Imperial College
London

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Introduction to Machine Learning - Decision Trees

Author:
Richard Mardiat, James Rowbottom, Francis Rhys Ward, Alexey Zakharov

Date: November 4, 2019

Contents

1 Implementation

1.1 Notation

First we establish our notation for the problem:

1. We have 7 wifi signals which we call Attributes = $\{W_1, ..., W_7\}$
2. We have 4 rooms which we identify with the Room Labels = $\{1, 2, 3, 4\}$
3. A Sample of data $S = \{s_1, ..., s_{2000}\}$ s.t. $s_n[:8] \in$ range(-100,0) and $s_n[8] \in \{1, 2, 3, 4\}$
4. depth = d

We aim to implement an algorithm that can build a Decision Tree to predict the Room (Label) of a datum given the 7 wifi signal strength values. Below we outline the high level implementation of this algorithm, later sections provide further detail:

1. **Build (unpruned) Decision Tree (1.2)** and perform **Cross Validation (1.5)** to generate unpruned **Evaluation (1.4)** statistics.
2. **Build (unpruned) Decision Tree (1.2)**, perform **Pruning (1.6)** and perform **Pruned Cross Validation (1.6)** to generate pruned **Evaluation (1.4)** statistics

1.2 Building the Decision Tree

The process of building a decision tree is deterministic (in our implementation), and depends entirely on the dataset inputted to the algorithm. At each node, our program decides how to split the data based on the implemented `find_split()` function. This functionality is, however, implicit in the operations of method `decision_tree_learning()` – please see `README` file to learn how to run code. Below we explain the process:

1. Given Training data S with attributes $\{W_1, ..., W_7\}$ and labels = $\{1, 2, 3, 4\}$
2. We represent the Decision Tree as a list of lists of **Nodes**

   $$\textbf{Tree Representation} = \Big[[\textbf{Node}_1], [\textbf{Node}_{2,1}, \textbf{Node}_{2,2}], ..., [\textbf{Node}_{d,1}, ..., \textbf{Node}_{d,...}]\Big]$$

   Where each node has attributes:

   - Decision **Tree**
   - Parent **Node**
   - Children nodes: C_L=$\textbf{Node}_{(d+1,L)}$, C_R=$\textbf{Node}_{(d+1,R)}$
   - `split_attribute` containing all the information about the split (incl. split datasets, split attribute, split value and maximum gain) in a tuple.
   - Depth = d how many layers down a node is on a tree
   - `label` $\in \{1, 2, 3, 4\}$, where only leaves receive labels
   - `data` = S_N, where C_L.`data` = S_L, C_R.`data` = S_R

3. The rule at each **Node** can be fully defined by the tuple:($k_{\texttt{split}}$, `split_value`), where $k_{\texttt{split}}$ is the index of the attribute the rule is applied on and `split_value` is the value we split the data, such that

$$S_L = \{s_n | W_k(s_n) \leq \texttt{split\_value},\ n = 1, ..., N\}$$
$$S_R = \{s_n | W_k(s_n) > \texttt{split\_value},\ n = 1, ..., N\}$$

The `find_split()` **(1.3)** algorithm gives us a formulaic way in which we determine this tuple.

4. Nodes at the lowest depth of the tree with no children are called leaves. We call a leaf pure if all data points with its subset have the same label.
5. Apply `decision_tree_learning()` with `find_split()` on our Training data and the subsets of Training data at each node until every node is a Pure leaf.

We can write the Pseudo code for `decision_tree_learning()`:

Algorithm 1 `decision_tree_learning`

```
procedure DECISION_TREE_LEARNING($S$)
    Given sample data $S_N = \{s_1, ..., s_n\}$ initialise the Parent Node
    Run find_split($S_N$) to get ($k_{\texttt{split}}$, split_value)
    Initialise Children Nodes: $C_L$ = Node$_{(d+1,L)}$, $C_R$ = Node$_{(d+1,R)}$
      # Run while new children nodes are being added
    new_nodes_being_added = True
    while new_nodes_being_added do:
        node_list = current list of nodes
      # Repeat for each node (excluding the starting node)
      # Starting with last nodes
      # Run through each child in a layer
        for child in deepest layer do:
            child.find_split()
            child.create_children()
        if node_list did not change then:
            new_nodes_being_added = False
    Return trained DecisionTree
```

1.3 Find_split()

To split the data we calculate the information gain yielded by splitting the data on every possible value, which each attribute (wifi strength) can take. Then, we choose the attribute and value that yield the maximum gain. We consider only those values which are taken within the data set we wish to split.

Algorithm 2 `find_split`

```
1: procedure FIND_SPLIT($S$)
2:     Given sample data $S_N = \{s_1, ..., s_N\}$ at each Node
3:     for each attribute $W_k$ for $k$ in $\{1, ..., 7\}$ do
4:         Set values = $\{s_1, ..., s_m\}$ (unique values realised for attribute $W_k$)
5:         for each split_value in values do
6:             $S_L = \{s_n | W_k(s_n) \leq split\_value, n = 1, ..., N\}$
7:             $S_R = \{s_n | W_k(s_n) > split\_value, n = 1, ..., N\}$
8:             Calculate:
9:             Entropy: $H(S) = -\sum_{i \in Labels} P(i|S) \log_2 P(i|S)$
10:            Information Gain: $G(S_N, W_k, s_i) = H(S_N) - (\frac{|S_L|}{|S_N|} H(S_L) + \frac{|S_R|}{|S_N|} H(S_R))$
11:     Return the $W_{k_{max}}$ and split_value which yields the maximum gain
```

1.4 Evaluation

Now that we have an automatic process to train decision trees, we can evaluate the accuracy of our trees on the provided datasets. This occurs in three places:

- Evaluating the initial unpruned Trees to collect base statistics
- Evaluation during pruning (on validation set) as a criterion for pruning
- Evaluation of the pruned trees to compare against base statistics

The *evaluate*() function is implemented as follows:

Algorithm 3 Evaluate

```
1: procedure EVALUATE($test\_data, tree$)
2:     Generate test set with labels predicted by tree
3:     Generate the confusion matrix
4:     Generate the statistical measures from the confusion matrix
5:     Return the statistical measures
```

This generates the statistical measures below:

- **Confusion Matrix** for example for Room 1:

	Room 1 Predicted	Room 2 Predicted	Room 3 Predicted	Room 4 Predicted
Room 1 Actual	TP	FN	FN	FN
Room 2 Actual	FP	TN	.	.
Room 3 Actual	FP	.	TN	.
Room 4 Actual	FP	.	.	TN

- **Classification Rate** = $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$, **Recall** = $\frac{\text{TP}}{\text{TP+FN}}$
- **Precision** = $\frac{\text{TP}}{\text{TP+FP}}$, **F1 Measure** = $2\frac{\text{Precision}\times\text{Recall}}{\text{Precision+Recall}}$

1.5 Cross validation

Before pruning we evaluate our decision tree algorithm using 10-fold Cross Validation on both the clean and noisy datasets. This process is executed as follows:

Algorithm 4 cross_validation

```
procedure CROSS_VALIDATION($S$)
    Randomly shuffle the data $S$                          2,000 data points
    10-fold partition the data into 10 subsets         10×200 data points
    for each subset do:
        test_data = subset                                   200 data points
        training_data = $S\setminus$ subset                  1,800 data points
        initialize a tree on the training data
        evaluate the tree on the test_data                   10 statistics
    Return the average statistical measures
```

In this way we can collect the average classification rate, confusion matrix, etc before pruning.

1.6 Pruning and Pruned Cross Validation

We implemented pruning on our unpruned trees by considering every parent node with only children that are leaves and calculating whether pruning that node would result in its accuracy to increase or stay at the same level (measured in classification rate). As per the pruning validation states, we use the Validation data for this step. The details of this can be seen in the pseudo code below:

Algorithm 5 Prune

```
procedure PRUNE(Tree,$S$)
    Given Validation data $S$ and unpruned Tree
    node_list ← Tree.node_list
    current_accuracy ← evaluate(S,Tree)
    for each layer in the Tree starting from the deepest layer do
        for Node in layer do
            if Node has no children then
                Continue
            else
                if Node has no grandchildren then
                    Make a copy of children
                    Remove children from node_list
                    pruned_accuracy ← evaluate(S,Tree)
                    if pruned_accuracy ≥ current_accuracy then
                        current_accuracy ← pruned_accuracy
                    else
                        Append removed children back to node_list
    Remove empty layers
    Return pruned Tree
```

To evaluate the statistical measures on pruned trees we conduct a 10-fold cross validation as follows:

Algorithm 6 `prune_validation`

```
 1: procedure PRUNE_VALIDATION($S$)
 2:     Randomly shuffle the data $S$                                    2,000 data points
 3:     10-fold partition the data into 10 subsets                       10×200 data points
 4:     for each subset (subset_test): do
 5:         test_data = subset_test                                      200 data points
 6:         9-fold partition $S\setminus$ subset_test into 9 subsets      9×200 data points
 7:         for each other subset in $S\setminus$subset_test, (subset_v) do:
 8:             validation_data = subset_v                               200 data points
 9:             training_data = $S\setminus$ (test_data $\cup$ validation_data)   1,600 data points
10:             initialize a tree on the training data
11:             apply prune() the tree using the validation data
12:             evaluate the tree on the test_data
13:             collect the statistical measures                         90 statistics
14:     Return the average statistical measures
```

Now we can compare our average results pre and post pruning.

2 Cross Validation results

2.1 Results

We now report the average statistical measures yielded from the cross validation before and after pruning, and on each of the clean and noisy data sets. We report the average: classification rate, confusion matrix (CM), and precision, recall, and F1 measure for each Room label. Note that different trees and results are likely to be generated each time, this is only due to the randomness introduced when shuffling the data.

2.1.1 Pre-Pruning Clean Results

Classification rate	average depth	min depth	max depth
0.9695	14.1	12	16

$$CM = \begin{pmatrix} 49.4 & 0. & 0.5 & 0.1 \\ 0. & 47.7 & 2.3 & 0. \\ 0.4 & 1.9 & 47.4 & 0.3 \\ 0.4 & 0. & 0.2 & 49.4 \end{pmatrix}$$

Room label	precision	recall	F1
1	0.9841	0.988	0.986
2	0.9617	0.954	0.9578
3	0.9405	0.948	0.9442
4	0.992	0.988	0.99

2.1.2 Pre-Pruning Noisy Results

Classification rate	average depth	min depth	max depth
0.8095	20.0	18	23

$$CM = \begin{pmatrix} 38.4 & 2.9 & 3.1 & 4.6 \\ 2.7 & 41.6 & 3.4 & 2. \\ 3.6 & 3.7 & 41.2 & 3. \\ 2.9 & 2.8 & 3.4 & 40.7 \end{pmatrix}$$

Room label	precision	recall	F1
1	0.8067	0.7837	0.795
2	0.8157	0.837	0.8262
3	0.8063	0.8	0.8031
4	0.8091	0.8173	0.8132

2.1.3 Post-Pruning Clean Results

Classification rate	average depth	min depth	max depth
0.9677	9.388	4	16

$$CM = \begin{pmatrix} 49.7 & 0. & 0.211 & 0.07 \\ 0. & 47.69 & 2.31 & 0. \\ 0.7 & 2.08 & 46.87 & 0.36 \\ 0.5 & 0. & 0.23 & 49.26 \end{pmatrix}$$

Room label	precision	recall	F1
1	0.9764	0.9944	0.9854
2	0.9582	0.9538	0.956
3	0.9445	0.9373	0.9409
4	0.9915	0.9853	0.9884

2.1.4 Post-Pruning Noisy Results

Classification rate	average depth	min depth	max depth
0.88	14.7	7	23

$$CM = \begin{pmatrix} 44.09 & 1.24 & 1.44 & 2.22 \\ 1.92 & 44.03 & 2.59 & 1.16 \\ 2.06 & 3.38 & 44.13 & 1.93 \\ 2.39 & 1.51 & 2.1 & 43.8 \end{pmatrix}$$

Room label	precision	recall	F1
1	0.8738	0.8998	0.8866
2	0.8777	0.886	0.8818
3	0.878	0.857	0.8673
4	0.8919	0.8795	0.8856

2.2 Analysis of the cross validation evaluation

According to the results displayed in section 2.1.1, for unpruned trees performance on the clean data set was remarkably good, with an average accuracy of around 97%. There is high precision, high recall, as well as high F1 measures uniformly, but with some variation between the rooms. For example, room label 4 has a precision, recall, and F1 measure of 0.99, 0.99, and 0.99, respectively, but room label 3 has precision, recall, and F1, of 0.94, 0.95., and 0.94.

Going class by class in the clean data set (section 2.1.1), false positives and false negatives for room 1 only show up in rooms 3 and 4. Rooms 1 and 2 are never misclassified with each other. For room 2, false positives and false negatives only appear in room 3 and nowhere else. But even though room 2 is only misclassified in room 3, the number of false negatives in room 3 is the single largest number of misclassifications in the confusion matrix for the unpruned trees on the clean data. Room 3 is the noisiest in the sense that there are false positives and false negatives in every other room label. Finally, room 4 is never misclassified in room 2 and vice-versa. These results could be due to, for example, room 2 being far away from room 4, etc.

In the noisy data set (section 2.1.2), the unpruned trees preform considerably worse, registering a decrease of around 16% in classification rate relative to the clean data set. The precision, recall, and F1 measures all decrease by similar magnitude, again with minute variation in these measures between the room labels. Unlike with the clean data, for each room there are false positives and false negatives that appear in every other room label. There are also subtle differences that emerge between the data sets in the statistics associated with the rooms. In the noisy case, in agreement with the clean data, room label 4 has the highest precision, but its recall falls from the best to third best on the noisy data. So while the change in performance between the two data sets is mostly characterized by a uniform decrement in accuracy and the classification statistics of roughly 16-19%, smaller differences in the relative performance on the individual statistics emerge as well.

For the pruned trees on the clean data (section 2.1.2), the accuracy and the notable features of the statistics, such as the fact that rooms 1 and 2, and rooms 2 and 4, are never mutually misclassified, were recapitulated. However, the performance on the noisy data (section 2.1.4) improved significantly relative to unpruned trees, with average accuracy rising from around 80% to 88%, a 10% performance increase. On the noisy data, though there are fewer false positives and false negatives for each room, there was still misclassification of each room in every other room. Thus, even though there was a quantitative improvement in misclassification, the qualitative and distributional nature of the misclassification for the noisy data on the unpruned trees persisted after pruning.

3 Visualization of the Trees and Results

For enlarged tree-visualizations see the appendix.

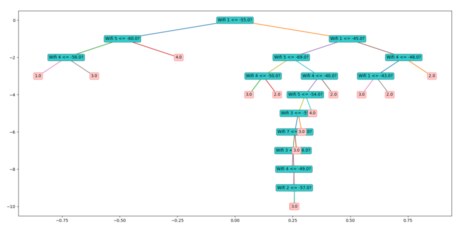

Figure 1: Pruned Tree clean

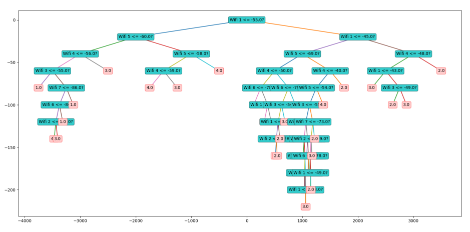

Figure 2: Unpruned Tree clean

Figure 3: Trees trained on the clean data set

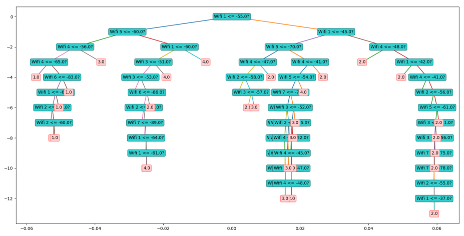

Figure 4: Pruned Tree noisy

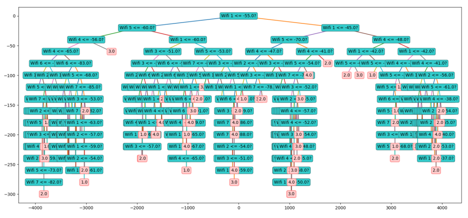

Figure 5: Unpruned Tree noisy

Figure 6: Trees trained on the noisy data set

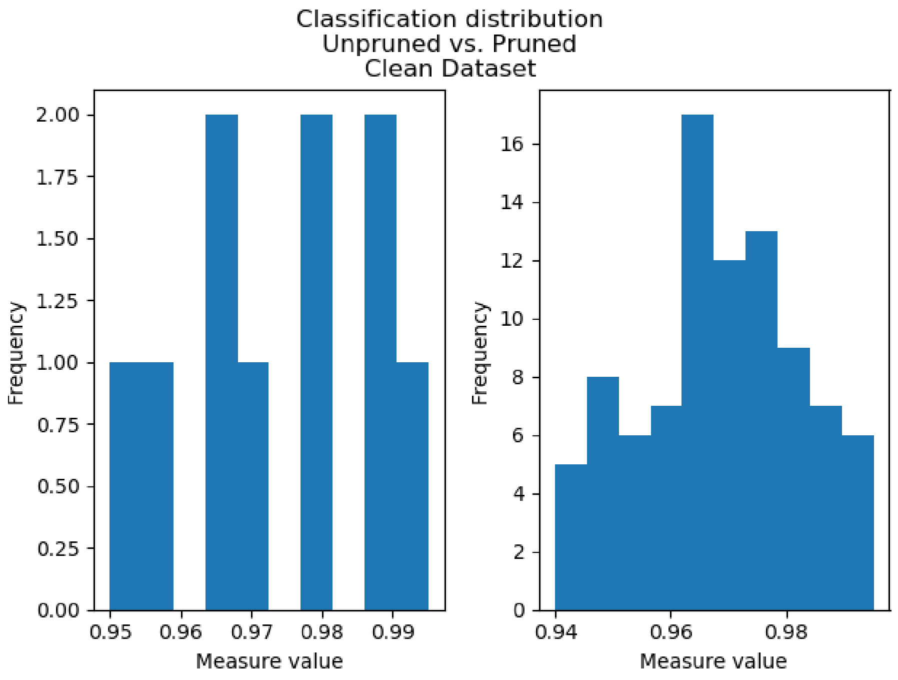

Figure 7: Distribution of classification accuracy measures for different splits during cross-validation based on clean dataset **Left:** Unpruned 10 measures. **Right:** Pruned 90 measures.

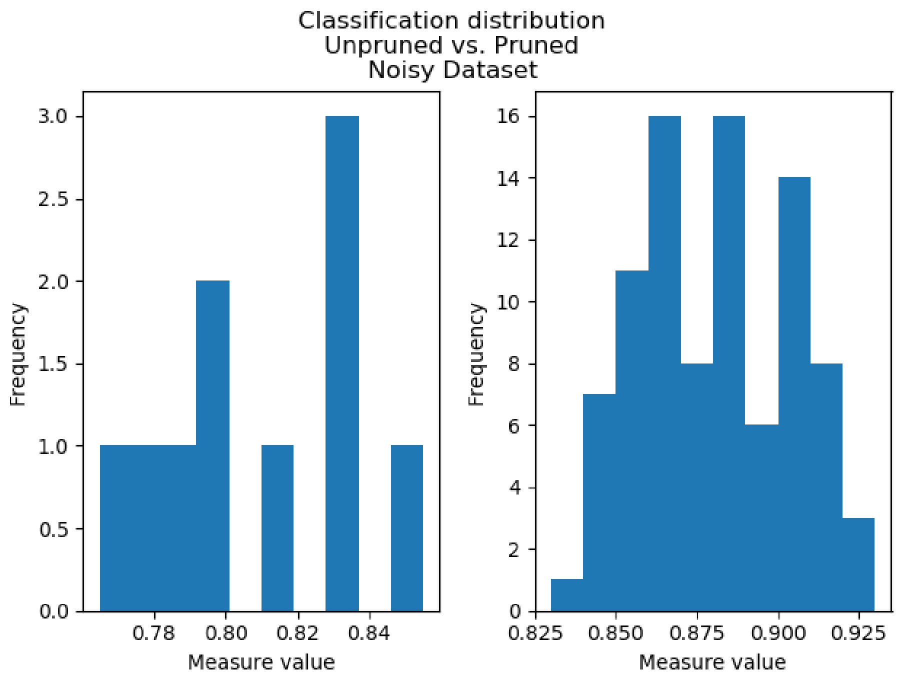

Figure 8: Distribution of classification accuracy measures for different splits during cross-validation based on noisy dataset **Left:** Unpruned 10 measures. **Right:** Pruned 90 measures.

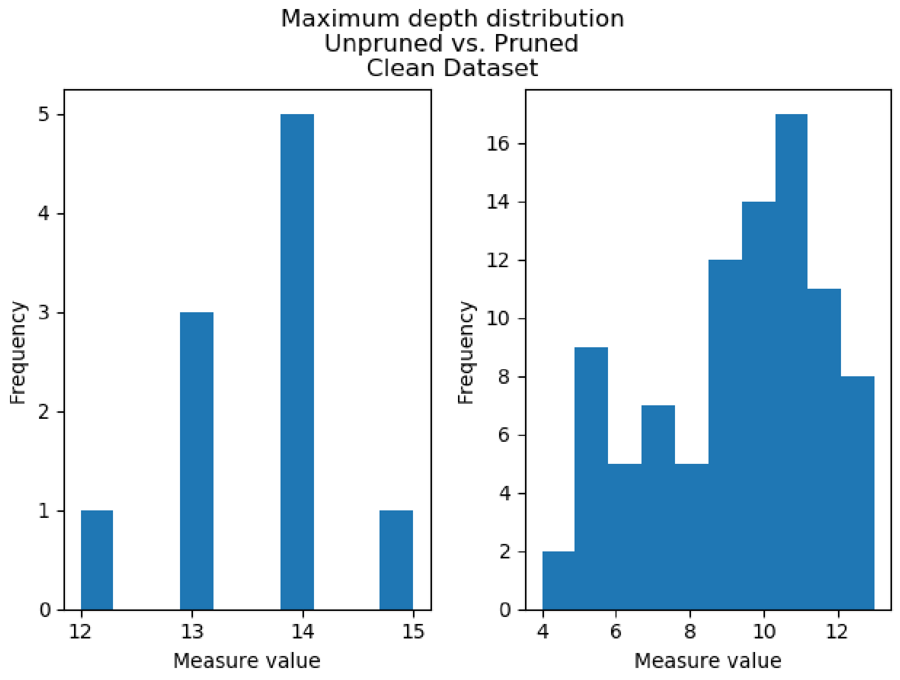

Figure 9: Distribution of maximum depth measures for different splits during cross-validation based on clean dataset **Left:** Unpruned 10 measures. **Right:** Pruned 90 measures.

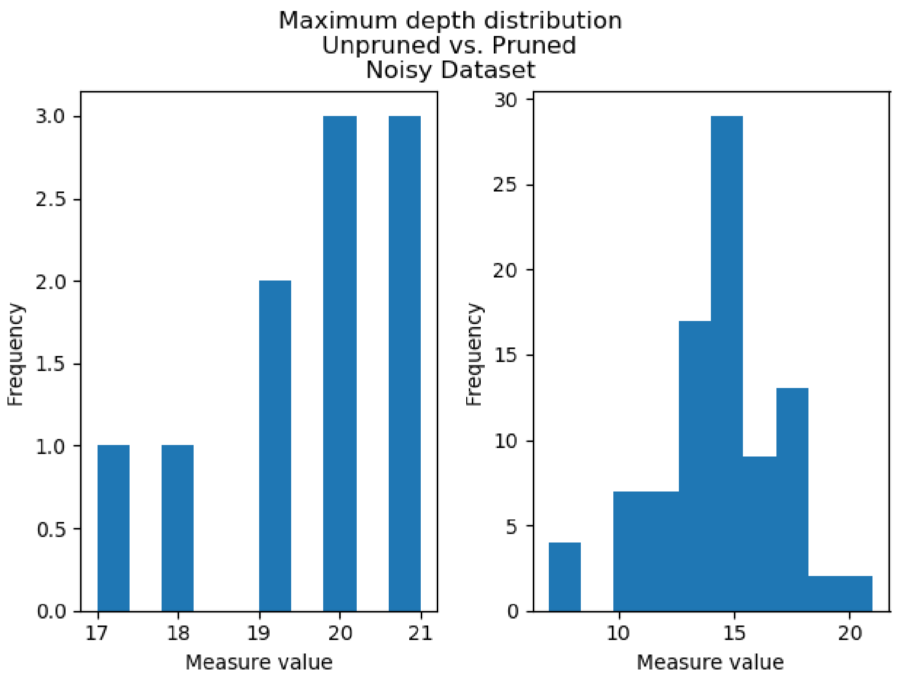

Figure 10: Distribution of maximum depth measures for different splits during cross-validation based on noisy dataset **Left:** Unpruned 10 measures. **Right:** Pruned 90 measures.

4 Answering the Questions

4.1 Q1. Noisy-Clean Datasets Question

Referring to the discussion on the cross validation results, we compared the results in sections 2.1.1 and 2.1.2, which showed a significant difference in the classification accuracy for the unpruned trees on the clean versus the noisy data. On the pruned trees, there was still a non-trivial performance difference between the data sets, but its size was attenuated. This is seen by comparing sections 2.1.3 and 2.1.4. As to why there was such a large disparity between accuracy on the clean and noisy data sets on the unpruned trees, the main reason is that because the noisy data is less linearly separable, the classification of the noisy data requires the tree to overfit, and as a result it fails to generalize well to the test data. This generalization failure is encoded in the confusion matrix and its associated statistics.
Another possible reason for the gap in performance that affects both pruned and unpruned trees is the way we split the data. For each value of each attribute, we computed the information gain, and split the data on the value that yielded the highest gain. But this has the obvious side-effect of being sensitive to noisy data, relative to choosing to split on, say, the mid point between each value. Thus, apart from pruning, the splitting criterion is a relevant explanatory factor for the decrease in classification accuracy of both sets of trees on the noisy data.
As a final point worth mentioning, we assumed that the labels would be well distributed among the data. This appears to be true. And since these data are well-

balanced, we rule out skewed data totals among the labels as an explanation for the observed performance decrease on the noisy data. If the data were unbalanced we could have performed some methods to down-sample the over-represented labels when performing cross-validation.

4.2 Q2. Pruning Question

Full implementation of the pruning algorithm is detailed in section 1.6. In short, the pruning algorithm employs a simple process of applying changes to the tree (by removing children nodes in an ordely fashion and under certain constraints) and observing how the tree 'reacts' by looking at changes of its performance on the validation set. After pruning is complete, we successfully reduced tree's complexity without compromising its performance on the dataset at hand. As a result, the tree becomes more robust and is able to generalise better.
Referring again to the data in sections 2.1, pruning had a considerable impact on classification accuracy. The pruned trees recapitulated, almost exactly, the performance on the clean data set, while dramatically improving classification performance on the noisy data. The increase in performance on the noisy data is explained by the fact that pruning prevented the over-fitting that previously occurred (as discussed in 4.1). The overall effect of the pruning on performance was to increase the robustness of the tree's classification to noisy data without compromising performance on clean data. Hence, importantly, pruning prevents over-fitting. To go deeper into the effect of pruning, see the following discussion of depth.
Another thing to mention is that pruning may not necessarily change the depth - often we observe that, post-pruning, there is a single deep branch in the tree (thus preserving the maximal depth) - e.g. see figure 1. However, even if this situation occurs most of the branches in the tree will have been cut shorter. In this way the depth statistics are by themselves an incomplete summary of the tree and can be misleading, as the average depth of the *branches* in a tree is certainly reduced by pruning, but this is not truly represented in the measure of average depth of the trees before and after pruning. To capture more information here we could have recorded the average number of nodes (or leaves) as a way of comparing the relative size and simplicity of the trees before and after pruning.

4.3 Q3. Depth Question

The depth variable exhibited dramatic variance between clean and noisy data on both pruned and unpruned trees (figures 9 and 10), and exercised considerable influence on the classification accuracy of the trees (figures 7 and 8). On the clean data, unpruned trees ranged in depth from 12 to 16 with an average of 14.1. On the noisy data, the depth of unpruned trees increased significantly, ranging from 18 to 23 layers with an average of 20. This represents an increase in average depth of around 43% relative to the clean data. On pruned trees, the clean data yielded depths between 4 and 16, with an average of 9.39, and the noisy data gave depths from 7 to 23 with an average of 14.7. Thus, pruned trees exhibited a much greater

range in depth than unpruned, whereas the increase in average depth between the data sets was reduced relative to the unpruned case.
Because the average maximal depth decreased significantly for the pruned trees relative to the unpruned trees on the same data, a lower depth - at least on average - coincides with superior classification accuracy when the data is noisy. Regarded in this light, the average depth of the pruned trees provides information about the degree of over-fitting of the unpruned trees when compared on the same data. Moreover, since the unpruned trees tend to become quite deep on noisy data, the depth variable also functions as a signal for the linear separability of the data.
An important observation is that, when the two data sets (noisy and clean) are taken together, the depth of the pruned trees ranges from 4 (on clean data, figure 9, right) to 23 (on noisy data, figure 10, right), and the depth of the unpruned trees is between 12 (on clean, figure 9, left) and 23 (noisy, figure 10, left). This dramatically increased variance in the depth of the pruned trees, relative to unpruned, shows that pruning increases the flexibility of the tree to classify the data while not over-fitting. By contrast, the relative rigidity in the depths of the unpruned trees signifies its propensity to over-fit.
Even though it is possible that the pruned trees can be as deep as the unpruned, the generation of very deep trees, even with noisy data, is strongly attenuated by pruning, as is shown by the decrease in the average depth with noisy data relative to unpruned trees on the same data. And since even in the clean data case, the depths of the pruned trees still tend to be significantly shorter than the unpruned trees, and there is no performance decrement, trees with less depth (from pruning) have more robust classification accuracy with noisy data, and do not under-perform compared to deeper trees on clean data.

5 Appendix: Tree Visualisation

These trees are trained on 80% of either the clean or noisy data set, and pruned on 10%.

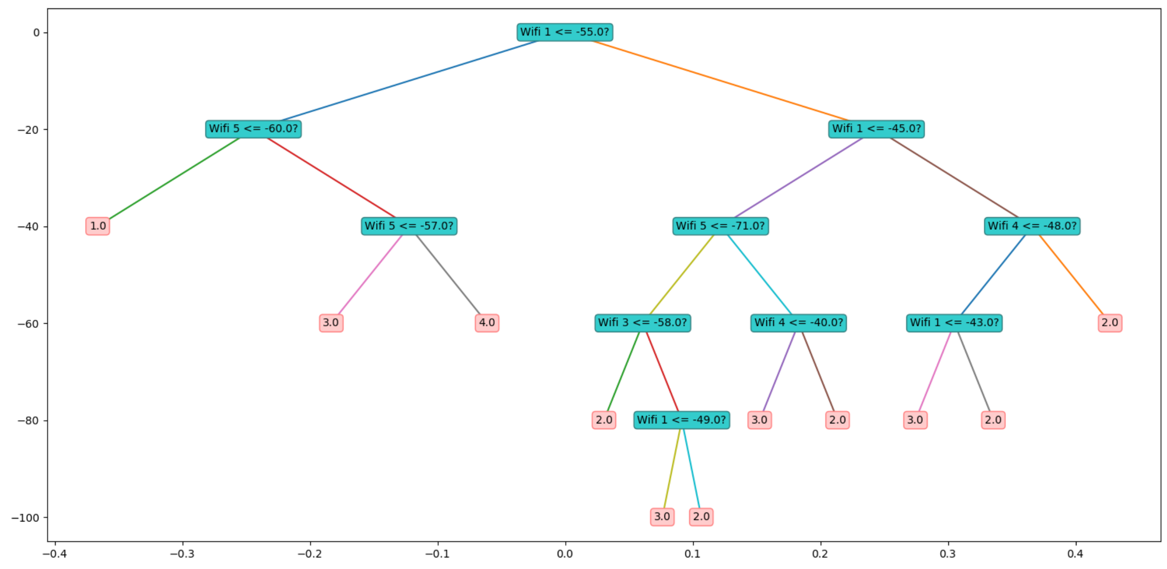

Figure 11: Pruned Tree clean

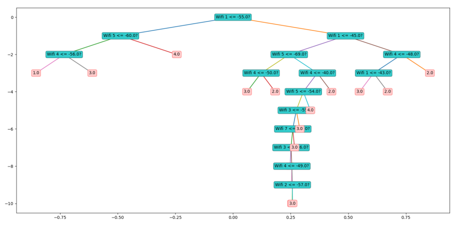

Figure 12: Pruned Tree clean

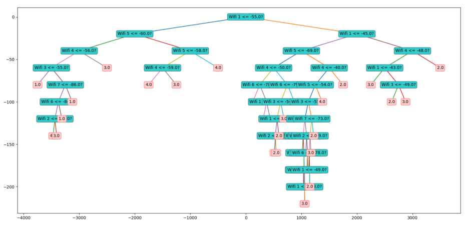

Figure 13: Unpruned Tree clean

Figure 14: Trees trained on the clean data set

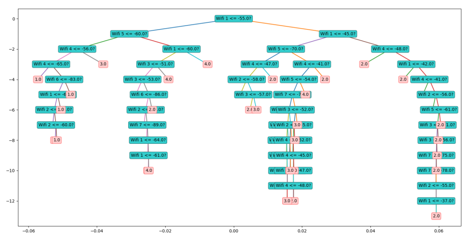

Figure 15: Pruned Tree noisy

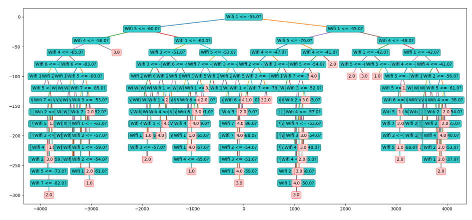

Figure 16: Unpruned Tree noisy

Figure 17: Trees trained on the noisy data set

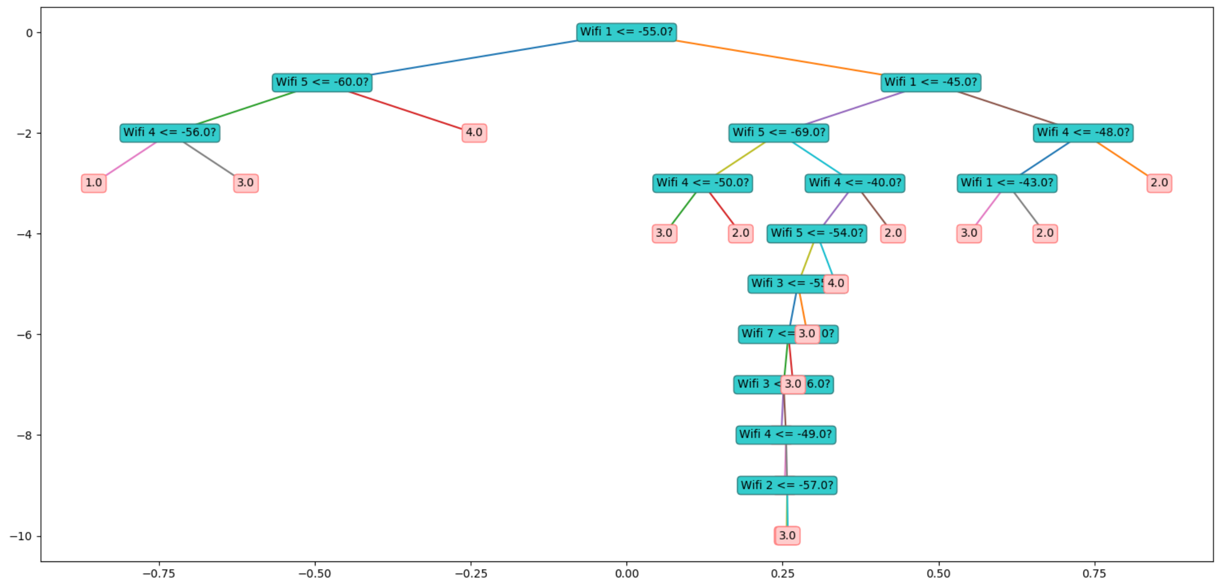

Figure 18: Clean pruned Tree

This is a complete pruned tree trained on 80% of the noisy data set and pruned on 10%:

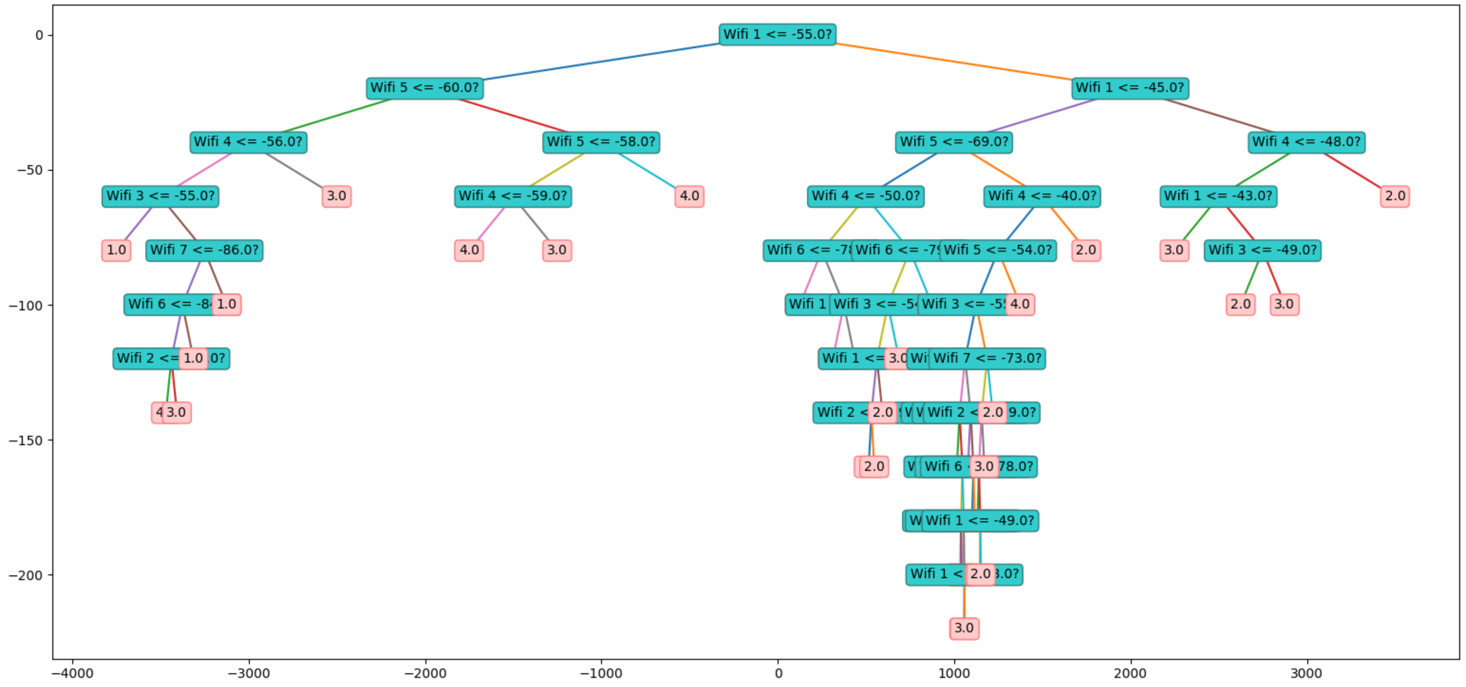

Figure 19: Clean Unpruned Tree

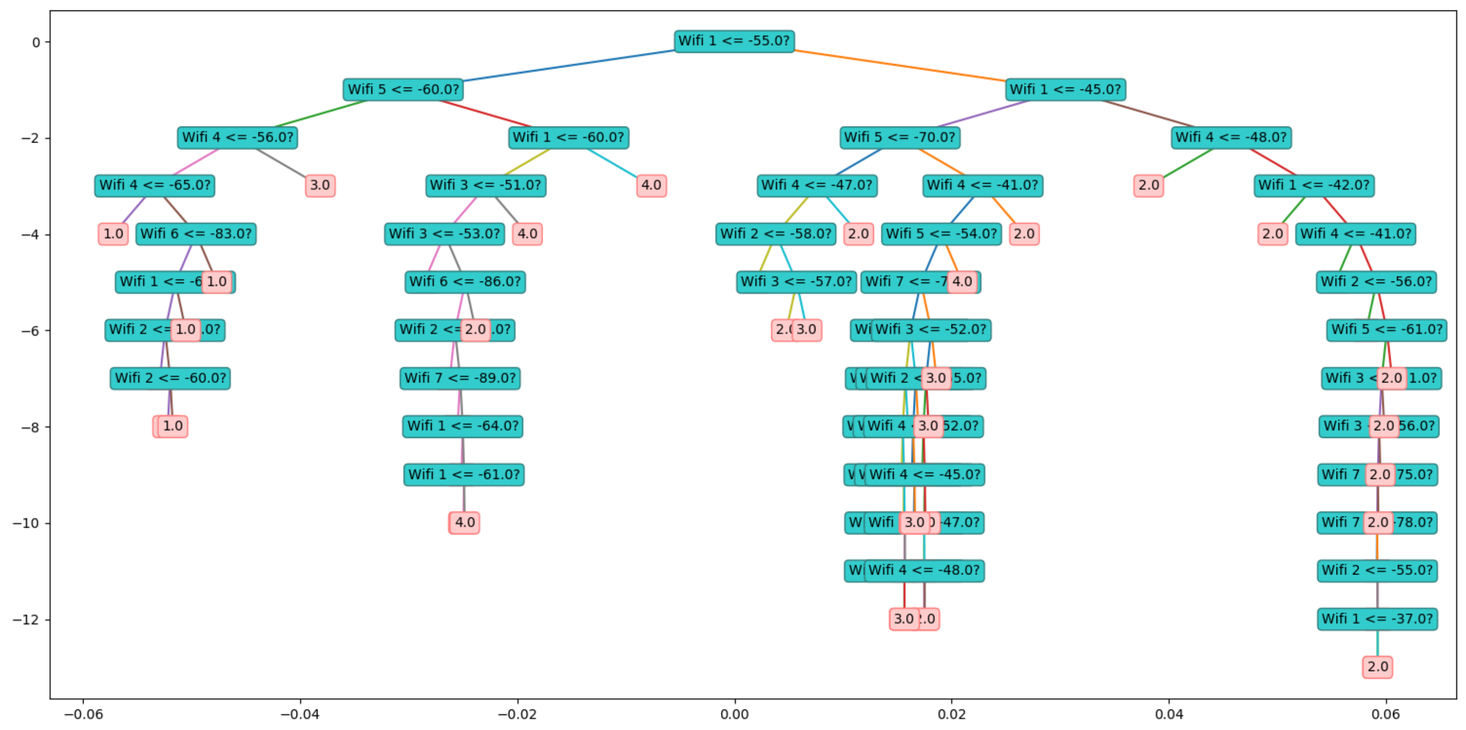

Figure 20: Noisy pruned Tree

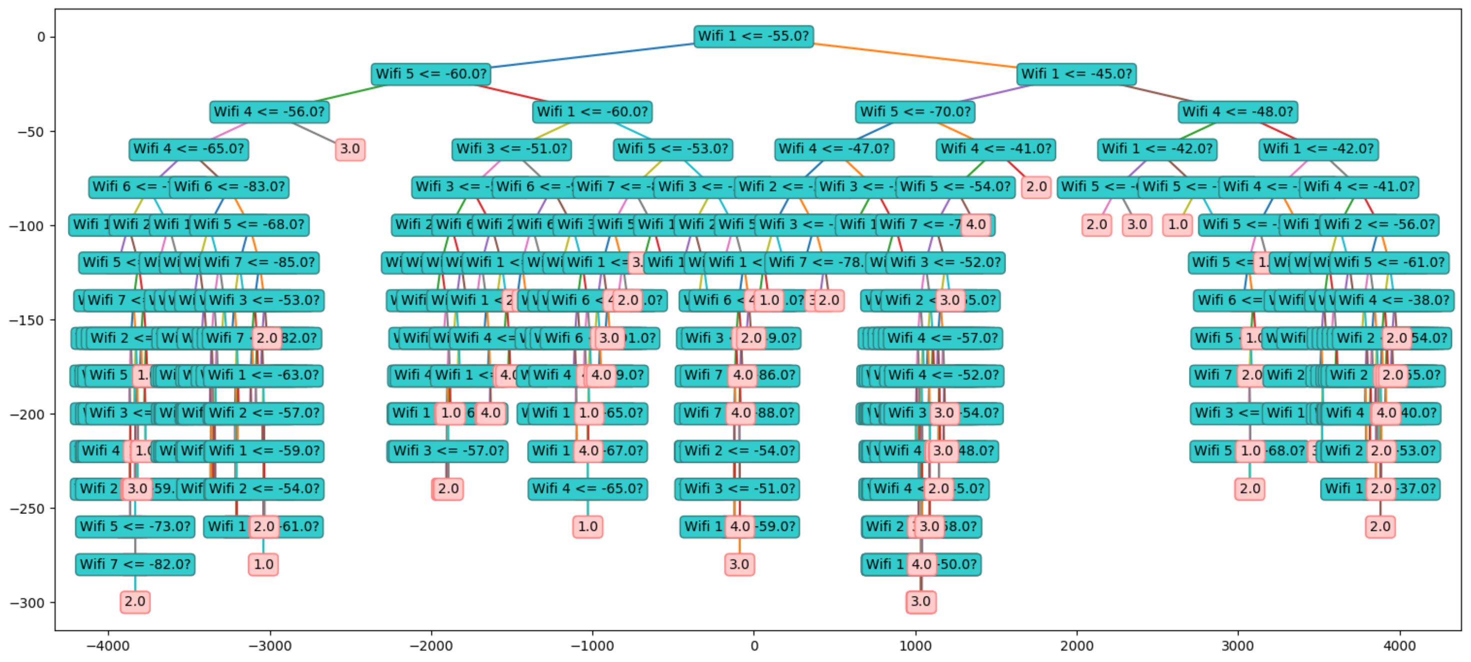

Figure 21: Noisy unpruned Tree