CS412 - Introduction to Data Mining
Report on MP4

Abhishek Avinash Narwekar
UIN: 668601661, NetID: an41

December 2016

1 Implementation of Decision Trees

I have implemented the standard decision tree algorithm using the Gini Index as the measure for purity. My implementation of decision trees is inside the class `DecisionTree`, that contains functions to build the tree and make predictions on test data.

I store the nodes in the decision tree in a dictionary. The key of the node is a pair of tuples: the first tuple being the sequence of attributes on which the split has been performed upto that node, and the second tuple being the values taken by the said split attributes. Essentially, the state gives us the *path* from the root of the tree to a given node.

While building the tree, I first check if a node is terminal. Specifically, that happens when one of the following conditions occurs:

- There are no more attributes to split on
- There are no training instances at that node
- All training instances have the same label

For non-terminal nodes, I find the best attribute to split on next using the measure of Gini Index over the remaining attributes. After performing a split, I *don't* expand the node over *all* possible values the attribute can take. Instead, I expand only over those values that are taken by the training instances that exist *at that node*. This results in a significant decrease in run-time without a deterioration in performance. The pruning is especially important when there are a relatively large number of attributes. For instance, in the Poker Dataset, that has 10 attributes, some attribute can take upto 13 values. The tree, which grows exponentially in size, can become difficult to store in such a case. The pruning can play a crucial role in reducing the size of the problem. In the pruned the tree, the number of leaf nodes is at most equal to the number of training instances, which can be significantly less than the full tree, whose size is exponential in the number of attributes.

Dataset	Accuracy
balance	0.582
nursery	0.975
led	0.858
poker	0.627

Table 1: Overall accuracies on the four datasets using the Decision Tree Classifier

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0	0.837	0	0	0	0	0
1	0.647	0.813	0.742	0.647	0.691	0.721	0.664
2	0.644	0.694	0.631	0.644	0.637	0.634	0.641

Table 2: Metrics using Decision Tree on the Balance Dataset

2 Implementation of Random Forests

The random forests are implemented in the `RandomForest` class in `RandomForest.py`. We first performing Bootstrapping on the data, i.e. given N training instances, we first sample N points from the training data with replacement. We then build M decision trees, each of which is constructed as follows: for each split, we sample p attributes randomly from the *remaining* attributes, i.e. the attributes not yet chosen for splitting.

3 Model Evaluation

The metrics using Decision Tree Model for the four given datasets are shown in tables 2 to 5. The overall accuracies are shown in table 1.

For the Random Forest Model with the number of trees, $N = 100$ and the subset size, $M = \left\lfloor \sqrt{P} \right\rfloor$ (where $\lfloor\ \rfloor$ is the ceiling function and P is the total number of attributes), I have reported the metrics in tables 7 to 10. The overall accuracies are in table 6.

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0.967	0.98	0.96	0.967	0.963	0.961	0.965
1	0.746	0.992	0.708	0.746	0.727	0.715	0.738
2	0.978	0.996	0.991	0.978	0.985	0.989	0.981
3	1	1	1	1	1	1	1
4	0	0.999	0	0	0	0	0

Table 3: Metrics using Decision Tree on the Nursery Dataset

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0.781	0.893	0.765	0.781	0.773	0.768	0.778
1	0.893	0.781	0.901	0.893	0.897	0.899	0.894

Table 4: Metrics using Decision Tree on the led Dataset

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0.778	0.311	0.703	0.778	0.738	0.717	0.762
1	0.311	0.778	0.4	0.311	0.35	0.378	0.325

Table 5: Metrics using Decision Tree on the Poker Dataset

Dataset	Accuracy
balance	0.622
nursery	0.969
led	0.858
poker	0.677

Table 6: Overall accuracies on the four datasets using the Random Forest Classifier

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0	0.921	0	0	0	0	0
1	0.608	0.813	0.729	0.608	0.663	0.701	0.629
2	0.772	0.629	0.629	0.772	0.693	0.653	0.739

Table 7: Metrics using Random Forest on the Balance Dataset

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0.983	0.962	0.927	0.983	0.954	0.937	0.971
1	0.569	0.999	0.937	0.569	0.708	0.83	0.618
2	0.956	0.994	0.986	0.956	0.971	0.98	0.962
3	1	1	0.999	1	1	1	1
4	0	1	0	0	0	0	0

Table 8: Metrics using Random Forest on the Nursery Dataset

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0.781	0.893	0.765	0.781	0.773	0.768	0.778
1	0.893	0.781	0.901	0.893	0.897	0.899	0.894

Table 9: Metrics using Random Forest on the led Dataset

Class	Sensitivity	Specificity	Precision	Recall	F1	F0.5	F2
0	0.965	0.073	0.686	0.965	0.802	0.728	0.892
1	0.073	0.965	0.5	0.073	0.127	0.231	0.088

Table 10: Metrics using Random Forest on the Poker Dataset

Forest size	Subset Size	Accuracy
1	1	0.680
1	2	0.676
1	3	0.677
2	1	0.681
2	2	0.670
2	3	0.684
3	1	0.677
3	2	0.677
3	3	0.686
5	1	0.668
5	2	0.674
5	3	0.677
10	1	0.677
10	2	0.677
10	3	0.681
50	1	0.678
50	2	0.681
50	3	0.667
100	1	0.678
100	2	0.680
100	3	0.680

Table 11: Parameter search for Random Forest on the Poker Dataset

4 Selection of Parameters

In the Decision Tree classifier, there aren't any parameters to select. In the Random Forest Classifier, however, we have to select values for M (the number of trees) and p (the size of the subset of attributes sampled). Large values of p can result in highly correlated trees, while small values of p can result in poor models due to overly randomizing the attribute selection. A value of $p = \sqrt{P}$ has been suggested in the literature. We therefore vary p from 1 to $\sqrt{P}$ and M from 1 to 100 over non uniform steps and observe the accuracies. The results for the Poker Dataset have been shown in table 11. A combination of $M = 100$ and $p = \sqrt{P}$ perform well over all the datasets.

5 Conclusion

A comparison of overall accuracies for the decision tree classifier and the random forest classifier with $M = 100, p = \left\lfloor \sqrt{P} \right\rfloor$ is done in table 12. For the balance and the poker datasets, random forest comfortably outperforms the decision tree classifier. For the led dataset, random forest performs just as good as the decision tree classifier. For the nursery dataset, random forests performs

Dataset	Accuracy	
	Decision Tree	**Random Forest**
balance	0.582	0.622
nursery	0.975	0.969
led	0.858	0.858
poker	0.627	0.677

Table 12: Comparison of performance for Decision Tree and Random Forest Classifiers

slightly worse than the decision tree classifier, but the performance is fairly high regardless of this fact (96.9%). Thus, we conclude that the random forest classifier performs favorably with respect to the decision tree classifier in most datasets.