CS 412 HW4 Report

Name: Yining Wang

Netid: ywang282

UIN: 677886936

1. Classification Framework Explanation

- Decision Tree with gini-index
    - Implementation overview:
        - Read in the training set data and build a reverse index structure to store the training data to allow fast look up
        - Recursively build the tree. At each node, I calculated the gini index using the training data and select the attribute with the lowest gini-index to be the splitting attribute of that node and pass the remaining attributes down the tree
        - The number of branches at each node is equal to the number of attribute values of that splitting attribute
        - Parameters:
            - Depth of tree : max depth
        - The decision tree I built is a full grown tree, where each node is a struct contatining the following information:
            - Bool: terminate
            - Str: attribute
            - Str: label
            - Dict: children -> {attribute_value : tree_node}
    - Implementation details:
        - print_matrix(): print the matrix to stdout
        - get_decision (): return the decision for a tuple in the testing file
        - test_classifier (): run the classifier on the testing data and get the decisions for all the tuples
        - compute_gini_index (): compute the gini-index on a given node and return the index of the attribute with the lowest gini-index
        - build_reverse_index (): build a reverse index structure for the training data to allow fast loop up
        - buildtree (): recursive function to build up the decision tree

    - construct_decision_tree (): wrapper function to build the decision tree
    - main(): the main function that calls all the functions described above
  - Results in terms of overall accuracy

	balance	nursery	led	poker
Sklearn	76.0%	97.9%	85.8%	59.7%
Mine	64.9%	97.7%	86.2%	62.2%

- Random Forest
  - Implementation overview:
    - Read in the training set data and build a reverse index structure to store the training data to allow fast look up
    - Linearly build m trees, where m = 25
    - For each tree:
      - Random select k= 0.05*total_num_sample tuples with replacement from the training set data to construct the decision tree.
      - Each decision tree is a full grown tree, the same as the decision tree as described above, except for at each node, select f =0.4*total_num_attributes random attributes and only calculate the gini-index on those f randomly chosen attributes and the one with the lowest gini-index will be used as the splitting attribute
      - When predicting a label, the most frequent predicted label from the forest of trees is used to determine the actual predicted class label

- Implementation details:
  - print_matrix(): print the matrix to stdout
  - get_decision (): return the decision for a tuple in the testing file
  - test_classifier () : run the classifier on the testing data and get the decisions for all the tuples
  - compute_gini_index (): compute the gini-index on a given node and return the index of the attribute with the lowest gini-index
  - build_reverse_index (): build a reverse index structure for the training data to allow fast loop up
  - construct_random_forest(): build the random forest
  - main(): the main function that calls all the functions described above

2. Performance Evaluation

- ### Performance Evaluation for Decision Tree
  - Balance Set with accuracy: 64.889%
    - ❖ Confusion matrix:

label	1	2	3	all
1	0	15	7	22
2	4	81	17	102
3	17	19	65	101
all	21	115	89	225

    - ❖ Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	0.000%	89.655%	0.000%	0.000%	0.000%	0.000%
2	79.412%	72.358%	70.435%	79.412%	74.654%	74.654%
3	64.356%	64.356%	80.645%	73.034%	64.356%	68.421%

- Nursery Set with accuracy: 97.699%
    - Confusion matrix:

label	1	2	3	4	5	all
1	1544	32	20	0	0	1596
2	35	92	0	0	3	130
3	22	0	1514	0	0	1536
4	0	0	0	1605	0	1605
5	0	0	0	0	0	0
all	1601	124	1534	1605	3	4867

- Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	96.742%	98.257%	96.440%	96.742%	96.591%	96.515%
2	70.769	99.324%	74.194%	70.769%	72.441%	72.441%
3	98.568%	99.400%	98.696%	98.568%	98.632%	98.632%
4	1.000%	1.058%	1.000%	1.000%	1.000%	1.000%
5	NA	99.938%	0.000%	NA	0.000%	NA

- Led Set with accuracy: 86.155%
  - Confusion matrix:

label	1	2	all
1	267	84	351
2	73	710	783
all	340	790	1134

- Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	76.068%	90.677%	78.529%	76.068%	77.279%	77.279%
2	90.677%	76.068%	89.421%	90.677%	90.044%	90.044%

- Poker Set with accuracy: 62.242%
  - Confusion matrix:

label	1	2	all
1	347	112	459
2	144	75	219
all	491	187	678

  - Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	75.599%	34.247%	70.672%	75.599%	73.053%	73.053%
2	34.247%	75.599%	40.107%	34.247%	36.946%	36.946%

- Performance Evaluation for Random forest
  - Balance Set with accuracy: 76.889%
    - Confusion matrix:

label	1	2	3	all
1	0	11	11	22
2	6	87	9	102
3	7	8	86	101
all	13	106	106	225

    - Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	0.000%	0.000%	0.000%	0.000%	0.000%	NA
2	85.294%	84.553%	82.075%	85.294%	83.654%	83.654%
3	85.149%	83.871%	81.132%	85.149%	83.092%	83.092%

- Nursery Set with accuracy: 97.699%
  - ❖ Confusion matrix:

label	1	2	3	4	5	all
1	1544	32	20	0	0	1596
2	35	92	0	0	3	130
3	22	0	1514	0	0	1536
4	0	0	0	1605	0	1605
5	0	0	0	0	0	0
all	1601	124	1534	1605	3	4867

  - ❖ Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	96.742%	98.257%	96.440%	96.742%	96.591%	96.515%
2	70.769	99.324%	74.194%	70.769%	72.441%	72.441%
3	98.568%	99.400%	98.696%	98.568%	98.632%	98.632%
4	1.000%	1.058%	1.000%	1.000%	1.000%	1.000%
5	NA	99.938%	0.000%	NA	0.000%	NA

- Led Set with accuracy: 86.067%
    - Confusion matrix:

label	1	2	all
1	266	85	351
2	73	710	783
all	339	795	1134

    - Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	75.783%	90.677%	78.466%	75.783%	77.101%	77.101%
2	90.677%	75.783%	89.308%	90.677%	89.987%	89.987%

- Poker Set with accuracy:66.372%
  - ❖ Confusion matrix:

label	1	2	all
1	419	40	459
2	188	31	219
all	607	71	678

  - ❖ Other measures:

label	sensitivity	specificity	precision	recall	F-1 Score	F \beta score
1	91.285%	14.155%	69.028%	91.285%	78.611%	78.611%%
2	14.155%	91.285%	43.662%	14.155%	21.379%	21.379%

3. Choice of parameter

- Depth of the decision tree: d
    - I made all my decision tree full grown trees, or in other words, they are all of max depth
    - Reason: for all the data sets, the number of unique attributes are not very big, and even with full grown trees, the program still terminates in reasonable time, therefore I set the tree to be max depth
- Number of samples in the ensemble methods for each tree: n
    - I set n = 0.05* total_num_samples
    - Reason: here is the matrix with different values for n (when other parameters are all set). Notice that the overall accuracy almost does not change at all for the following values for n (the small change is because of the randomization in the random forest) so with the same performance, I choose the smallest n which is n = 0.05* total_num_samples so that the running time could be shortened.

n	Overall averaged (3 runs) accuracy for led set
n = 0.05* total_num_samples	86.4%
n = 0.2* total_num_samples	86.2%
n = 0.4* total_num_samples	86.4%
n = 0.8* total_num_samples	85.8%

- Number of attributes randomly selected at each node during random forest construction: f
    - I set f = 0.4* total_num_attributes
    - Reason: here is the matrix with different values for f (when other parameters are all set). When f = 0.4* total_num_attributes the overall performance is the best.

f	Overall averaged (3 runs) accuracy for led set
f = 0.2* total_num_attributes	85.9%
f = 0.4* total_num_attributes	86.2%
f = 0.6* total_num_attributes	84.6%
f = 0.8* total_num_attributes	82.8%

- Number of decision trees in the forest: t
    - I set t = 25
    - Reason: here is the matrix with different values for t (when other parameters are all set) . I noticed that the accuracy slightly increased as I increased t from 5 to 25 and then stays almost the same as I increased t to above 25. However, the accuracy tends to be more stable (less fluctuation among different runs) as I increase t. Therefore, taking time into consideration I choose t = 25

t	Overall averaged (3 runs) accuracy for balance set
t = 5	73.3%
t = 10	74.4%
t = 15	75.0%
t = 20	75.5%
t = 25	74.9%
t = 40	74.4%
t = 80	74.7%
t = 100	75.3%

4. whether ensemble method improved result

Here is the matrix I got in terms of accuracy:

	led	nursery	balance	poker
Decision tree	86.155%	97.699%	64.889%	62.242%
Random forest	86.067%	97.699%	76.889%	66.372%

The conclusion is that my ensemble method does improve my result!
the accuracy for led didn't change that much, the accuracy for nursery is already good enough without the ensemble method and the accuracy for balance and poker set increased a lot!