Learning Project Evaluation and Analysis Report

Tingjun Li

li400@purdue.edu

Problem 1:

```
pod1-5 52 $ python regression-tree.py train-set test-set
Number of Feature splits=8
Threshold=0.1
SQUARED LOSS=0.776120005813
```

Training set(size:102) and test set(size:103) are both obtained from the original data set with same probability. Threshold is 0.1. Based on the squared loss result 0.776, I would say the prediction quality is pretty good. This model can effectively predict the riskiness for a new case.

Problem 2:

Result:

Squred Loss, Num of feature split	Train: T1+T2, Test: T3	Train: T1+T3, Test: T2	Train: T2+T3, Test: T1
Threshold: 0.75	1.6875, 0	1.255625, 0	1.705625, 0
Threshold: 0.075	0.890580746153, 8	0.987367791797, 23	0.760896909268, 18

Analysis:

From the different combination of training set, test set and threshold, we can see that the size of the threshold can affect the result. Setting the threshold smaller will significantly enlarge the tree size(number of feature split). However, setting a threshold too large will not be able to record significant feature split. As we see in the demo, when threshold is 0.75, 0 feature split is recorded. The result was predicted over the whole set of training data, therefore the result was terrible. In the other hand, when setting the threshold more reasonable smaller (0.075), the result is much better.

However, when I test with other threshold value. Smaller threshold did not guarantee a better result. Theoretically setting the threshold smaller will enhance the performance, but it will also takes features that has no significant relationship with our predicting target value (noise). Therefore, choosing the “sweet spot” for the threshold is really important in getting a good prediction. In our case, setting a threshold too small is not quite effective. It’s probably because only small numbers of features are related to our predicting value. There might be other reasons such as training set too small that can also affect prediction accuracy.