MACHINE LEARNING (CSCI 8950): HOMEWORK-1

Accompanying GitHub Repository:
https://github.com/Chintan2108/CSCI-8950-HW/tree/main/HW-1

FEBRUARY 7, 2022

CHINTAN B. MANIYAR
chintanmaniyar@uga.edu

NOTE: Python's *scikit-learn* package was used to implement the solution to this problem set.

Solution to 1(a):

The dataset used for this question is the *PlayTennis* dataset. The Decision Tree was learnt on the entire dataset, all 14 instances. Since *scikit*-learn does not work with categorical variables, the dataset in question was encoded using *Label* Encoder, so that the variables such as 'Outlook', 'Temp', 'Humidity' can be converted from text strings to encoded numerical values (Figure 1). Hyperparameters such as max-depth, entropy were not modified from their default settings.

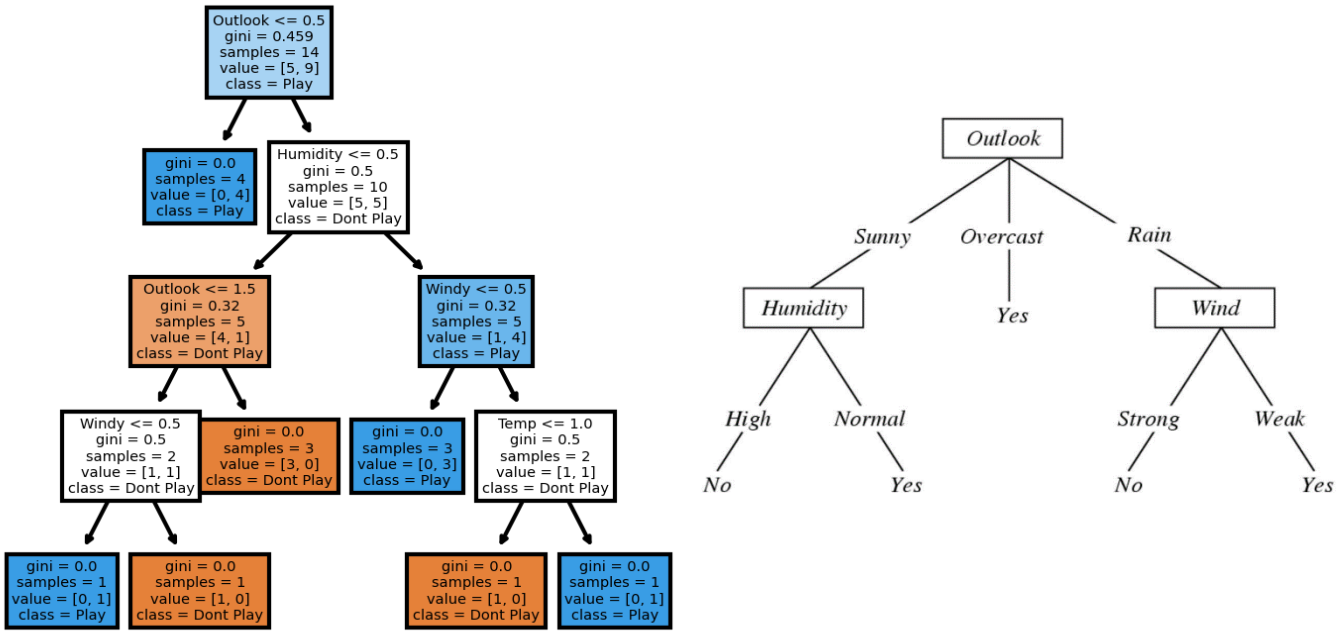
	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

Label Encoding

	outlook	temp	humidity	windy	play
0	2	1	0	0	0
1	2	1	0	1	0
2	0	1	0	0	1
3	1	2	0	0	1
4	1	0	1	0	1
5	1	0	1	1	0
6	0	0	1	1	1
7	2	2	0	0	0
8	2	0	1	0	1
9	1	2	1	0	1
10	2	2	1	1	1
11	0	2	0	1	1
12	0	1	1	0	1
13	1	2	0	1	0

Figure 1: Snapshot of the Play Tennis dataset

Figure 2 shows the tree obtained after learning on the entire dataset. Max-depth was not restricted and *gini* was used as the information gain measure.



(a) (b)

Figure 2: (a) Decision tree as obtained from *scikit-learn* after learning the entire *PlayTennis* dataset. (b) Decision tree as described in the class notes

I get a different decision tree than the one described in class notes. *Scikit-learn* also uses the same information gain measure, *gini*, however, the tree in class notes is not strictly binary while the one obtained from *scikit-learn* is strictly binary. The branching follows the same fashion in both cases, root node is 'Outlook' variable, and the tree traversal is further split by 'Humidity' and 'Wind' variables respectively in both cases. Hence, the only difference between the two trees is that one is strictly binary while the other is not. This is because *scikit-learn* uses CART for decision trees which can only deal with numerical data, and when we have numerical data, it is typically difficult to do exhaustive branching as there can be infinite possibilities (as compared to categorical variables, which would have a finite number of possibilities). Thus, in *scikit-learn* the splitting threshold is set to 2 and we get a strictly binary tree.

Solution to 1(b):

Leave one out cross-validation technique resulted into 14 different trees, since there were 14 samples in the *PlayTennis* dataset. Every sample was included in the testing set once and, in the training set, 13 times through the whole process. *Accuracy* was used as the error metric. The mean accuracy from all 14 runs was 71.43%, indicating that in 10 cases the tree predicted corrected for the test sample while in 4 cases the prediction was incorrect. *Scikit-learn* does not give one best tree, unlike *Weka*.

Solution to 2(a):

For this question, I used the *WisconsinBreastCancerDetection* dataset. It contains 9 predictor variables and one target variable which has two classes namely 'Benign' and 'Malignant', denoted by 2 and 4 respectively. The data was already encoded, however, there were some missing values in one of the predictor variables. Data cleaning was done by dropping the samples (as it was a categorical variable) with any missing values which reduced the sample size from 699 to 683. Figure 3 shows the head of the dataset with 5 samples.

Clump_Thickness	Cell_Size_Uni	Cell_Shape_Uni	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoi	Mitoses	Class
5	1	1	1	2	1	3	1	1	2
5	4	4	5	7	10	3	2	1	2
3	1	1	1	2	2	3	1	1	2
6	8	8	1	3	4	3	7	1	2
4	1	1	3	2	1	3	1	1	2

Figure 3: Snapshot of the Wisconsin Breast Cancer Dataset

Figure 4 shows the decision tree obtained by learning on the full dataset. Accuracy on the training set was 100%, which means *scikit-learn* allowed the tree to grow exhaustively. The learning time was 8.86ms. (Figure 4 is at the end of the document).

Solution to 2(b):

Learning the tree with 10-fold cross-validation (training: 615 samples, testing: 68 samples) without pruning, the best model fetched a train accuracy of 100% (all samples classified correctly) and a test accuracy of 94.16% (out of 68 samples, 64 classified correctly and 4 classified incorrectly). Learning time was 2.72s.

Solution to 2(c):

For this question, I used Cost Complexity Pruning (CCP). This is most likely a form of pre-pruning. While the decision tree is being created, the cost and complexity of creating each new node is calculated in terms of generalization, and based on this cost, the depth is decided (when to stop growing the tree). In *scikit-learn*, a decision tree can be CCP-pruned by setting the parameter called 'ccp-alpha', which ranges from [0,1]. The higher the value of this parameter, more the tree will be pruned. In other words, if 'ccp-alpha' is set to 0, the tree will overfit. As the value of 'ccp-alpha' increases, more of the decision tree is pruned and hence it will generalize better. Figure 5 shows the variation of train and test accuracy for 'ccp-alpha' in the range [0, 0.35]. It can be noted that as 'ccp-alpha' increases, the both train and test accuracies decrease but the difference between then starts to decrease and becomes 0 at roughly 'ccp-alpha'=0.17, indicating generalization.

Learning the tree with 10-fold cross-validation with pruning allowed fetched a train accuracy of 100% and the test accuracy increased to 96.05% (out of 68 samples, 65 classified correctly). Learning

time was 428ms. 'ccp-alpha' was set to 0.015. Table 1 shows the performance comparison between pruning restricted and pruning allowed.

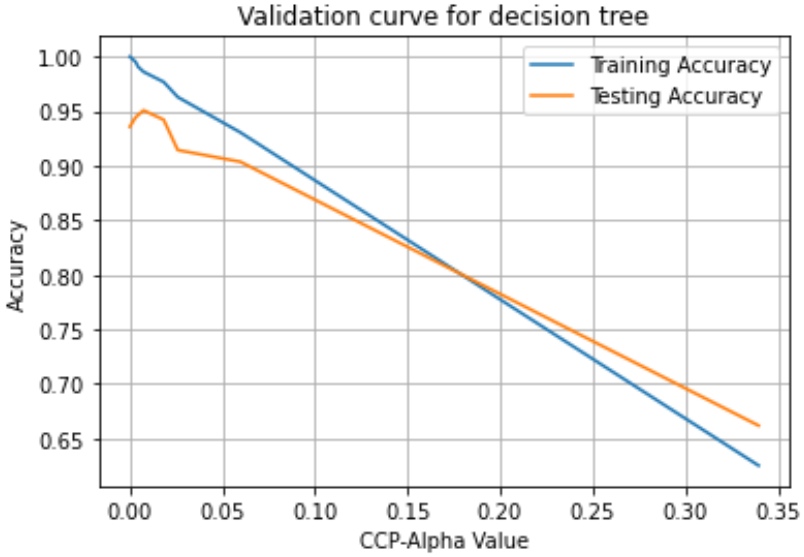


Figure 5: Effect of 'ccp-alpha' on tree learning

Table 1: Effect of pruning while learning a decision tree with 10-fold cross-validation

	Without Pruning	**With Pruning**
Train Accuracy	100%	100%
Test Accuracy	94.16%	96.05%

Solution to 2(d):

When CCP pruning was applied, the test accuracy improved by roughly ~2%. In *scikit-learn*, the default value of 'ccp-alpha' is set to 0. However, to allow pruning along with cross validation, it was changed to 0.015, which means that the tree was pruned, and the tree growth was not exhaustive (it was stopped early). Interestingly, this did not affect the training accuracy. Ideally, one would expect the training accuracy to slightly decrease, but in this case, training accuracy remained the same even after pruning most likely because of the small value of 'ccp-alpha'. However, this small amount of pruning increased testing accuracy which resulted into one more sample being classified correctly, indicating generalization.

References

1. Scikit-learn documentation (https://scikit-learn.org/0.21/documentation.html)

APPENDIX

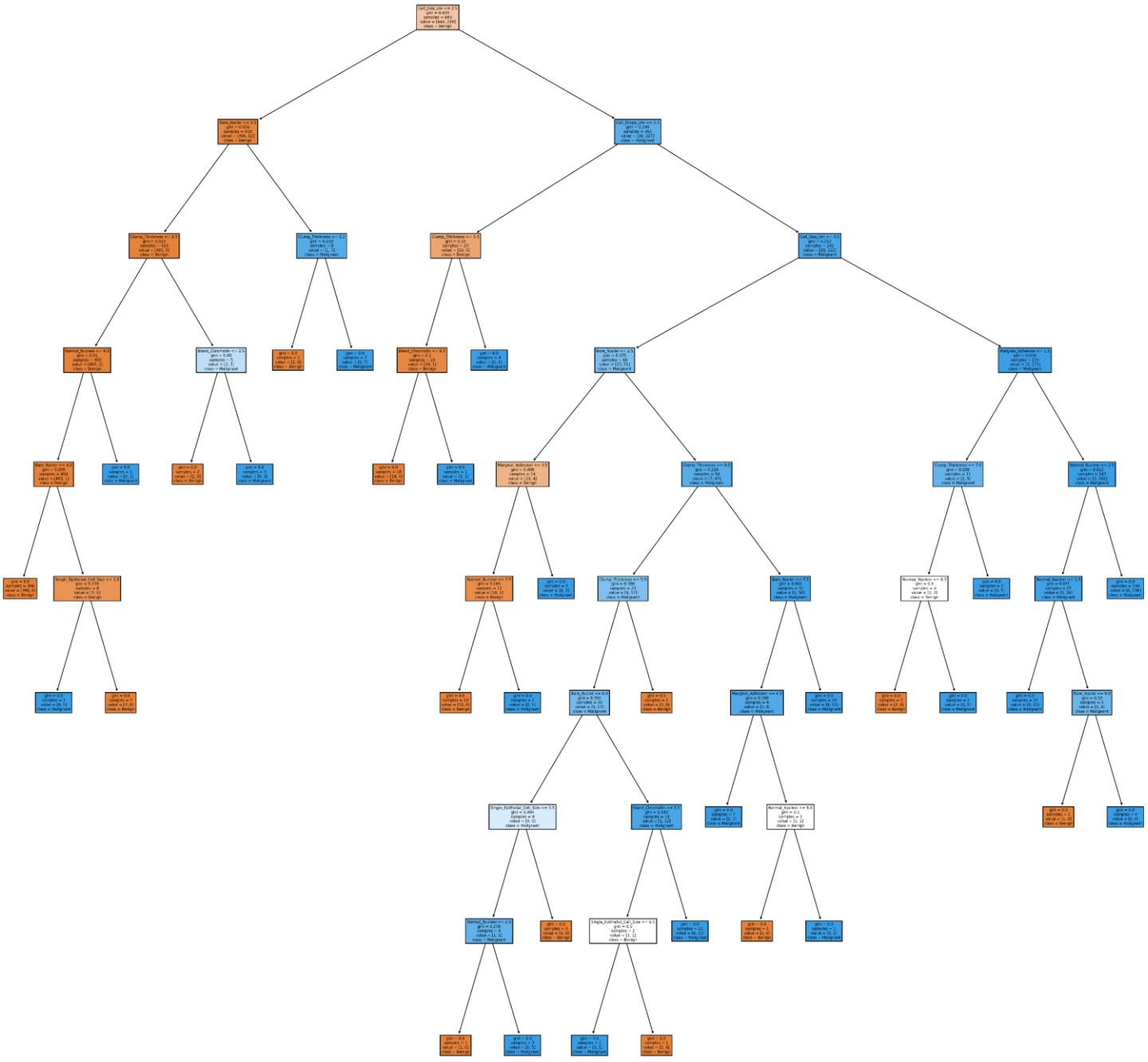


Figure 4: Decision tree as obtained from *scikit-learn* after learning the entire *WisconsinBreastCancerDetection* dataset