CS-550 - Spring 2023 HW 1: Decision Trees

Salih Deniz Uzel (deniz.uzel@bilkent.edu.tr)
22201382

I. Part 1

To train decision tree I used "scikit-learn" python library for the DecisionTree Classifier. For the for the splitting criterion I used "gini". I have splitted the training dataset into training and validation dataset with a ratio of $90/10$.

A. Draw the decision tree

In order to pick the best parameters I used random search. The parameters space is given below.

criterion = ['gini', 'entropy', 'log_loss']
splitter = ['best', 'random']
max_depth = [None, 5, 10, 15, 20, 25, 30, 35, 40, 45]
min_samples_split = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20]

I trained the tree on the training dataset with randomly selected settings and I evaluated the performance on the validation set due to available computation power. I run this process 1000 times and picked the parameters that gives the best validation set accuracy. The best combination calculated as:

criterion = 'gini'
splitter = 'best'
max_depth = 15
min_samples_split = 8

The tree graph is located in the homework file with the name of "part1-tree.svg".

B. Evaluation of the model

I trained the decision tree with selected parameters on train dataset. I evaluated the model with train and test sets.

TABLE I: Training, and Test set class based accuracies

	Accuracy	
Class No	Training	Test
1	0.9762	0.8767
2	1	1
3	0.9997	0.9946

C. With Pruning

I used "minimal cost-complexity pruning". This method removes the least efficient nodes depending on value called ccp_alpha. I applied pruning by increasing the ccp_alpha value until only 1 node left in the tree. Model seems to less overfit the train dataset where "Max Depth" is 3 in the table. For this reason pruning parameter ccp_alpha is picked 0.004562. $c.1, c.2, c.3$ columns are the class accuracies. And the class accuracies can be seen in the first row where $ccp\_alpha = 0.000000$.

TABLE II: Confusion Matrix of Train dataset.

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	82	0	2
	Class 2	0	172	0
	Class 3	0	1	3137

TABLE III: Confusion Matrix of Test dataset.

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	64	0	9
	Class 2	0	177	0
	Class 3	6	11	3161

TABLE IV: Pruning Train dataset results. $c.1, c.2, c.3$ columns are the class accuracies.

Number of Nodes	Max Depth	α	c.1	c.2	c.3
25	**8**	**0.000000**	**1.00**	**1.00**	**1.00**
21	6	0.000293	1.00	1.00	1.00
17	6	0.000575	1.00	1.00	1.00
15	5	0.000579	1.00	1.00	1.00
13	5	0.000707	0.98	1.00	1.00
11	5	0.000923	1.00	1.00	1.00
9	4	0.003875	1.00	1.00	1.00
7	**3**	**0.004562**	**1.00**	**1.00**	**0.99**
5	2	0.019417	1.00	1.00	0.98
3	1	0.027828	0.00	1.00	0.98
1	0	0.082357	0.00	0.00	1.00

TABLE V: No-Pruning - Training, and Test set class based accuracies

	Accuracy	
Class No	Training	Test
1	1	0.8767
2	1	0.9944
3	1	0.9947

It can be seen on the Table VI that the preventing over-fitting with pruning increased the class accuracies on the Test set. It generalized the model better for the given sample distribution.

TABLE VI: With-Pruning - Training, and Test set class based accuracies. alpha = 0.004562

	Accuracy	
Class No	Training	Test
1	1	1
2	1	1
3	0.9927	0.9871

D. Normalization

z-score normalization applied to each column, which corresponds to value of a each feature, for all samples. Normalization didn't have significant effect on the train and test datasets.

TABLE VII: Without Normalization - Training, and Test set class based accuracies.

	Accuracy	
Class No	Training	Test
1	1	0.8767
2	1	0.9944
3	1	0.9947

TABLE VIII: With Normalization - Training, and Test set class based accuracies.

	Accuracy	
Class No	Training	Test
1	1	0.8630
2	1	0.9944
3	1	0.9934

E. Balanced Dataset

Balancing the sample distribution of the dataset put more emphasis on the least represented classes. Model learned the class 1 better and probably over-fitted due to small amount of samples in the balanced training set. Class 2 and Class3 accuracies dropped about 1%. The result may or may not be the desired outcome. Desired outcome, may vary depending on which class is preferred for high accuracy in real life data.

TABLE IX: Without Balancing - Training, and Test set class based accuracies.

	Accuracy	
Class No	Training	Test
1	1	0.8767
2	1	0.9944
3	1	0.9947

TABLE X: With Balancing - Training, and Test set class based accuracies.

	Accuracy	
Class No	Training	Test
1	1	1.0
2	1	0.9774
3	1	0.9846

II. PART 2

For the splitting criteria I used "$gini\ impurity$". "$gini$" method in the implementation calculates the gain of the given split conditions. Splitting feature is decided based on this metric. However, picking the feature and the threshold implemented with the following strategy. The "$calculate\_best\_split$" method searches for the best feature and the best threshold. Superficially, this is achieved by calculating the gain for every feature. Then the feature with the maximum gain is selected. During the gain calculation for each feature a threshold value is selected. This selection is also determined by the different algorithm for float feature values. Threshold selection for 0/1 values selected as 0.5 by default. Best threshold selection is achieved as follows: For the selected feature, the feature values are sorted and squentially each value selected for the threshold. The gain is calculated for each value of these feature values. The value that yields the maximum gain selected for the threshold value and overall feature gain is calculated with this threshold value. The sorted feature values yiels a collection of calculated gain values which is a "$unimodal$" function. Therefore the algorithm implemented as $peak\ \ finding$ with $binary\ \ search$. The searching process of maximum gain for a feature has $\mathcal{O}(\log n)$ time complexity.

Algorithm 1 Calculate Best Split

Require: data, labels

```
maxGain = -Infinite
initialize a list named leftSamples to store splitted samples for the left child
initialize a list named rightSamples to store splitted samples for the right child
maxFeatureIndex = -1
maxFeatureThreshold;
for featureIndex ← 0 to features.length − 1 do
    initialize featureCol that stores the sample value for the given feature
    tmpleftSamples, tmprightSamples, threshold = Partition(featureCol, labels)
    parentInfo, leftInfo, reightInfo, gain = Gini(tmpleftSamples, tmprightSamples)
    if costOfFeature is True then
        cost = featureCosts[featureIndex]
        gain = gain^2/cost
    end if
    if maxGain <= gain then
        maxGain = gain
        maxFeatureIndex = featureIndex
        maxFeatureThreshold = threshold
        leftSamples = tmpLeftSamples
        rightSamples = tmpRightSamples
    end if
end for
return maxFeatureIndex, maxFeatureThreshold, parentInfo, leftSamples, rightSamples
```

The classifier splits until the leaf nodes. The maximum depth and minimum samples split determined by user for pre pruning. Classifer assign classes to leaves. To determine the class, the class with the maximum number of samples in the leaf is calculated. If the number of samples of each class is equal, the first class value in the array is assigned to leaf.

A. Draw the decision tree

In order to pick the best parameters I used random search. The criterion is default 'gini' and splitter is 'best' due to my DecisionTreeClasssifier imlementation. The parameters space is given below.

max_depth = [None, 1, 2, 3, .., 28, 29]
min_samples_split = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20]

I trained the tree on the training dataset with randomly selected settings and I evaluated the performance on the validation set. I run this process 1000 times and picked the parameters that gives the best validation set accuracy.

The training set is splitted to training and validation set with ratio of 80/20. The best combination calculated as:

max_depth = 7
min_samples_split = 3
validation accuracy = 0.9464133112150002

As the size of the validation has increased, it is more likely to include less common samples. Therefore, the accuracy of class1 in the test set has also increased. And over-fitting is decreased.

The tree graph is located in the homework file with the name of "part2-pruned-tree.svg".

B. Evaluation of the model

TABLE XI: Part2 - Best Configuration. Training, and Test set class based accuracies.

	Accuracy	
Class No	Training	Test
1	1	0.9589
2	1	0.9944
3	0.9996	0.9918

TABLE XII: Confusion Matrix of Train dataset.

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	74	0	0
	Class 2	0	153	0
	Class 3	0	1	2789

In the part1 and part2 'gini' criterion was used. The selecting best feature implementation used in part2, "peak finding" and "best gain", are the same methods sklearn library uses for decision tree implementation.

TABLE XIII: Confusion Matrix of Test dataset.

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	70	0	3
	Class 2	0	176	1
	Class 3	10	16	3152

III. Part 3

For the new criterion I selected the modified version of the gain calculation. The "gain" value, which is calculated by using the best split, squared in order to emphasize the gain and divided by the cost value to penalize higher costs. The equation derived from (Tan, 1993) is given below.

$$criterion = \frac{Gain(S,F)^2}{cost(\text{feature})} \tag{1}$$

A. Draw the Decision Tree

Best parameters are calculated with the new criterion with 200 random initialization. Training/Validation split ratio is 80/20 same with Part 2. Parameter space is as follow:

max_depth = [None, 1, 2, 3, .., 28, 29]
min_samples_split = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20]

Best parameters are as follows:

max_depth = 3
min_samples_split = 2
val_acc = 0.998567335243553

The tree graph is located in the homework file with the name of "part3-decision-tree.svg".

B. Evaluation of the model

TABLE XIV: Part3 - Best Configuration. Training, and Test set class based accuracies.

	Accuracy	
Class No	Training	Test
1	0.9595	0.9726
2	1	1
3	0.9896	0.9852

TABLE XV: Confusion Matrix of Train dataset.

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	71	3	0
	Class 2	0	153	0
	Class 3	4	25	2761

TABLE XVI: Confusion Matrix of Test dataset.

		Predicted Class		
		Class 1	Class 2	Class 3
Actual Class	Class 1	71	2	0
	Class 2	0	177	0
	Class 3	11	36	3131

C. *"the average cost of classifying an instance"*

Class 1 = 74.61999999999992
Class 2 = 51.38949152542369
Class 3 = 50.10292007551934

IV. Additional, Cost Calculation Proof

In order the prove cost calculation of an instance a balanced data set that has 74 samples for each class was created. Minumum sample split value was set to 85 for demonstration. For cost calculation, test set containing only 1 sample from each class is created.

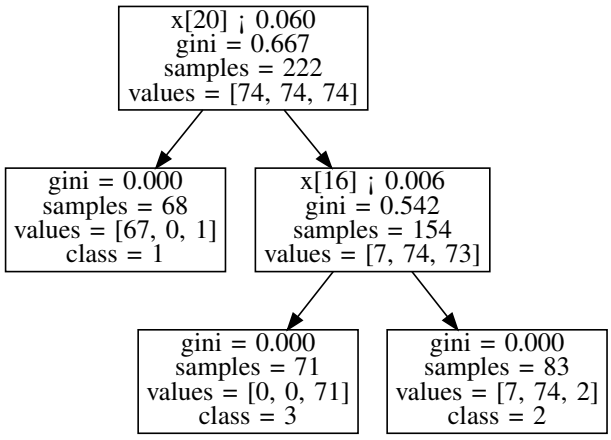

Fig. 1: A small decision tree trained on balanced data set to prove cost calculation of an instance is correct.

Class 1 = 25.92
Class 2 = 48.7
Class 3 = 48.7

x[20] is the 21st feature. Feature 18 and feature 19 are not extracted therefore the cost of x[20] (21st feature) is $14.51 + 11.41 = 25.92$. The cost of correctly classified Class 1 is 25.92. Cost of x[16] is 22.78. Therefore cost of correctly classified Class 2 and Class 3 is 48.7.

References

[1] Tan, M. (1993). Cost-sensitive learning of classification knowledge and its applications in robotics. Machine Learning, 13(1), 7–33. https://doi.org/10.1007/bf00993101