Application of an Artificial Intelligence System to Detect and Treat Heart Disease

Ayman Mezghani
ayman.mezghani@epfl.ch

Victoria Adcock
victoria.adcock@epfl.ch

In an ever-evolving world, data has become readily accessible and crucial to modern society. We are able to analyse collected data to identify patterns and trends. Artificial intelligence (AI) applications in medicine primarily use computer techniques to perform clinical diagnoses and suggest treatments. AI has the capacity for detecting meaningful relationships in a dataset, and it is largely used in many clinical situations to diagnose, treat, and predict results. For this project, we apply a decision tree approach on datasets, in order to predict whether a patient has heart disease. By re-evaluating our first model and considering an alternative approach we are able to improve the performance of our model, and ultimately achieve an accuracy of 62.5%.

TASK 1

We used training data which had previously been discretised into bins, in order to obtain a model to predict whether a patient has heart disease. By implementing the Iterative Dichotomiser 3 (ID3) algorithm, we were able to successfully generate a decision tree with 70 leaves (see figure 1). ID3 is a *top-down greedy* algorithm which repeatedly divides features into two or more groups at each step. It starts building from the root node, and the feature with the greatest information gain is then selected at each iteration, as it calculates how well a given feature separates or classifies the target classes. Our model starts the classification process by separating the patients based on their chest pain type (cp). Depending on the patients' answers, their information will be passed through the appropriate subtree, and thus, the process continues until a classification is determined. At most, the tree will consider seven attributes in order to classify a patient as having/ not having heart disease (the height of the tree is 7), and a minimum of two attributes will be considered. Our model will consider on average 4.8 attributes before a classification can be determined.

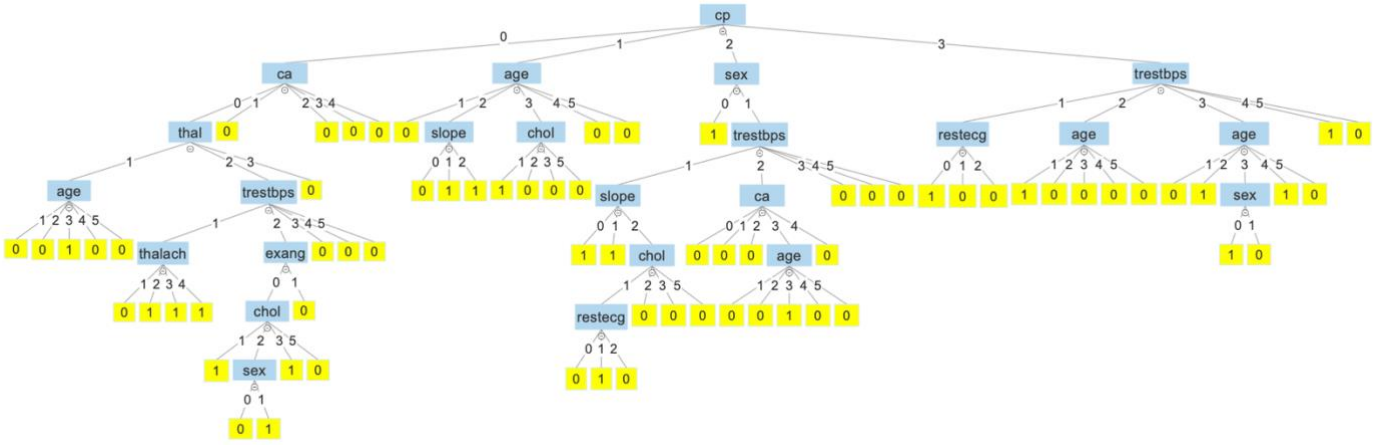

Figure 1: Decision tree generated after performing the ID3 algorithm on training data, which had been discretised into bins.

TASK 2

Once our decision tree had been generated, it was necessary to test the accuracy of the model on a similar test dataset. Our model was able to correctly classify a patient with an accuracy of 56.25%; 45 out of the 80 patients were correctly classified. As determined by the resulting confusion matrix (see figure 2), our model has high precision and specificity (with values of 0.94 and 0.86 respectively). The high precision indicates that the majority of those who were classified as having heart disease, did in fact have heart disease. Likewise, the high specificity shows that the majority of people who did not have heart disease were correctly classified as such. However, our model has low sensitivity (0.50). This could be fatal in a real-world context, as this indicates that only half of the patients with heart disease were correctly identified as having heart disease. Thus, the patients with heart disease who were misclassified as healthy would miss out on potentially lifesaving treatments. As a result, our model has a high type 2 error. It would be preferable to optimise sensitivity in order to reduce the number of false negative cases, even if this would result in an increase in false positive cases.

n = 80		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP) 33	False Negative (FN) 33	Sensitivity $\frac{TP}{TP+FN}$ = 0.50
	Negative	False Positive (FP) 2	True Negative (TN) 12	Specificity $\frac{TN}{TN+FP}$ = 0.86
		Precision $\frac{TP}{TP+FP}$ = 0.94	Negative Predictive Value $\frac{TN}{TN+FN}$ = 0.27	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$ = 0.56

Figure 2: Confusion Matrix for our decision tree model.

TASK 3

By analysing the training data, we were able to establish a list of *'initial facts',* as we already know which patients have heart disease, and their information regarding all the attributes recorded. Therefore, a list was created which contains all the information of the patients in the training set, where the i_{th} element in the list is a list containing all the attributes, with their corresponding values, for the i_{th} patient in the set.

By implementing the Depth-First Search (DFS) algorithm, we were able to find all of the paths in our decision tree, starting from the root node and ending at the terminal nodes. We would use these paths to determine a set of *rules* stated by the tree; these rules explain the outcome of a classification. Since there are 70 end nodes, this resulted in 70 rules being established.

TASK 4

If we have a model which is able to predict whether a person has heart disease, it would equally be beneficial to be able to point to possible treatments. By separating the patients in the test data that were diagnosed as having heart disease, from the patients that were classified as healthy, and by tracing the path they followed in the decision tree, we are able find a potential treatment. As we had already established a set of rules for both positive and negative classification, we are able to use abduction to find a cause of heart disease and offer a solution. Since a person's age and sex cannot be easily changed, nor can be a cause for heart disease, thus, cannot be considered faulty, these attributes were ignored when offering treatments. By comparing the rules for negative classification to the paths taken in the tree by the patients classified as having heart disease, we were able to find an attribute that leads to positive classification, and consequently highlight a change which results in negative classification.

The algorithm to find treatments successfully found a solution for all 35 patients who were diagnosed as having heart disease, according to our decision tree, with two or less attribute changes.

TASK 5

Our original decision tree was determined by a training dataset whose attribute values had previously been discretized into bins. However, this can lead to loss of performance. Instead, we generated a more advanced decision tree which takes values from a continuous dataset, and performs a split on the attribute values, based on the notions of entropy minimisation. Therefore, the patient will be categorised based on whether their attribute value is greater or less than the split. The updated decision tree is generated in a similar fashion to the original tree, using the ID3 algorithm, however, the output will be a binary tree. Figure 4 is a representation of our updated decision tree, indicating these splits. The split for a particular attribute is achieved by sorting the data into ascending order, according to the values of this attribute. Once sorted, the average for each adjacent pair of values is calculated and the entropy is determined for each average. The value with the lowest entropy is consequently selected as the split point, as this results in the highest information gained. The attribute values that are less than the split point pass through to the left subtree. Subsequently, the attribute values that are greater than the split point pass through to the right subtree.

Using this more sophisticated technique for classifying the data, we were able to increase the accuracy of our model up to 62.5%, when performed on a similar test set. As demonstrated by the resulting confusion matrix (see figure 3), the precision and specificity remain high, and their corresponding values have equally increased from our first model. Although still relatively low for what would be acceptable in a real-world setting, the sensitivity for the updated decision tree has improved, and increased to 0.56. This indicates that 56% of patients with heart disease were correctly diagnosed as having heart disease, thus, reducing the type 2 error.

n = 80	Predicted Class: Positive	Predicted Class: Negative	
Actual Class: Positive	True Positive (TP) 37	False Negative (FN) 29	Sensitivity $\frac{TP}{TP+FN}$ = 0.56
Actual Class: Negative	False Positive (FP) 1	True Negative (TN) 13	Specificity $\frac{TN}{TN+FP}$ = 0.93
	Precision $\frac{TP}{TP+FP}$ = 0.97	Negative Predictive Value $\frac{TN}{TN+FN}$ = 0.31	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$ = 0.63

Figure 3: Confusion Matrix for the advanced decision tree.

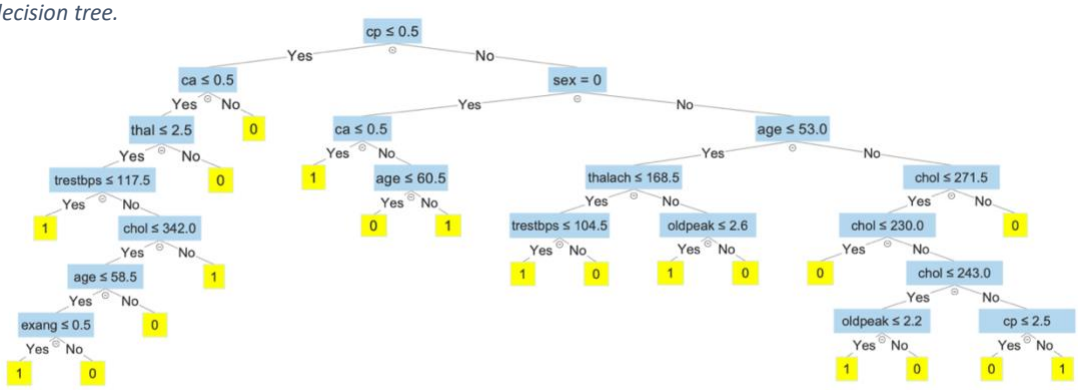

Figure 4: Updated, advanced decision tree using a continuous dataset.

CONCLUSION

Although we were able to generate a decision tree with accuracy greater than 50%, using discrete datasets, it is favourable to use a continuous data set and perform a *'split'* per attribute, as this yields a higher accuracy. By implementing a more sophisticated tool, we were able to improve performance, and achieve a final accuracy of 62.5%. An advantage to using a decision tree classification system is that they are invariant to data scaling. Since each attribute is processed separately, and the potential splits do not depend on scaling, pre-processing techniques such as normalisation and standardisation are redundant. Particularly, decision trees work well when we have features that are on completely different scales, or a mix of binary and continuous features.

Alternatively, decision trees are not quite as robust as other classification methods. A small change in the data can cause a large change in the final estimated tree. They generally do not have the same level of predictive accuracy. This can be witnessed by the confusion matrices of our two models; they have a relatively high type 2 error, which would not be acceptable in this context, when applied in the real-world. Therefore, other classification methods would more likely be used. Nonetheless, by aggregating multiple decision trees, using methods like *bagging, random forests*, and *boosting*, the predictive performance of decision trees can be substantially improved.