CS 412 ONL : Assignment 4

Ramya Narayanaswamy (rpn2)

Goal of the project:

The goal of the project was to implement a classification framework using Decision Tree and Random Forest ensemble methods, evaluate the framework for the given test cases and report various classification metrics. The report is organized as follows. Part 1 gives a brief description of the classification framework. Part 2 is model evaluation for every test case.

Part1 : Brief Overview of Classification Framework

The framework was implemented in Python 3.6.1. The following are the components of the framework.

Step1: Reading and storing the training and test files : (ReadTrFile.py and ReadTeFile.py)

The training data file is parsed and stored as a Python dictionary. The keys of the dictionary are line numbers or unique index of each training data. Each value is a 2 element list. The first element of the list is the label and second element is the attribute dictionary. The attribute dictionary is a key-value pair of attribute name with attribute values. Thus, the entire training data is parsed and stored as a Python dictionary. In a similar fashion, the testing data is parsed and stored as testing dictionary.

Data structures used in Classification framework

During parsing of training file,the following data structures were populated to aid in Gini Index calculation. They are AVC dictionary, AV dictionary, class dictionary. AVC (Attribute, Value, Class) dictionary has the tuple (attribute,value,class) as key and value is number of occurrences in training data set that belong to a class with the respective attribute and value. The AV dictionary has attribute name as keys and value is another dictionary. This second dictionary has “attribute value” as keys and number of occurrences of the “attribute value” as values. The class dictionary has class labels as keys and number of instances in training data that have this label as values.

Step2: Implementation of a basic classifier: (DecisionTree.py, Training.py, Testing. py)

A Decision Tree based basic classifier was implemented in the code. Decision Tree’s could naturally handle both binary and multi-class classification problems, hence no special provisions were needed in the code to handle multi-class classification. The top-level of the basic classifier is DecisionTree.py.

Training (Training.py):

A recursive greedy version of decision tree was implemented in Training.py using the method generatetree(self,D,al,cl,avc,depth) and and implementation closely followed the algorithm steps provided in the course textbook. Gini Index was used to determine the best attribute to split and the number of branches of the tree is determined by the number of possible values that an attribute could take. The AttributeSelection(self,D,al,cl,avc) method calculates the gini index and returns the required attribute for split. The AVC, AV and class data structures were calculated at each node to facilitate gini index calculation and to prevent repeated traversal of the input training data. *This technique helped in run-time* *reduction and helped to achieve run-time constraints of the project*. The maximum height of a tree could be upto the number of attributes in the training set,with height as a tunable parameter for improving accuracy of test cases in this framework.

Testing(Testing.py):

For each test data point, the decision tree is traversed once until a leaf node is reached. The class label of the leaf node becomes the predicted class for the test instance. While traversing the tree, if the attribute used for node split is not defined for a test instance or if the attribute value of test instance is unseen in the training set, then a random branch is chosen to complete the traversal.

Further Notes on Decision Tree Implementation :

The Decision Tree generator recursive method has the following termination conditions: 1) All of the current data have one class associated with them. A leaf node is created and assigned the same class 2) No feature/attribute is available for further branching,a leaf node is created and assigned a label with majority voting on current data 3) A required depth has been reached, a leaf node is created and assigned a label with majority voting on current data. The majority voting scheme could have a tie if one or more labels occur with same frequency. The following techniques were tried and their use/no-use is justified

1) Use Python's inbuilt MAX method on label dictionary. This technique was simple, fast and had predictability and less variation in overall accuracy
2) Implement a tie-breaker as follows. The framework keeps track of number of leaf nodes belonging to a class. The class with lowest number of leaf nodes is chosen as label when a tie-breaking is required. The accuracy of this technique was not beneficial for all the test cases. It was found to be less than method 1. Hence, it was unused.
3) Randomly choose a label among the high-frequency labels. This method has a significant variation in overall accuracy (~ 8%). Hence, it was unused

The Attribute Selection method chooses an attribute with lowest Gini index. There could be a scenario where there could be more than one attribute with same low Gini index values. The following techniques were tried and their use/no-use is justified

1) Use Python's inbuilt MIN method on label dictionary. This technique was simple, fast and had predictability and less variation in output accuracy.
2) Randomly choose an attribute. This method has a significant variation in accuracy (~ 4%). Hence, it was unused for metric calculations of the report.
3) Create a list of attributes with lowest gini index. Pick last element of the list. This technique had about 2-3% better accuracy than method1. However, the final code was reverted to choose random attribute after Piazza discussions

Step3: Implementation of an ensemble classifier (RandomForest.py)

An ensemble classifier was implemented using Random Forest with Random input selection. Multiple decision trees were built and each decision tree was traversed once for every test instance. Each decision tree returns a predicted class. The ensemble method uses the most frequent class among the classes returned by individual trees to predict a class.

Each decision tree (of random forest) training is quite similar to training of a basic classifier with a few differences as stated below. Firstly, the attribute selection in decision tree of a random forest picks "M" random attributes to be considered for node splitting at each node. The attribute with lowest Gini Index among these M attributes is picked for node split. The number of attributes (M) for node split is fixed for all nodes and for all trees of the random forest. At each node, random attribute selection (feature bagging) was used. The tree generation and attribute selection for random forests were optimized by refactoring logic for AVC, AV data structure updates. The respective methods are generatetreeRF(self,D,cl,depth) and AttributeSelectionRF(self,D,cl) in Training.py   Secondly, data bagging feature was coded as an option to experiment for various test cases. If there are N samples in training set, each decision tree was trained with N randomly chosen (with replacement) samples from the training set.

Miscellaneous codes : utilities.py has code for confusion matrix calculation and QualityCalc.py has code for evaluation metrics

Step4: Classifier evaluation for various data sets

The basic classifier and ensemble classifier have the following parameters that could be tuned.

Parameter tuning:

The parameters of decision tree and random forest were tuned based on the test data and fixed to a certain value. The parameters were tuned until the accuracy plateaued. The results of the training data set were obtained with the same fixed parameters.

Decision Tree : *Height of the tree is an explicit parameter* tunable in the range 1 to number of attributes. Height of a tree is chosen such that increasing the height does not bring in any additional significant increase in accuracy. In general, lowest height at which accuracy plateaus is considered.

Randomforest : *Parameters are number of trees, number of random attributes for feature bagging at each node, height of trees, data bagging ON or OFF*. For databagging, the sample set size is same as overall data set, but the samples for each tree are obtained by sampling with replacement.

Until a certain point, number of trees directly influence the accuracy. This is because the random input selection tries to reduce the correlation among various trees. Also, higher number of trees reduce the run-run variation in accuracy of results. The number of trees is found by experimentation based on trade-off of overall accuracy (and accuracy variation) vs run-time limit of project (under 3 mins). The number of random attributes were tuned from 1 to MAX(Number of attributes, Log_2 (Number of training samples) + 1) and fixed to a value that happened to produce better overall accuracy. The above empirical formula was arrived after experimentation with the given test data for all 4 data samples.

Notes on metrics evaluation:

The parameters were tuned against the test set until desirable accuracy was achieved. With the parameters fixed after tuning, metrics for both test and training set were obtained. There was no separate tuning to improve the results of training set as that would overfit the classifier to training data. Test set was treated as hold-out validation set was this assignment purpose.

RESULTS

All the results are rounded to three decimal places.

i) Balance data set :

Parameters for Decision Tree: Tree Height = 3

Parameters for Random Forest : Tree Height = 2, Number of Trees = 150, Data bagging : ON, Number of random attributes: 2. It has been observed that results are pretty much within the same range beyond 80 trees. However, 150 was used to reduce the variation range of results and as run-time was under 0.7 seconds.

A) Basic method on training data:

i. Overall Accuracy : 0.838

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.93	0.984	0.455	0.185	0.263	0.352	0.210
Label2	0.873	0.850	0.839	0.898	0.868	0.850	0.885
Label3	0.873	0.873	0.856	0.872	0.865	0.860	0.869

B) Basic method on test data:

i.Overall Accuracy : 0.64

iii. Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.84	0.931	0.0	0.0	UNDEF	UNDEF	UNDEF
Label2	0.729	0.715	0.685	0.745	0.714	0.696	0.732
Label3	0.711	0.742	0.68	0.683	0.677	0.678	0.675

C)Ensemble method on training data:

i. Overall Accuracy : 0.898

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.933	1.0	UNDEF	0.0	UNDEF	UNDEF	UNDEF
Label2	0.943	0.925	0.918	0.962	0.94	0.927	0.953
Label3	0.92	0.883	0.878	0.963	0.918	0.893	0.944

D) Ensemble method on test data

i. Overall Accuracy : 0.836

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.902	1.0	UNDEF	0.0	UNDEF	UNDEF	UNDEF
Label2	0.876	0.805	0.803	0.961	0.875	0.831	0.925
Label3	0.893	0.895	0.874	0.891	0.882	0.877	0.888

Conclusion: The ensemble technique improves the overall accuracy significantly ~(17-21%) over the basic technique on the test set. The overall accuracy of ensemble techniques varies run-run and the maximum range difference is ~5% for balance data test set. However, there is only ~6% improvement with ensemble techniques for training data.

ii) Nursery data set

Parameters for Decision Tree: Tree Height = 7
Parameters for Random Forest : Tree Height = 7, Number of Trees = 100, Data bagging : ON, Number of random attributes: 8. The lower limit for number of trees to get similar accuracy is 80. 100 was set to improve certain class-label metrics. Data bagging ON/OFF did not change the overall accuracy by a significant percentage for this test case.

A)Basic method on training data:

i. Overall Accuracy : 0.995
ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.996	0.997	0.995	0.992	0.993	0.994	0.993
Label2	0.998	0.998	0.932	0.97	0.950	0.939	0.962
Label3	0.998	0.998	0.996	0.997	0.997	0.996	0.997
Label4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label5	0.999	1.0	UNDEF	0.0	UNDEF	UNDEF	UNDEF

B) Basic method on test data:

i. Overall Accuracy : 0.969
ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.969	0.977	0.952	0.953	0.953	0.953	0.953
Label2	0.983	0.990	0.664	0.715	0.689	0.674	0.705
Label3	0.986	0.991	0.981	0.975	0.978	0.980	0.976
Label4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label5	1.0	1.0	UNDEF	UNDEF	UNDEF	UNDEF	UNDEF

C) Ensemble method on training data:

i. Overall Accuracy : 0.999

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.999	1.0	1.0	0.999	0.999	0.999	0.999
Label2	0.999	0.999	0.99	1.0	0.995	0.992	0.998
Label3	0.999	0.999	0.999	1.0	0.999	0.999	0.999
Label4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label5	0.999	1.0	1.0	0.5	0.667	0.833	0.556

D) Ensemble method on test data:

i. Overall Accuracy : 0.981

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.981	0.983	0.966	0.977	0.971	0.968	0.975
Label2	0.991	0.997	0.876	0.761	0.815	0.851	0.782
Label3	0.990	0.993	0.985	0.984	0.985	0.985	0.985
Label4	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label5	1.0	1.0	UNDEF	UNDEF	UNDEF	UNDEF	UNDEF

Conclusion: The overall accuracy of the training data set is already high using a single decision tree classifier and renders ensemble technique to be futile. Similarly, for the test data, there is a small improvement in overall accuracy and and Label 2 Recall and F1 scores using ensemble techniques.

If overall accuracy is the determining factor for an application, then ensemble techniques does not bring in significant improvement for “nursery” data set. However, individual label related metrics are better for classes with ensemble techniques in the test data set.

iii) Led data set

Parameters for Decision Tree: Tree Height = 7

Parameters for Random Forest : Tree Height = 7, Number of Trees = 65, Data bagging : ON, Number of random attributes: 7.Data bagging ON/OFF did not change the overall accuracy by a significant percentage for this test case.

A)Basic method on training data:

i. Overall Accuracy : 0.86

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.86	0.899	0.771	0.77	0.77	0.77	0.77
Label2	0.86	0.77	0.899	0.899	0.899	0.899	0.899

B)Basic method on test data:

i. Overall Accuracy : 0.859

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.859	0.897	0.771	0.775	0.773	0.771	0.774
Label2	0.859	0.775	0.899	0.897	0.898	0.898	0.897

C)Ensemble method on training data:

i. Overall Accuracy : 0.86

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.86	0.905	0.778	0.757	0.767	0.773	0.761
Label2	0.86	0.757	0.894	0.905	0.899	0.896	0.902

D)Ensemble method on test data:

i. Overall Accuracy : 0.862

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.862	0.903	0.780	0.769	0.775	0.778	0.771
Label2	0.862	0.769	0.897	0.902	0.900	0.898	0.901

Conclusion: By analyzing the above results, it is clear that ensemble techniques do not have any advantage over basic classifier for “led” data set, which involved binary classification.

iv) Synthetic social dataset

Parameters for Decision Tree: Tree Height = 8
Parameters for Random Forest : Tree Height = 8, Number of Trees = 135, Data bagging : ON, Number of random attributes: 12

A)Basic method on training data:
i. Overall Accuracy : 0.937
ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.968	0.983	0.947	0.922	0.934	0.942	0.927
Label2	0.967	0.979	0.937	0.934	0.935	0.936	0.934
Label3	0.976	0.984	0.954	0.953	0.953	0.954	0.953
Label4	0.961	0.969	0.91	0.937	0.923	0.915	0.931

B)Basic method on test data:
i. Overall Accuracy : 0.478 (run-run variations cause accuracy to be in range 45%-49%)
ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.726	0.824	0.488	0.459	0.473	0.482	0.465
Label2	0.733	0.828	0.454	0.441	0.447	0.451	0.443
Label3	0.75	0.819	0.465	0.522	0.492	0.476	0.509
Label4	0.747	0.833	0.504	0.494	0.499	0.501	0.496

A)Ensemble method on training data:
i. Overall Accuracy : 1.0
ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Label4	1.0	1.0	1.0	1.0	1.0	1.0	1.0

D) Ensemble method on test data:

i. Overall Accuracy : 0.783

ii.Label Statistics:

	Accuracy	Specificity	Precision	Recall	F-1	F-Beta0.5	F-Beta2
Label1	0.886	0.93	0.80	0.765	0.782	0.793	0.772
Label2	0.888	0.935	0.788	0.743	0.765	0.778	0.751
Label3	0.906	0.938	0.795	0.801	0.798	0.796	0.800
Label4	0.886	0.907	0.753	0.824	0.787	0.766	0.808

Conclusion: By analyzing the above results, it is clear that ensemble techniques produce significant results when compared to basic classifier. The accuracy is 100% for the training data set and increased by ~30% for test data set.

General conclusions : It has been observed that accuracy of the classifier depends on the data set and data distribution in every training set.