CSE_601: DATA MINING

PROJECT 3 Report

Shuo Zhang
Chen Liang
Muhammad Aamir Masood

Project 3 Classification Algorithm

Decision Tree, Random Forest and Boosting

1 Code Workflow

1.1 Decision Tree

In the implementation of decision tree for project 3, the datasets project3_dataset1.txt and project3_dataset2.txt were used. These files were converted into CVS files, in order to handle them in a better way and have been attached with the code for this project. The purity of datasets was checked and then entropy was used as a measure of splitting the nodes in the dataset. Firstly, entropy of the whole dataset was calculated which acted as a reference for later stages. The splits with the lowest entropy were found. So, the information gain for a given attribute was computed by taking the entropy of the whole set and subtracting it with the entropies of sets that are obtained by breaking the whole set into one piece per attribute category.

$$\boldsymbol{E(S) = \sum_{i=1}^{n} -p_i \log_2 p_i}$$

The goal of decision tree was to find those descriptive features which contain the most "information" regarding the target feature and then split the dataset along the values of these features such that the target feature values for the resulting subsets were as pure as possible based on the measure of entropy. The descriptive feature which left the target feature most purely was the most informative one. This process was done until a stopping criterion was met which was the max_depth of the tree. The leaf nodes contained the rules for traversing the tree.

First dataset contained 30 features in addition to the labels. The decision tree for first dataset for max_depth of 3 was as follows:

```
{'f_23 <= 105.95': [{'f_28 <= 0.13505': [0.0,
                                         {'f_22 <= 27.575': [0.0,
                                                             1.0]}]},
                    {'f_23 <= 117.45': [{'f_25 <= 0.1361': [0.0,
                                                            1.0]},
                                        {'f_28 <= 0.085865': [0.0,
                                                              1.0]}]}]}
```

Second dataset consisted of nine features in addition to labels. The decision tree for second dataset for max_depth of 5 was as follows:

```
{'f9 <= 31.5': [{'f8 <= 11.105': [{'f7 <= 18.48': [{'f7 <= 18.175': [0,
                                                                      1]},
                                                   0]},
                                  {'f2 <= 0.51': [0,
                                                  {'f7 <= 25.59': [{'f3 <=
4.265000000000001': [0,
```

```
1]},
                                                                    {'f9 <=
26.5': [1,

0]}]}]}]},
                {'f9 <= 50.5': [{'f6 <= 68.5': [{'f7 <= 19.83': [1,
                                                                  {'f4 <= 3
9.495000000000005': [0,

1]}]},
                                                        {'f7 <= 23.990000000000002
': [0,

{'f1 <= 191.0': [1,

0]}]}]},
                                    {'f5 = Present': [{'f3 <= 6.705': [{'f2 <=
4.15': [0,

1]},
                                                                       {'f6 <=
34.0': [0,

1]}]},
                                                          {'f2 <= 7.605': [0,
                                                                           1]}]}]}
]}
```


1.2 Random Forest:

Random forest was created by an amalgamation of decision trees. 20 percent of the features were randomly selected. Number of trees for random forest was decided to be 3. For each tree, randomly 80% of the records in the dataset were chosen with replacement. The best split for the decided feature was selected by using the decision trees implemented before. At the end, the results were combined and majority voting was used to ascertain the label of the record.

1.3 Boosting:

In boosting, the weights were initially assigned uniformly. A bootstrap sample based on weights was used in each case and the classification was kept track of for each record. The records who had been misclassified were given higher weights in new iterations and the records which were correctly classified were given the reduced weights for next iteration. Final prediction was weighted average of all the classifiers with weight representing the training accuracy.

1.4 10-fold Cross Validation:

In 10-fold cross validation, data was divided into 10 disjoint subsets. The data was trained on 9 partitions whereas the last partition was used for testing purposes.

1.5 Performance Measures

Accuracy, precision, recall, and F-1 measures were used for calculating the performance of datasets and their equations are given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - 1 = \frac{2TP}{2TP + FP + FN}$$

2 Results

The performance was measure for all the algorithms and results have been tabulated below:

Table 1 Performance measures for decision tree

Data	Accuracy	Precision	Recall	F-1
Dataset1	0.92	0.99	0.92	0.96
Dataset2	0.47	0.85	0.52	0.64

Table 2 Performance measures for random forest

Data	Accuracy	Precision	Recall	F-1
Dataset1	0.91	0.98	0.92	0.96
Dataset2	0.48	0.84	0.51	0.64

Table 3 Performance measure for Boosting

Data	Accuracy	Precision	Recall	F-1
Dataset1	0.89	0.90	0.87	0.85
Dataset2	0.43	0.78	0.49	0.60

3 Analysis of Algorithms

3.1 Decision Trees

PROS:

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

CONS:

- Not expressive enough for modelling continuous variables
- Possibility of spurious relationships
- Instability, small changes can cause huge changes in output

3.2 Random Forest

PROS:

- Decorrelates trees (relative to bagged trees)
- Important when dealing with multiple features which may be correlated
- reduced variance (relative to regular trees)

CONS:

- No interpretability
- Overfitting can easily occur
- Need to choose the optimum number of trees

3.3 Boosting

PROS:

- Can be used to solve almost any objective function
- Can easily handle qualitative (categorical) features

CONS:

- Training takes longer time
- More sensitive to overfitting
- Harder to tune parameters

3.4 Nearest Neighbor

PROS:

- Intuitive and simple
- Constantly evolves
- Very easy to implement

CONS:

- Curse of dimensionality
- Sensitive to outliers

3.5 Naïve-Bayes

PROS:

- Computationally fast
- Simple to implement
- Works well with high dimensions

CONS:

- Relies on independence assumption and will perform badly if this assumption is not met