CS 412 Assignment 4 Report

Lanxiao Bai

1 Introduction

1.1 Decision Tree

Classification has long been a classic and important task of supervised learning. And for ordinary data, since it's not allowed to be compared by its magnitude, decision tree, in this case, is usually one of the best methods.

By recursively splitting datasets for training based on some attributes of the datasets, the decision tree algorithm can partition the input space $\mathcal{X}$ into different regions, so that the all (or at least majority of) points $x_i \in \mathcal{X}_j$ has the same label y.

In order to increase the purity of each branch after splitting, there're several evaluation metrics to determine based on which attribute and the corresponding value can the post-split purity reaches the highest value. In general, there're several properties that the impurity metric ϕ should meet in order to be use:
For simplicity, we consider the metric for binary classification, and let p be the probability of a label if uniformly randomly picked, then

1. $\forall p \in [0, 1], \phi(1/2, 1/2) \geq \phi(p, 1 - p)$
2. $\phi(0, 1) = \phi(1, 0) = 0$
3. p is increasing on $[0, 1/2]$ and decreasing on $[1/2, 1]$

These properties and be easily extend to k-ary classification, and some of the most commonly used methods are

- Entropy: $\phi(p) = -(p \log_2 p + (1 - p) \log_2(1 - p))$
- Gini index: $\phi(p) = 2p(1 - p)$

In the experiment, the k-ary Gini index

$$\phi(\mathbf{p}) = 1 - \sum_{p_i \in \mathbf{p}} p_i^2$$

is used to measure the performance of splits by each attribute.

Since decision tree suffers greatly from overfitting and thus generalize not well, we applied the following strategies to reduce the generalize error

- Set the maximum height h that a tree can grow
- Set the minimum Gini gain required to grow a node

1.2 Random Forest

In order to reduce the variance of the model, we applied random forest algorithm. By uniformly sampling data from training dataset with replacement and randomly choosing f attributes for a node to pick and grow, we made use of bagging strategy to randomize the training process. And other details of training algorithm we use is similar to the decision tree algorithm.

Theoretically speaking, as the number of trees we use in the tree grows, the model will converge greatly, and the phenomenon of overfitting will also be mitigated by majority vote of the trees we train.

1.3 Evaluation

Since it is possible that a node in the tree has unseen values or unseen features, in order that every data can be predicted as accurately as possible, I applied the strategy that in each node in the tree, we stored the majority label in the training dataset remaining, so that if a value or feature is unseen, we can use the majority vote the determine the prediction to be returned.

2 Experiment

To test our models, we ran our algorithm on four datasets: *balance.scale*, *nursery*, *led* and *synthetic.social*.

For simplicity, we fixed the Gini gain tolerance $\tau = 0.01$ and use the best depth for decision tree and best number of trees for random forest(which are explained later in Section 3).
And our results are as following:

		Overall Accuracy	Accuracy	Specificity	Precision	Recall	F_1	$F_{0.5}$	F_2
balance	1		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	2	1.000	1.0	1.0	1.0	1.0	1.0	1.0	1.0
train	3		1.0	1.0	1.0	1.0	1.0	1.0	1.0
balance	1		0.805	0.892	0.0	0.0	N/A	N/A	N/A
	2	0.644	0.765	0.756	0.725	0.775	0.749	0.734	0.764
test	3		0.72	0.774	0.702	0.653	0.677	0.692	0.663
	1		1.0	1.0	1.0	1.0	1.0	1.0	1.0
nursery	2		1.0	1.0	1.0	1.0	1.0	1.0	1.0
train	3	1.000	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	4		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	5		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	1		0.976	0.983	0.965	0.961	0.963	0.964	0.961
nursery	2		0.985	0.991	0.706	0.777	0.74	0.719	0.762
test	3	0.975	0.99	0.994	0.986	0.98	0.983	0.985	0.981
	4		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	5		0.999	0.999	0.0	N/A	N/A	N/A	N/A
led	1	0.860	0.86	0.899	0.77	0.771	0.77	0.77	0.771
train	2		0.86	0.771	0.899	0.899	0.899	0.899	0.899

led	1	0.857	0.857	0.894	0.766	0.775	0.77	0.768	0.773
test	2		0.857	0.775	0.899	0.894	0.896	0.898	0.895
synthetic	1		0.962	0.966	0.899	0.952	0.925	0.909	0.941
	2		0.973	0.98	0.941	0.951	0.946	0.943	0.949
	3	0.939	0.973	0.985	0.956	0.936	0.946	0.952	0.94
train	4		0.971	0.988	0.963	0.918	0.94	0.954	0.927
synthetic	1		0.734	0.816	0.504	0.511	0.507	0.505	0.51
	2		0.737	0.841	0.459	0.416	0.436	0.45	0.424
	3	0.501	0.772	0.837	0.508	0.556	0.531	0.517	0.546
test	4		0.759	0.84	0.528	0.522	0.525	0.527	0.523

Table 1: Results from Decision Tree on Four Datasets

		Overall Accuracy	Accuracy	Specificity	Precision	Recall	F_1	$F_{0.5}$	F_2
balance	1		0.955	1.0	1.0	0.333	0.5	0.714	0.384
	2	0.945	0.968	0.958	0.953	0.978	0.965	0.958	0.973
train	3		0.968	0.939	0.935	1.0	0.966	0.947	0.986
balance	1		0.898	0.995	0.0	0.0	N/A	N/A	N/A
	2	0.729	0.778	0.797	0.755	0.755	0.755	0.755	0.755
test	3		0.782	0.718	0.713	0.861	0.78	0.738	0.827
	1		1.0	1.0	1.0	1.0	1.0	1.0	1.0
nursery	2		0.966	0.956	0.918	0.985	0.95	0.931	0.971
train	3	0.966	0.977	1.0	0.929	0.066	0.123	0.257	0.081
	4		0.989	0.993	0.984	0.98	0.982	0.983	0.981
	5		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	1		0.945	0.941	0.888	0.951	0.918	0.9	0.938
nursery	2		0.974	1.0	1.0	0.015	0.03	0.071	0.019
test	3	0.939	0.971	0.977	0.95	0.958	0.954	0.952	0.956
	4		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	5		1.0	1.0	N/A	N/A	N/A	N/A	N/A
led	1	0.841	0.841	0.937	0.814	0.624	0.706	0.767	0.655
train	2		0.841	0.624	0.85	0.937	0.891	0.866	0.918
led	1	0.863	0.863	0.916	0.799	0.746	0.772	0.788	0.756
test	2		0.863	0.746	0.89	0.916	0.903	0.895	0.911
synthetic	1		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	2		1.0	1.0	1.0	1.0	1.0	1.0	1.0
	3	1.000	1.0	1.0	1.0	1.0	1.0	1.0	1.0
train	4		1.0	1.0	1.0	1.0	1.0	1.0	1.0
synthetic	1		0.881	0.933	0.802	0.739	0.769	0.789	0.751
	2		0.879	0.936	0.782	0.702	0.74	0.765	0.717
	3	0.775	0.9	0.926	0.768	0.815	0.791	0.777	0.805
test	4		0.89	0.905	0.753	0.847	0.797	0.77	0.826

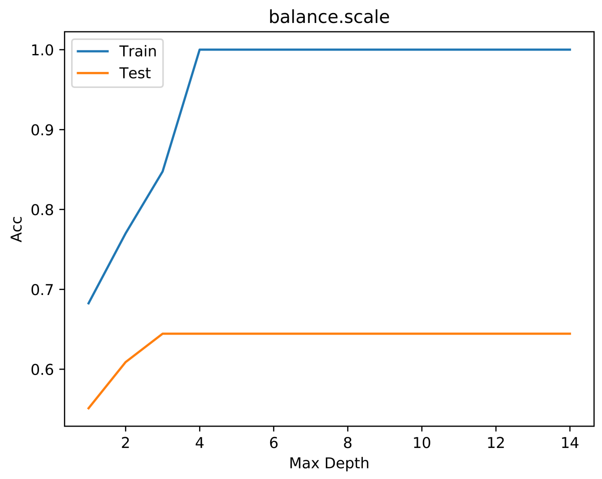
Table 2: Results from Random Forest Tree on Four Datasets

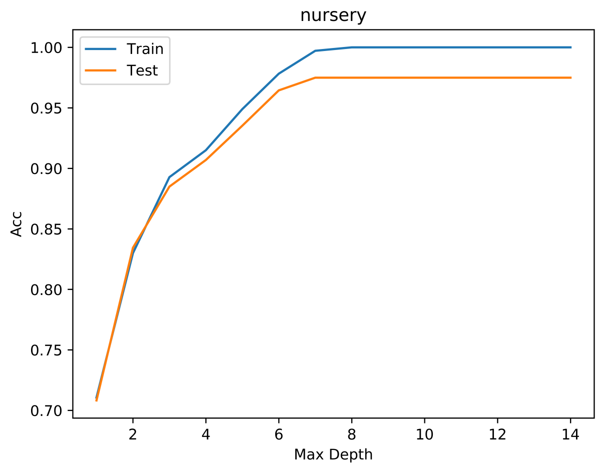
3 Model Tuning

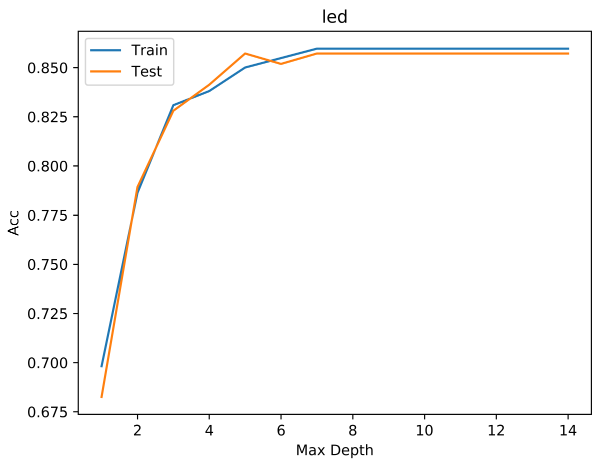
The tuning principle is following:

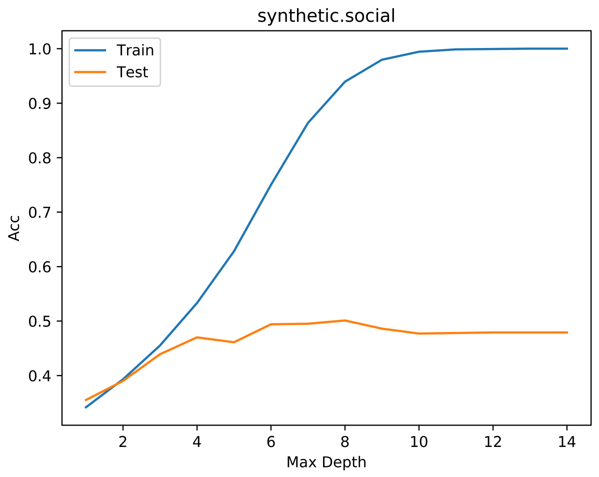
1. For Decision Tree, we fix the tolerance to 0 and change the max depth of the tree
2. For Random forest, we fix data sample rate to 0.8, feature sample number to the square root of the number of all features and change the number of trees in the forest

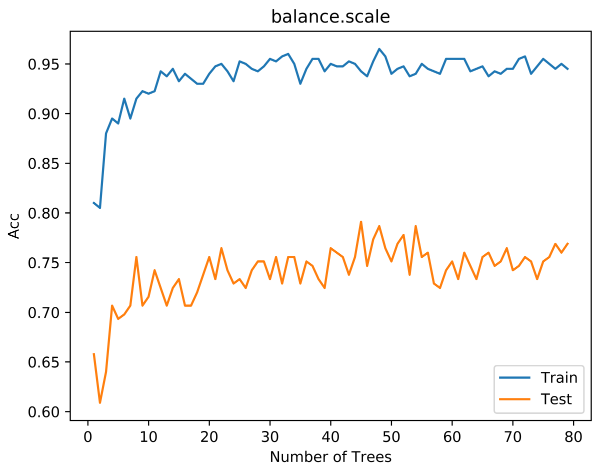
We got the following plots from the experiments above:

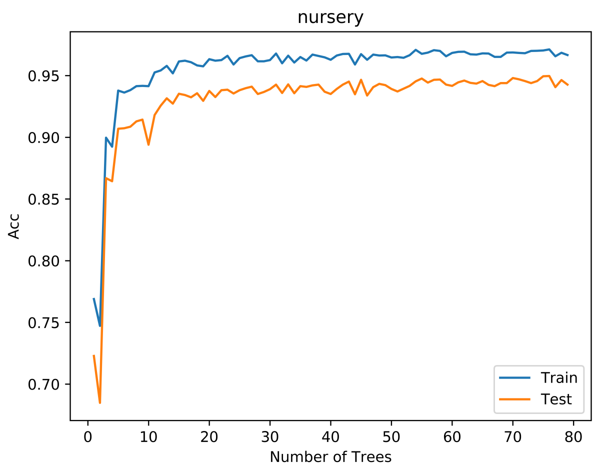


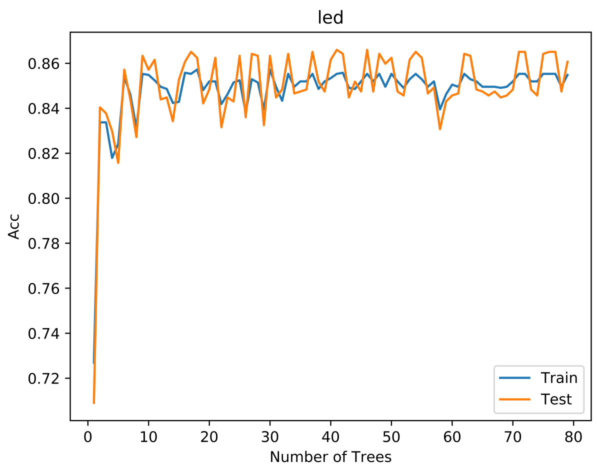


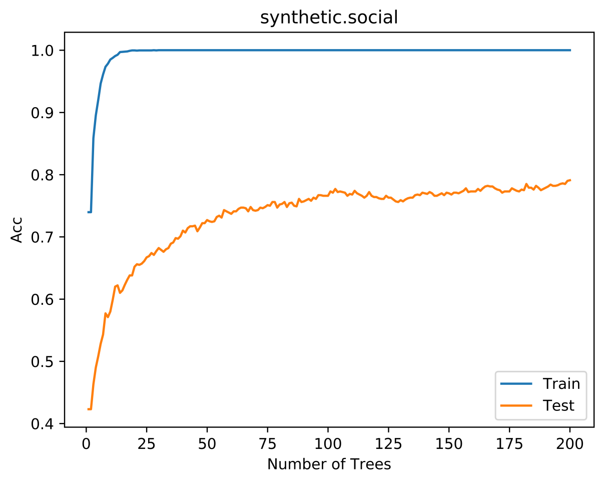

led
Train
Test
Acc
0.675
0.700
0.725
0.750
0.775
0.800
0.825
0.850
2
4
6
8
10
12
14
Max Depth


synthetic.social
Train
Test
Acc
0.4
0.5
0.6
0.7
0.8
0.9
1.0
2
4
6
8
10
12
14
Max Depth


balance.scale
Train
Test
Acc
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
0
10
20
30
40
50
60
70
80
Number of Trees







In order to save training time, we pick hyper-parameters that has least depth or number of trees possible while reaching the max accuracy, so that we choose:

	Decision Tree (Depth)	Random Forest (Number)
balance.scale	4	45
nursery	8	42
led	7	20
synthetic.social	8	200

4 Conclusion

From the results above, we conclude that ensemble method improves the performance of the basic classification method by reducing the variance and increase the testing accuracy.