Imperial College London

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Introduction to Machine Learning

Author:
Jiawei Mo (CID: 01899002),
Weitong Zhang (CID: 01904870),
Yunhao Zhang (CID: 01917956),
Zhenghui Wang (CID: 01988162)

Date: November 5, 2020

1 Implementation Details

1.1 Data Processing and Cross Validation

As given two data sets, clean_dataset and noisy_dataset with the same dimension 2000×8 for both, our first purpose is to implement a data splitting function, which could apply required 10-fold cross validation. Before splitting, in the code script, we use np.random.shuffle to shuffle the fed data set. To enable 10-fold cross validation, the data set is divided into 10 sub sets, having 200×8 dimension of each. Then, choosing each one of the sub sets in turns, marked as the test set. Further, we use np.concatenate to combine the rest sub sets to form our training data set with dimension 1800×8. This preparation helps to avoid overfitting and increase potential testing accuracy. In this assignment, it is mandatory to manipulate and prepare different datasets for original decision tree and pruned decision tree.

For Step 3, the evaluation on the non-pruning decision tree, the group is using 9 folds for training and 1 fold for testing, which is clarified in the previous paragraph.

On the other hand, a different data structure is applied for Step 4, evaluation on the pruning. The dataset has been prepared by adopting a ratio, Train : Validation : Test = 8 : 1 : 1. The detailed implementation of pruning would be placed in the following section.

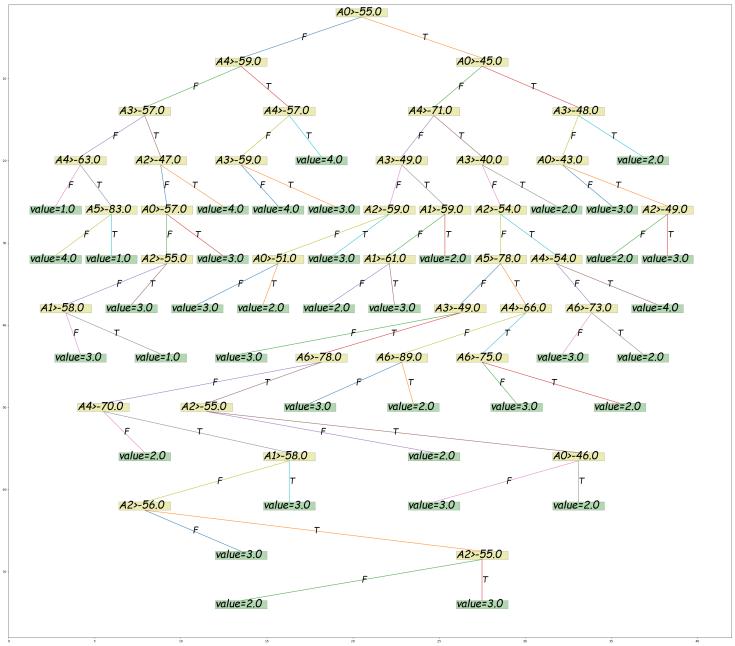
1.2 The Tree

We are using a class object to create nodes and build up the decision tree. Apparently, every node should have self.left, self.right, and self.parent, which allows node tracing in the later pruning function. No doubt that the root's parent is nothing that indicates we are located at the root node.

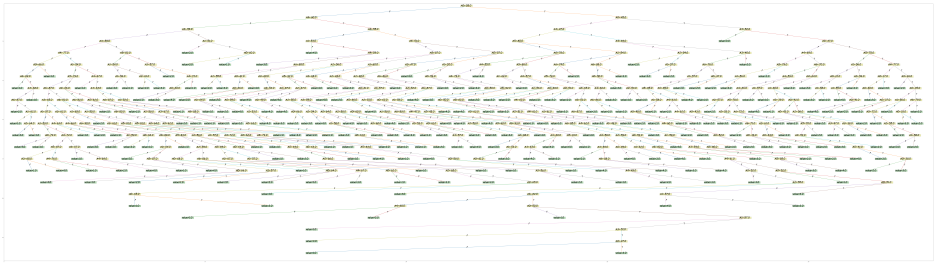
Necessarily, each node has its significant information: information gain, feature column, threshold, depth, and label. Clearly, the information gain of the current node is the maximum value compared by choosing each available feature under a certain status and compute the entropy. Features in this coursework are signals from 7 WiFi receptors. We are treating the column value (0 to 6, 7 is the room label) as the name of the feature. Additionally, the threshold is a number that we choose, which breaks a single feature into two pieces, giving maximum information gain. The depth literally discloses the depth of the current node. Finally, if the current node owns some data that has identity room label, we split the set no more and assign the label to the node. That is to say, this node becomes the leaf. We have tried using object's iterator to construct the tree and also creating a function to build up the tree. Both work well.

1.3 Split

To build our tree, we need a mechanism to split the training set. As clarified in the lecture, the attribute (feature) with the maximum information gain would be selected and used to split the tree. Specifically, the procedure takes the training_set.shape[1], which is 7, as the number of iteration. In each iteration for the features, we have to determine a threshold that tells us what value should be moved to the left or right node. To identify the threshold, values in the column are sorted at first. We can directly use each value as a threshold or we choose the mean of two adjacent values for it. Both methods are tried and it gives the same feedback. For the root, the feature 0 with threshold, -55, is chosen for split, where the mean method identifies -54.5 as the threshold. In the end, to separate the training set, for example, we assign values in the matrix such that $A0 < -55$ to the left, otherwise to the right



(a) Clean Data Decision Tree



(b) Noisy Data Decision Tree

Figure 1: Decision Tree View.

1.4 Entropy H & Information Gain

Unlike the common binary classification, this assignment has given a label with four different rooms. To calculate every entropy for the current node and its two children, we apply the equation:

$$entropy = -p_1 \log_2(p_1) - p_2 \log_2(p_2) - p_3 \log_2(p_3) - p_4 \log_2(p_4) \tag{1}$$

where the p_1 represents the probability that it is in the room 1 within the scope, and so on for p_2, p_3, and p_4. For information gain of current node, we adopt

$$info\_gain = H(current) - p_{current.left} H(current.left) - p_{current.right} H(current.right) \tag{2}$$

. There is an exception that might lead to code running termination. In the programming, $log(0)$ is a special case that we need to take care of.

1.5 Visualization of Tree

We do implement a procedure to visualize the structure of the trees for both clean and noisy sets. The common Python package, matplotlib.pyplot, helps to draw the decision tree graphs and store them. Briefly, Depth-First Search algorithm is the key to track each node and the depth value indicates the coordinates of nodes, then they are connected through left or right relationship. Figure 1 is an instance for a non-pruning decision tree built by clean_ and noisy_dataset.

2 Confusion Matrix

This part concludes commented results of the evaluation including the average confusion matrix, the average classification rate and the average precision, recall rates and F1-measure for each of the four classes for both clean and noisy datasets. Firstly, we comment on two confusion matrix and the parameters. Recall, Precision, F1 measure, Classification rate and Cross Validation will be discussed and compare the results between both datasets before and after pruning in the subsections.

2.1 Analysis of Cross Validation Without Pruning

After the implementation of 10 cross-validation, a confusion matrix can be obtained by calculating the average of ten results. Two matrics are generated for clean and noisy datasets. Since the shuffling mechanism, the output values are varying a little bit after each running. Also, the average of ten confusion matrices are computed but the sum of each matrix is 200, as $\frac{1}{10}$ of the whole 2000 samples. Table 3 and Table 4 disclose confusion matrices for both clean and noise datasets. (pred. for prediction and act. for actual)

	Room 1 Pred.	Room 2 Pred.	Room 3 Pred.	Room 4 Pred.
Room 1 Act.	49.7	0.0	0.5	0.5
Room 2 Act.	0.0	47.5	1.8	0.0
Room 3 Act.	0.2	2.5	47.4	0.2
Room 4 Act.	0.1	0.0	0.3	49.3

Table 1: Confusion Matrix - Clean Data.

	Room 1 Pred.	Room 2 Pred.	Room 3 Pred.	Room 4 Pred.
Room 1 Act.	38.1	3.1	3.0	3.9
Room 2 Act.	3.3	40.	4.6	2.7
Room 3 Act.	3.3	4.3	40.6	4.0
Room 4 Act.	4.3	2.3	3.3	39.2

Table 2: Confusion Matrix - Noise Data.

2.2 Recall

Recall represents the percentage of the results out of the total data. In this coursework, it means the percentage of the correct classification result to all the data in the four classes of samples.

2.2.1 Recall for Clean Data

$$Recall_{(Room1)} = 49.7/(49.7 + 0.0 + 0.5 + 0.5) = 98.0276\% \quad (3)$$

$$Recall_{(Room2)} = 47.5/(0.0 + 47.5 + 1.8 + 0.0) = 96.3489\% \quad (4)$$

$$Recall_{(Room3)} = 47.4/(0.2 + 2.5 + 47.4 + 0.2) = 94.2346\% \quad (5)$$

$$Recall_{(Room4)} = 49.3/(0.1 + 0.0 + 0.3 + 49.3) = 99.1952\% \quad (6)$$

Macro-average recall = 96.9516%

2.2.2 Recall for Noise Data

$$Recall_{(Room1)} = 38.1/(38.1 + 3.1 + 3.0 + 3.9) = 79.2100\% \quad (7)$$

$$Recall_{(Room2)} = 40.0/(3.3 + 40.0 + 4.6 + 2.7) = 79.0514\% \quad (8)$$

$$Recall_{(Room3)} = 40.6/(3.3 + 4.3 + 40.6 + 4.0) = 77.7778\% \quad (9)$$

$$Recall_{(Room4)} = 39.2/(4.3 + 2.3 + 3.3 + 39.2) = 79.8371\% \quad (10)$$

Macro-average recall = 78.9691%

2.3 Precision

Precision represents the percentage of the true positives out of the generated predictions. In this coursework, it means accuracy of the selected label room among the predictions.
The average $Precision_{(Clean)} = 96.95\%$ and the average $Precision_{(Noise)} = 78.95\%$

2.4 F measure

Precision and Recall are sometimes contradictory. F-Measure, as the weighted harmonic average of Precision and Recall, combines the results of them. When F is higher, it means that the experiment is ideal. We choose $\beta = 1$ to give the balanced weight of both Recall and Precision. Therefore,

$$F_{1(Clean)} = \frac{2 \cdot Precision_{(Clean)} \cdot Recall_{(Clean)}}{Precision_{(Clean)} + Recall_{(Clean)}} = 0.969508 \tag{11}$$

$$F_{1(Noise)} = \frac{2 \cdot Precision_{(Noise)} \cdot Recall_{(Noise)}}{Precision_{(Noise)} + Recall_{(Noise)}} = 0.789595 \tag{12}$$

2.5 Classification rate

Classification rate represents the percentage of samples with correct predictions to the total data samples. Generally speaking, the closer to 1, the better. The classification rate = 1 - classification err = accuracy. Thus,

$$Classification - Rate_{(Clean)} = 96.95\% \tag{13}$$

$$Classification - Rate_{(Noise)} = 78.95\% \tag{14}$$

3 Noisy-Clean Datasets Analysis

This part discusses the difference in the performance when using the clean and noisy datasets. And explain why such results exist. The differences in the overall performance and per class will also be analyzed.
There is a significant feature that the performance of clean dataset is higher than the noisy one. It can be observed from the datasets. The difference between them is also one of the main reasons for the difference in performance. As for the clean dataset, the signal features show less noise and the difference between signal values is greater. Thus, the decision tree can define the nodes with clear values. And for the noisy dataset, the data of each group is not as clear as the former, which greatly affects the determination of boundaries. It's hard to determine the nexus among similar data on some key nodes as well. There are four classes in each dataset. The influence of noise is not only reflected in the difference in overall effect, but in the comparison of each category, the performance of noisy dataset is lower than that of clean dataset.

4 Answer To the Pruning Question

This part explains how the pruning be implemented with pruning function and details. And give the comments of influence of the pruning on the accuracy of decision tree on both datasets.

4.1 Pruning Function Implementation

The entire pruning process of the decision tree aims to update nodes and recursively retain values. Thus, through this operation, the problem of overfitting in the decision tree is reduced.
• First, the verification data containing nodes, leaves and their evaluation values will be copied as input.
• Then, determine the current node situation. If the two nodes currently connected to the tree are both leaf nodes, it will try to evaluate the quality by comparing the verification accuracy with two leaf nodes.
• After the comparison, it is decided whether to delete the leaf node to improve the final verification accuracy of the tree. At the same time, the complexity is further reduced after construction
• However, if only one node is connected below the node, its value will not change, and it will continue to traverse deeper parts. Then search for the next node that can be pruned and repeat the above steps.

4.2 Analysis of Cross Validation With Pruning

Here are the data after pruning, (pred. for prediction and act. for actual)

	Room 1 Pred.	Room 2 Pred.	Room 3 Pred.	Room 4 Pred.
Room 1 Act.	49.34	0.00	0.43	0.43
Room 2 Act.	0.00	47.87	2.40	0.00
Room 3 Act.	0.29	2.13	46.80	0.51
Room 4 Act.	0.37	0.00	0.37	49.06

Table 3: Confusion Matrix - Clean Data After Pruning.

	Room 1 Pred.	Room 2 Pred.	Room 3 Pred.	Room 4 Pred.
Room 1 Act.	39.22	3.02	3.32	3.84
Room 2 Act.	3.43	40.58	3.91	2.68
Room 3 Act.	2.57	4.26	40.51	3.22
Room 4 Act.	3.78	1.84	3.76	40.06

Table 4: Confusion Matrix - Noise Data After Pruning.

The 10 cross validation for the both clean and noisy datasets results are shown in table 3 and table 4. The pruning result of clean confusion matrix is accurate for the three classes. Only room 3 is confused with other results. As for the after pruning noisy result, overall performance is not as good as that of another dataset. According to the results before pruning in table 2, the main problem is because of the design of the decision tree. Make noise excessively affect the decision-making process, leading to reduced generalization.

4.3 Data After Pruning

4.3.1 Recall for Clean Data

$$Recall_{(Room1)} = 49.34/(49.34 + 0.00 + 0.43 + 0.43) = 98.29\% \quad (15)$$

$$Recall_{(Room2)} = 47.87/(0.00 + 47.87 + 2.40 + 0.00) = 95.26\% \quad (16)$$

$$Recall_{(Room3)} = 46.80/(0.29 + 2.13 + 46.80 + 0.51) = 94.11\% \quad (17)$$

$$Recall_{(Room4)} = 49.06/(0.37 + 0.00 + 0.37 + 49.06) = 98.51\% \quad (18)$$

Macro-average recall $= 96.54\%$

4.3.2 Recall for Noise Data

$$Recall_{(Room1)} = 39.22/(39.22 + 3.02 + 3.32 + 3.84) = 79.39\% \quad (19)$$

$$Recall_{(Room2)} = 40.58/(3.43 + 40.58 + 3.91 + 2.68) = 80.20\% \quad (20)$$

$$Recall_{(Room3)} = 40.51/(2.57 + 4.26 + 40.51 + 3.22) = 80.12\% \quad (21)$$

$$Recall_{(Room4)} = 40.06/(3.78 + 1.84 + 3.76 + 40.06) = 81.03\% \quad (22)$$

Macro-average recall $= 80.19\%$

4.3.3 Precision

The average $Precision_{(Clean)} = 96.03\%$ and the average $Precision_{(Noise)} = 80.21\%$

4.3.4 F measure

$$F_{1(Clean)} = \frac{2 \cdot Precision_{(Clean)} \cdot Recall_{(Clean)}}{Precision_{(Clean)} + Recall_{(Clean)}} = 0.9628 \quad (23)$$

$$F_{1(Noise)} = \frac{2 \cdot Precision_{(Noise)} \cdot Recall_{(Noise)}}{Precision_{(Noise)} + Recall_{(Noise)}} = 0.8103 \quad (24)$$

4.3.5 Classification rate

$$Classification - Rate_{(Clean)} = 97.35\% \quad (25)$$

$$Classification - Rate_{(Noise)} = 80.55\% \quad (26)$$

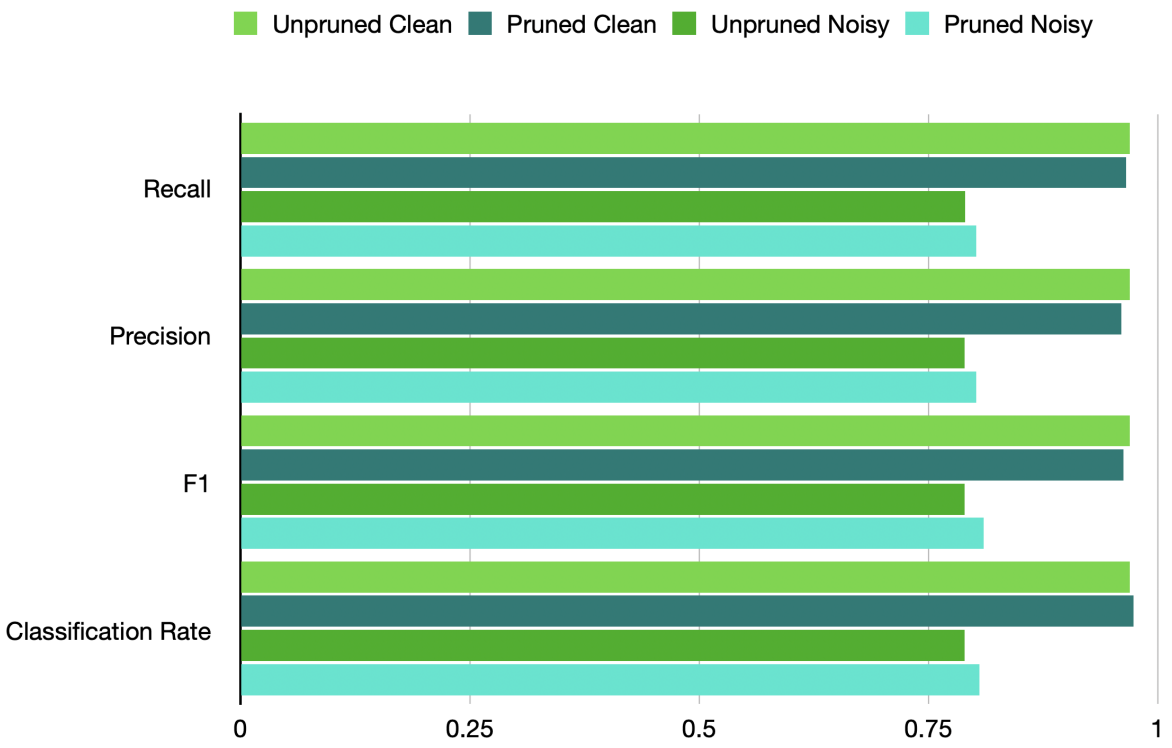


Figure 2: Evaluate data before and after pruning for both clean and noisy datasets.

4.4 Influence of Pruning

The pruning operation of the decision tree has different degrees of impact on the two datasets. Part of the reason is that it is affected by the characteristics of the datasets themselves. Besides that, for clean dataset, the improvement obtained before pruning has had a significant effect, and the improvement after pruning has become extremely limited. For another dataset, pruning has a more intuitive improvement effect.

It can be seen from figure 2 that after pruning, the improvement of the former classification rate is 0.4%, while the improvement of the noisy dataset is 1.6%. After pruning, the performance of the decision tree is further improved. After removing unimportant feature branches, it not only reduces the complexity of the decision tree, but also improves its predictive ability by reducing degree of overfitting. However, judging from the recall precision and F1 measure indicators, the pruning operation does not bring improvements in all aspects, but while improving accuracy, it also increases the degree of confusion with other room labels. Therefore, while pruning judges whether a node contributes to the improvement of the overall index, it also needs to increase the judgment of the impact between room classes. This will involve the node stopping when the impurity is still high. At this time, due to insufficient foresight of the future and insufficient growth, the error rate of inter-class classification is higher.

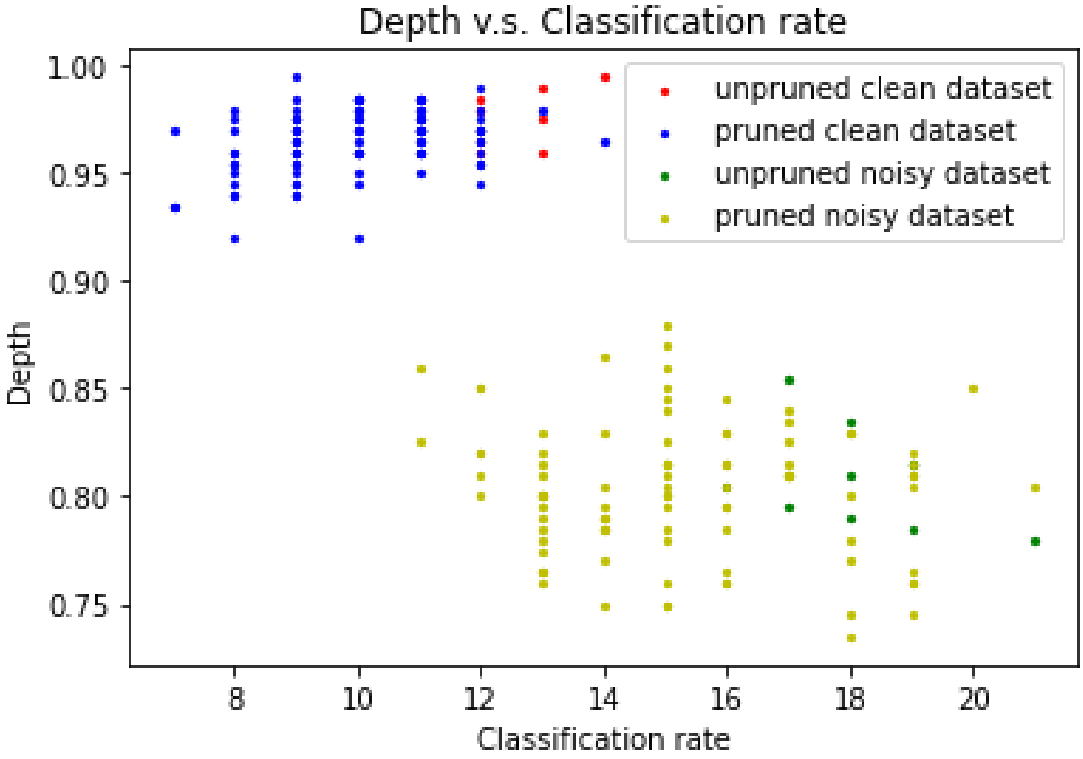


Figure 3: The distribution of maximum depth and classification rate of decision tree in different datasets

5 Answer To the Depth Question

This part gives the comments on the maximal depth of the tree that we generated for both datasets and before and after pruning. And the relationship between maximal depth and prediction accuracy will be analyzed in the last section.

5.1 Maximum Depth

Based on the instructor's reply on Piazza, the average maximum depth for the clean dataset is based on ten maximum depths. The average maximum depth for the noise one is based on 90 maximum depths.

The average maximal depth of clean dataset before pruning is 12.3, and 10 for after pruning. The average maximal depth of clean dataset before pruning is 18.2, and 15.3 for after pruning. The maximum depth of clean dataset & noise dataset had a clear improvement after pruning.

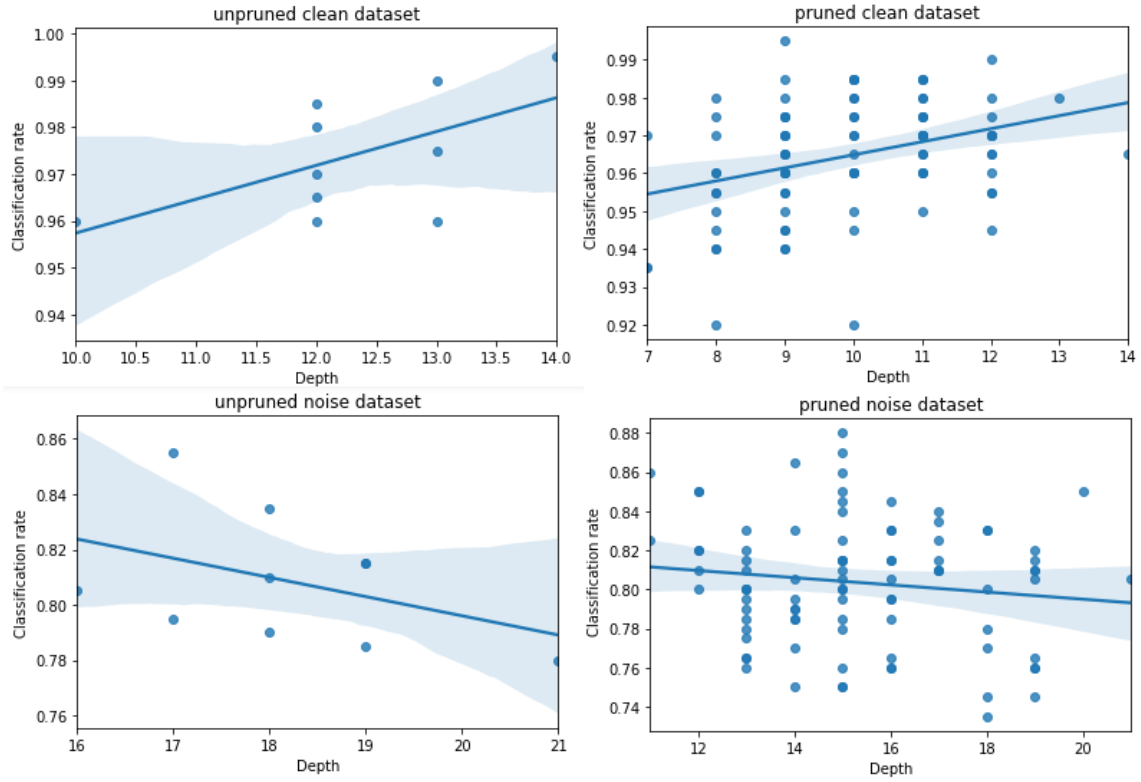


Figure 4: The trendline between depth and classification rate in different datasets

5.2 Relationship between maximal Depth and prediction

Figure 3 shows the maximum depth of each decision tree we build with the corresponding classification rate for different decision tree.As we used the strategy of 90 folds in the pruned dataset and ten folds in the unpruned dataset, the correct prediction rate of the pruned dataset is not increased as usual. The average classification rate of the unpruned clean dataset is 0.974, and the rate is decreased to 0.965 after pruning.

In figure 4, it showed the trendlines between maximum depth and the classification rate. It clearly is shown that the classification rate increases with a higher maximum depth. The reason for that is because of the clean dataset don't have any unseen data. Even the decision tree is overfitted, it could still provide pretty good classification rate.

The average classification rate of unpruned noise dataset is 0.808, and the rate is decreased to 0.804 after pruning. However, the relationship between the classification rate and maximum depth of noise dataset is in the opposite direction of the clean dataset. The correct classification rate of noise data is decreased with the higher maximum depth of the decision tree. It is because of the decision is overfitted, the predication result will become worse when there are some unseen data.

In summary, pruning is a necessary process for building a decision tree. It can effectively avoid the overfitting of the decision tree. For an overfitted decision tree, the higher the maximum depth, the lower performance of handling unseen data.