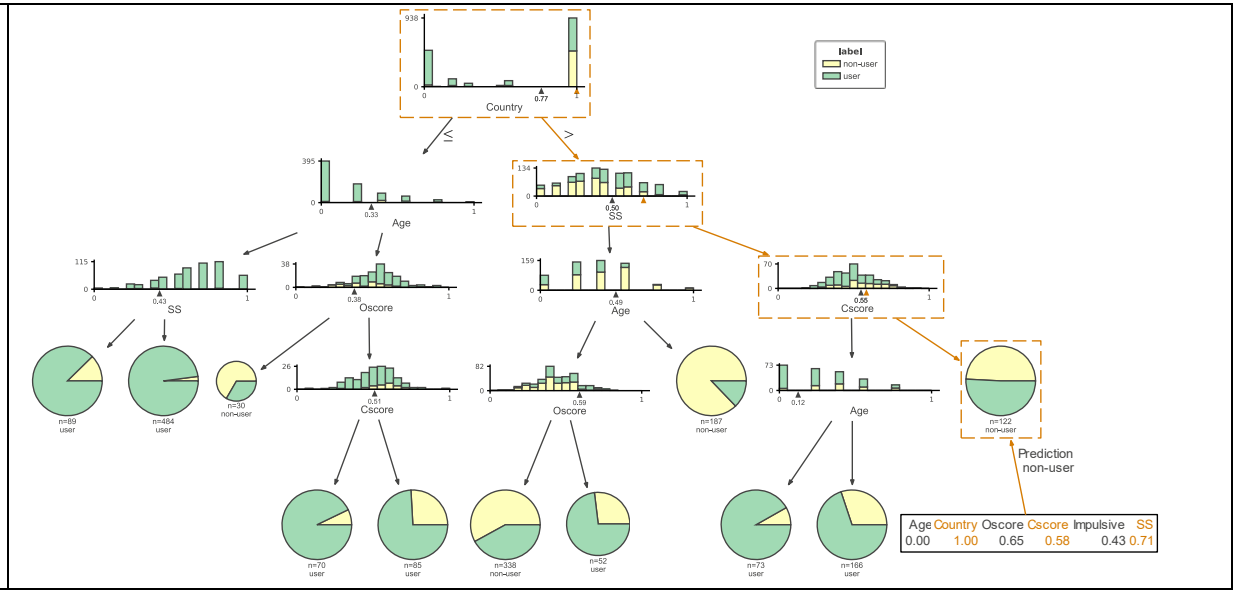
Edwin Thomas (Student Number: 300278402)

	In-correct (The misclassified example belongs to the minority class. In this case, if C-score was <0.55, the classification would be correct regardless of Age. This also means that the right-most leaf could have been further split to account for more homogenous segment splits)	
Non-users	Correct (In this case, tracing the DT leads to the leaf node with a larger proportion of the non-user class. It is to be noted that only 3 features are used to decide this case.)	

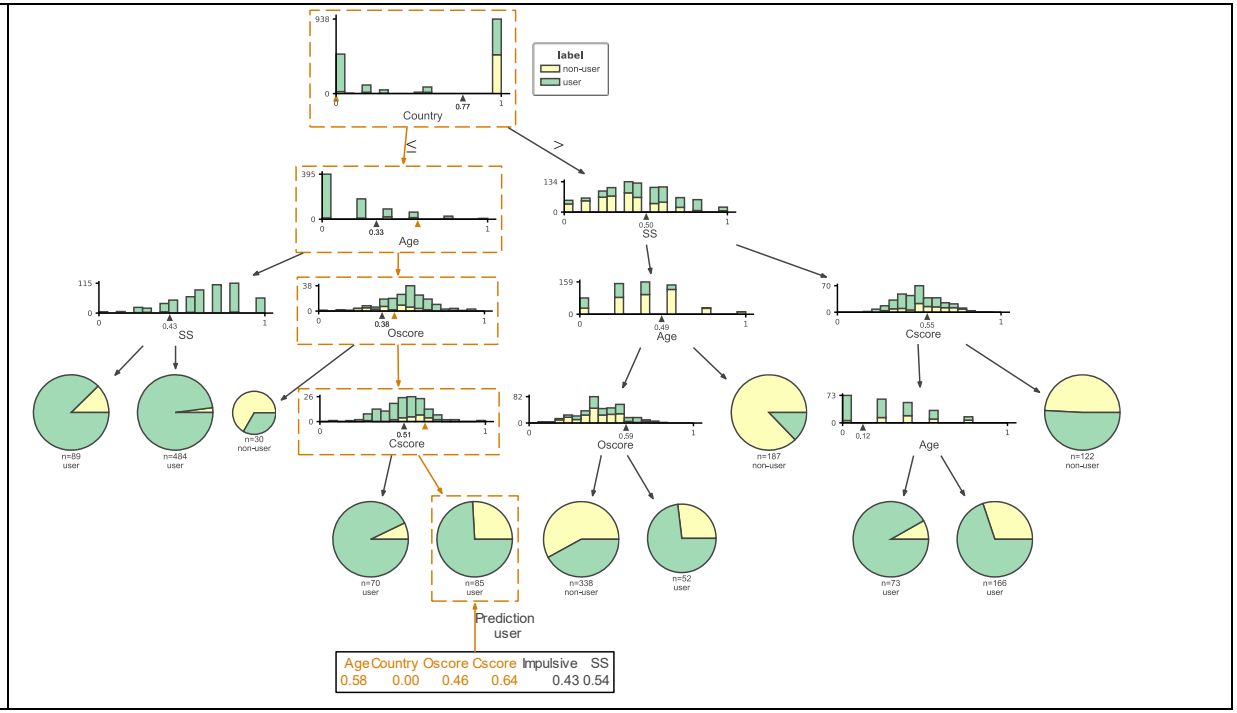
Edwin Thomas (Student Number: 300278402)

	Correct	

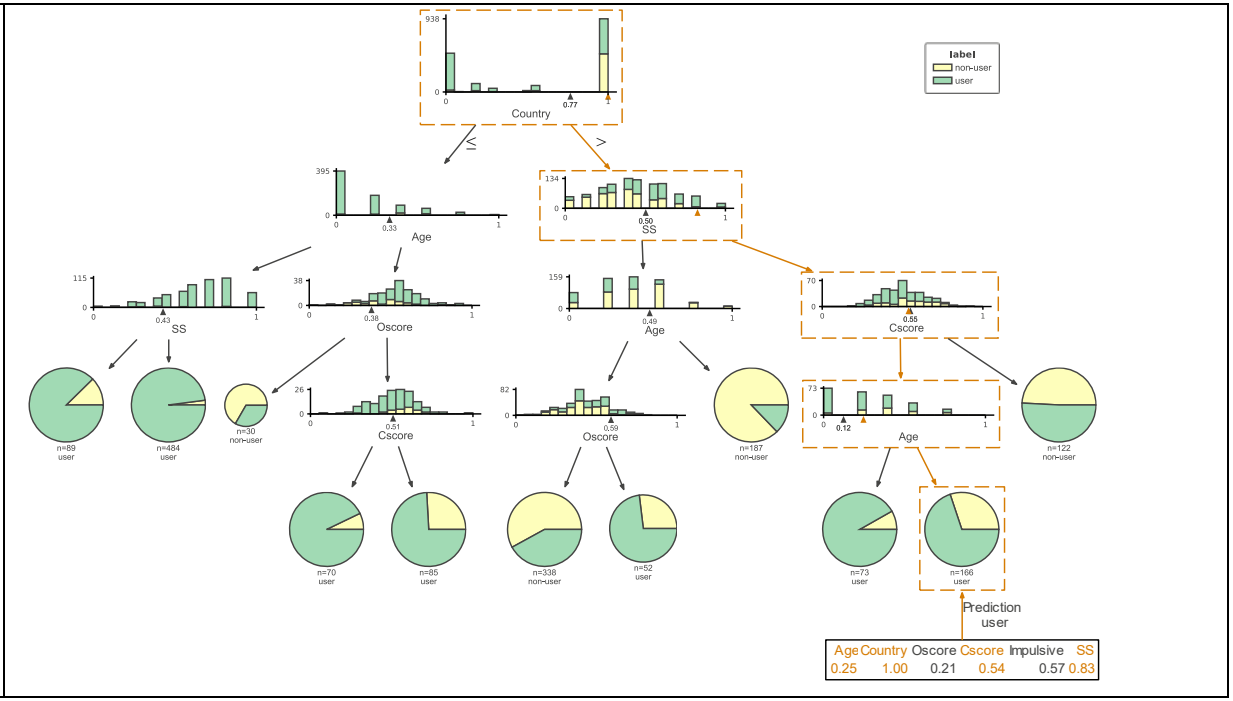
Edwin Thomas (Student Number: 300278402)

	Correct	

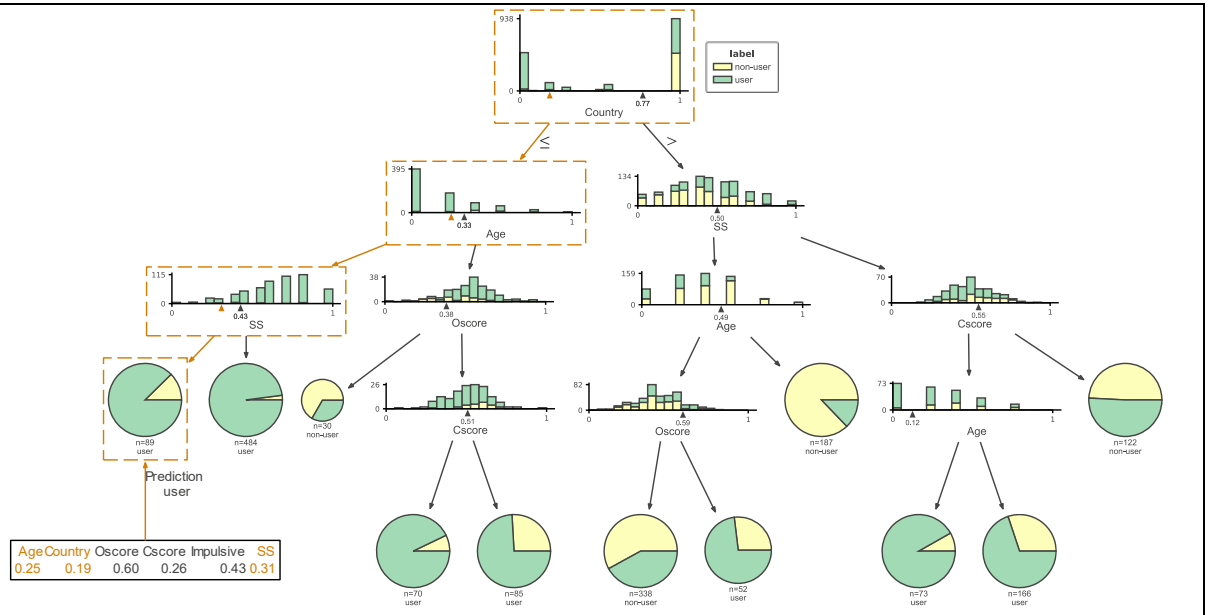
Edwin Thomas (Student Number: 300278402)

Non-users	In-correct (Here, again a non-user class is misclassified as a user class because the leaf node corresponding to the decision path has user as the majority class. If Oscore was less than 0.38, the sample would have classified correctly as a non-user.)	

Edwin Thomas (Student Number: 300278402)

	In-correct (A majority class based misclassification where if Cscore was greater than 0.55 the sample would have been classified correctly)	

Edwin Thomas (Student Number: 300278402)

	In-correct (The sample belongs to the small proportion of non-users in the terminal leaf (left-most). This can be considered as an out of distribution example as multiple feature values need to be changed in-order to correctly classify the example.)	

To further study the relationship between the errors observed in users' and non-users classes, the lowest ancestor in the DT for varying subsets of test samples for the error cases was recorded. Fig 4. and Fig 5. demonstrates the same for the users and non-users class where the y-axis denotes the number of test samples with errors where the last column corresponds to the whole test set. The x-axis is the quantity by which the support count of the number of samples with the least common ancestor is reduced from the considered subset of samples given by the y-axis. The corresponding value of the matrix gives the node id in the DT (Fig. 1) corresponding to the lowest common ancestor due to which the error could have propagated. We observe that node id 10 is the point from which the user's class gets misclassified and a similar trend for the non-users class. This indicates that the SS split at depth-1 and their sub-trees cause most of the erroneous misclassifications (**right subbranch** of the root node).

Edwin Thomas (Student Number: 300278402)

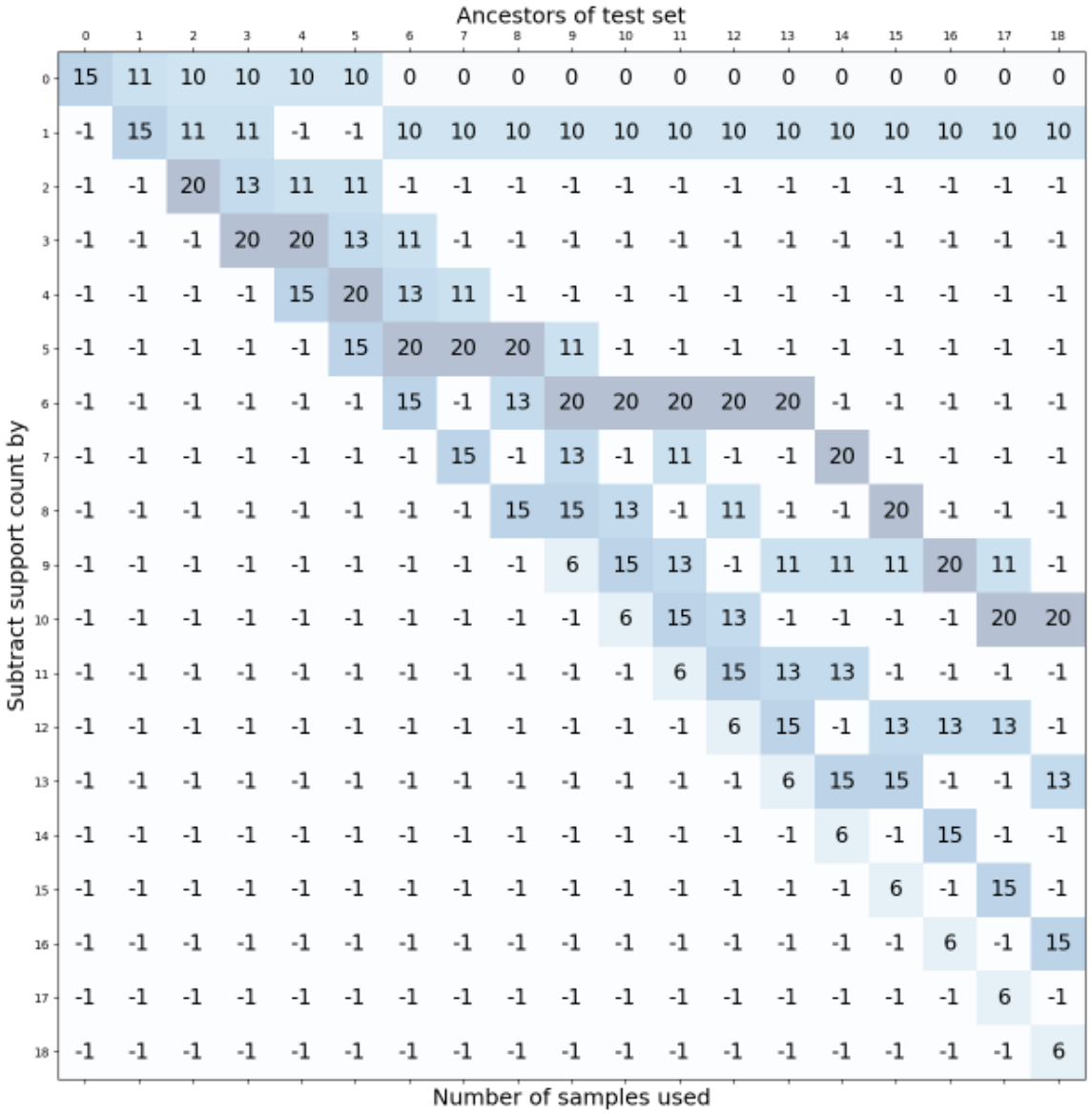


Fig 4. Node ids of the closest ancestor of test samples with error in the DT for the user's class

Edwin Thomas (Student Number: 300278402)

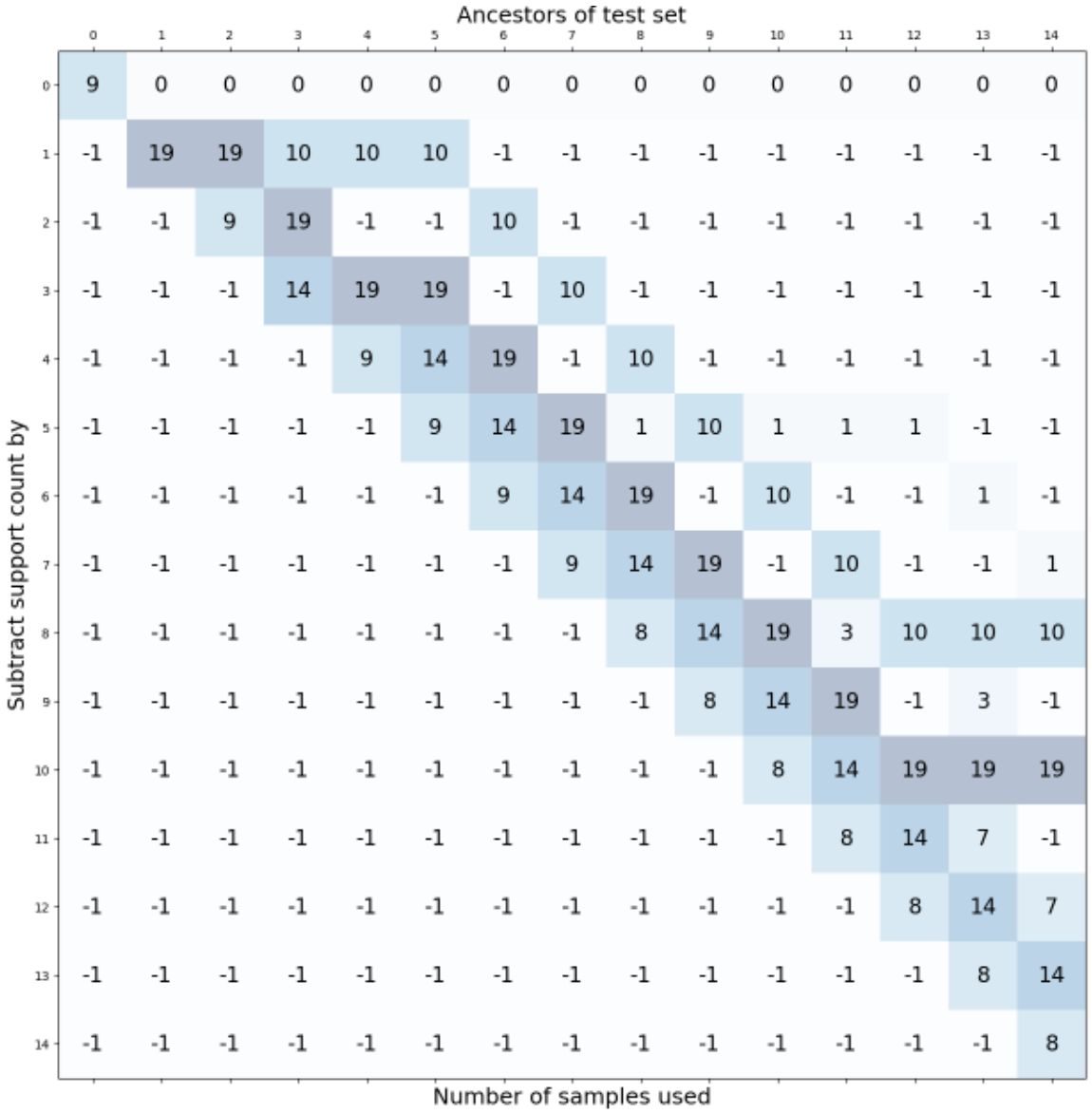


Fig 5. Node ids of the closest ancestor of test samples with error in the DT for the non-user's class

Edwin Thomas (Student Number: 300278402)

Part 4: Explain how you would decide if the resultant model can be trusted

One way to measure the trustworthiness of a model is by assessing its useability for various applications. The useability can be determined by observing the application-specific metric scores of the decision tree classifier. A higher score can indicate the trustworthiness of the classifier. Listed below are some of the real-world domains where the DT classifier can be used:

1. **Medical Domain:** Here, the **false negatives** should be generally very low, as the detection of the presence of a disease is of at most significance. For instance, for cancer detection, a benign tumor is classified as malignant and is considered more tolerable than the other way around. The DT classifier has a **recall** rate of 84.92%. This implies that ~16/100 +ve cases are missed. This makes the model less trustworthy for critical medical diagnosis-related tasks where recall rates should ideally touch 99%.
2. **Search Engines:** In search engine web page retrieval tasks, the identification of true negatives is less crucial than **true positives**. This is because there can be only 1 correct web page per query. Therefore, **precision** can be a good metric to assess the trustworthiness of the model. Assuming the current DT classifier was trained for this task and achieved a precision of 87.70%, it can be a good model as ~88% of the time it retrieves the right webpage.
3. **Fraud Detection (all applications where there exists significant class imbalance):** In such applications, the **macro_accuracy** or the **average recall** metric can be a good indicator to assess the trustworthiness of the decision tree classifier model. This is because the metric equally weighs the accuracy of both classes by averaging them and hence gives an unbiased view of the classifier.

In most other applications where the class imbalance doesn't exist and the accuracy of both classes is equally important, **accuracy** can be a good measure to judge the trustworthiness of the DT classifier. Table 2. shows some of the other metrics computed on the DT classifier used for this assignment.

Metric Name	Value (%)
Precision	87.70
Recall	84.92
Negative Precision	71.21
f1	86.29
sensitivity(Recall)	84.92
specificity(Negative Recall)	75.81
gmean	80.23
accuracy	81.91
macro_accuracy	80.36

Table 2: Metric of the Decision Tree Classifier

Edwin Thomas (Student Number: 300278402)

Part 5: Explain how the algorithm could potentially improve its predictions.

The DT classifier algorithm can be improved by observing two specific aspects of the performance dips:

1. Variance
2. Bias

An ideal model achieves low variance and low bias. A DT can be **highly biased** if we constrain the depth of the model or prune the tree to the extent it is highly biased and under-fits the dataset. On the other hand, the DT has a **high variance** when the model overfits the training data (an extreme case could be when every leaf represents a single instance of the instance space is partitioned with the smallest possible granularity).

In the former case, we can employ a **Gradient Boosting Ensemble** built on DT. The gradient boosting algorithm first starts by training the tree on the initial dataset. The samples that are misclassified due to the bias factor, are then weighted more than the correctly classified samples to train another DT model. This process continues till no significant error cases are observed. Finally, a weighted combination of the decision tree ensembles is employed to make a decision (where higher weights are assigned to the more confident model or the model with lesser error rates).

The variance of the DT classifier can be reduced by a **Bagging ensemble** approach, commonly referred to as **Random Forests**. The idea behind these models is to use a multiple DT instead of a single tree to make decisions; this introduces more confidence in the prediction and enough randomness to reduce the variance of a single DT. A series of DT classifiers can be trained on bootstrapped samples of data (subsets of data from the original set picked using replacement strategy). Further, we can initialize the DT such that each one sees only a subset of the features. After this is done, the trained models learn to focus on different subsets of features and data distributions and give predictions from several perspectives (reducing the effect of overfitting). Finally, a voting-based mechanism can be used to obtain a single prediction from the ensemble of the models. Further, the extent of overfitting in DT can be reduced by using a higher pruning factor, lesser depth, and higher minimum samples per leaf.

Therefore, bagging can be used to reduce variance and boosting to decrease bias, thus improving the overall performance of the DT classifier algorithm. Another method to improve tree predictions is by converting it to a ranking estimator and thresholding the users and non-user class based on the operating condition in the test set. A **ROC curve** can be used for the same where we can move the **accuracy isometric** from the top left point (ROC heaven) parallelly downwards till it intersects the ROC plot. This point would be the point where maximum accuracy can be achieved. The accuracy isometric needs to be multiplied by the class ratios if we know the distributions beforehand.

Finally, better **feature engineering** and **feature selection** techniques can be employed that are suitable for the Decision Tree model. For instance, instead of using a statistical technique like ANOVA, the feature selection can be done by first balancing the classes and training a decision tree model on it, followed by the selection of features based on relative feature importance returned by the tree (Part 3: Feature importance); highly correlated features such as SS and Impulsive could be avoided and more superior features can be chosen.