CS446: Machine Learning **Spring 2017**

Problem Set 2

KaiZhao Liang(kl2) *Handed In: February 10, 2017*

1. **Learning Decision Trees**

   a. The root of the decision tree should be Holiday.

$$Entropy(S) = -\frac{35}{50}log(\frac{35}{50}) - \frac{15}{50}log(\frac{15}{50}) \approx 0.611$$

If the root is Holiday,i.e. a= Holiday,

$$Entropy(S_{Holiday=yes}) = -\frac{20}{21}log(\frac{20}{21}) - \frac{1}{21}log(\frac{1}{21}) \approx 0.191$$

$$Entropy(S_{Holiday=no}) = -\frac{15}{29}log(\frac{15}{29}) - \frac{14}{29}log(\frac{14}{29}) \approx 0.693$$

$$Gain(S,a) = Entropy(S) - [\frac{\|S_{Holiday=yes}\|}{\|S\|}Entropy(S_{Holiday=yes}) + \frac{\|S_{Holiday=no}\|}{\|S\|}Entropy(S_{Holiday=no})]$$

So

$$Gain(S,a = Holiday) = 0.611 - (\frac{21}{50} \times 0.191 + \frac{29}{50} \times 0.693) \approx 0.129$$

If the root is ExamTomorrow,i.e. a= ExamTomorrow,

$$Entropy(S_{ExamTomorrow=yes}) = -\frac{10}{15}log(\frac{10}{15}) - \frac{5}{15}log(\frac{5}{15}) \approx 0.637$$

$$Entropy(S_{ExamTomorrow=no}) = -\frac{25}{35}log(\frac{25}{35}) - \frac{10}{35}log(\frac{10}{35}) \approx 0.598$$

$$Gain(S,a) = Entropy(S) - [\frac{\|S_{ExamTomorrow=yes}\|}{\|S\|}Entropy(S_{ExamTomorrow=yes}) + \frac{\|S_{ExamTomorrow=no}\|}{\|S\|}Entropy(S_{ExamTomorrow=no})]$$

So

$$Gain(S,a = ExamTomorrow) = 0.611 - (\frac{15}{50} \times 0.637 + \frac{35}{50} \times 0.598) \approx 0.0013$$

Because $Gain(S,a = Holiday) > Gain(S,a = ExamTomorrow)$, the root attribute should be Holiday.

b.
```
if Color = Blue:
    if Size = Small:
        Inflated = False
    if Size = Large:
            if Act = Stretch:
                if Age = Adult:
                    Inflated = False
                if Age = Child:
                    Inflated = True
            if Act = Dip:
                Inflated = True
if Color = Red:
    if Size = Small:
            if Act = Stretch:
                if Age = Adult:
                    Inflated = False
                if Age = Child:
                    Inflated = True
            if Act = Dip:
                Inflated = True
    if Size = Large:
            if Act = Stretch:
                if Age = Adult:
                    Inflated = False
                if Age = Child:
                    Inflated = True
            if Act = Dip:
                Inflated = True
```

c. No, ID3 does not guarantee a globally optimal decision tree. Since it's based on the greedy selection of the "best" attribute, it could be stuck at the local minima. When selecting the current node, it is myopic and only cares about the information gain after one split. However, the total information gain after multiple splits could be higher, if not choosing this "best" attribute. In addtion, it also tends to favour the attributes that have more values.

2. **Decision Trees as Features**

a. The following algorithms are trained on the features mentioned in the question.

b. Explannation of the implementations: The implementations of the 3 algorithms are in java and based on the example code given. The .java files are located in $./kl2-hw2/decision-trees/src$ The implementation of SGD is in the SGD.java. The implementation of the SGD with stumps is in the main of the wekaTest.java. The order of experiments, also the order of printing out results in terminal is from the top to bottom: normal SGD, DT with unlimited depth, DT with depth of 8, DT with depth of 4, SGD with stumps.

SGD:keep updating until the change of the Cost is lower than some value or the Cost is less than the threshold. After loop through all the training data in the train set, randomly shuffles the set.
DT with certain height: Using the weka.classifer.trees.Id3 package, by changing the parametric value of the setMaxDepth() instance method, we get DTs with different Depth.
SGD with stumps: first training 100 DTs on the trainning data. Before trainning, randomly shuffle the data set and train on half of the data everytime. Repeat this process until we get target quantity of new features. Then use the new data set with the new features to train the SGD mentioned above. When testing, do the same as what we do for the train set.

$\Re$. **Evaluation and Report**
Ranking the algorithms with decreasing p_A

algorithm	feature set	tree depth	l_rate	e_threshold	p_A	99% *interval*
SDG with stumps	100 stumps	4	0.0001	20	73.48%	[0.66,0.81]
DT	original	unlimited	N.A	N.A	72.23%	[0.67,0.77]
DT	original	8	N.A	N.A	69.67%	[0.63,0.77]
DT	original	4	N.A	N.A	65.63%	[0.56,0.76]
SDG	original	N.A	0.0001	20	65.54%	[0.54,0.77]

Note that the "original" in the above table stands for ten feature types mentioned in the question.
In the tunning of the learning rate and threshold, the average accuracy for the pair I chose was 67.41% and the standard deviation is 0.06
The $\{\lambda_i, \epsilon_i\}$ pairs that have been tried are $\{\lambda_i = 0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1; \epsilon_i = 5, 10, 15, 20, 25\}$

Analysis:
The differences between performance of SGD with stumps and the one of DT with unlimited depth, the performance of DT with unlimited depth and the one of DT with depth of 8, and the performance of DT with depth of 4 and SGD are not statisitically significant. Although there is gap between the performances of the DT with dpeth of 8 and the one of the DT with depth of 4, they are still not statistically significantly different.
It is understandable that the normal SGD perform the worst and the performance of the Decision tree improves as the depth grows, since more features are considered. Yet, it's curious that the decision tree with unlimited depth did not overfit the data.
Theoratically speaking, the performances of SGD with stumps and normal Decision tree with unlimited should have had significant difference. However, it is not the case

in my experiment, although the SGD with stumps still have the highest performance anyhow. My theory to explain this weird phenomenon is that when extracting features we left out some important feature types that were used to generate the data set. Adding more features, such as, first 10 characters of both the first and last name, improves the performance of the SGD with stumps dramatically, while it does little to the DT with Unlimited depth and three of the folds even reported lower performace than the DTs with depth of 8, presumably because of overfitting.

Conclusion:
From the p_A, the SGD with stumps perform the best. The accuracy for the SGD with stumps could reach over 80% in some of the testing sets, while the other algorithms have way lower performance. This conclusion based on the assumption that the average of the accuracy on the train data set reveals the true performance and also that the trainning data and the test data come from the same set or sets with same destribution.

Tree display:

ID3 with unlimited depth, fold3
lastName0=m = 1
— firstName2=e = 1: +
— firstName2=e = 0
— — lastName1=o = 1: +
— — lastName1=o = 0
— — — firstName0=p = 1: -
— — — firstName0=p = 0
— — — — firstName0=r = 1: -
— — — — firstName0=r = 0
— — — — — firstName0=y = 1: -
— — — — — firstName0=y = 0
— — — — — — firstName1=u = 1: -
— — — — — — firstName1=u = 0
— — — — — — — lastName3=a = 1
— — — — — — — — firstName3=r = 1: +
— — — — — — — — firstName3=r = 0: -
— — — — — — — lastName3=a = 0: +
lastName0=m = 0
— lastName1=l = 1
— — firstName0=d = 1: -
— — firstName0=d = 0: +
— lastName1=l = 0
— — lastName2=l = 1
— — — firstName2=r = 1: -
— — — firstName2=r = 0
— — — — lastName4=n = 1: -

```
— — — — lastName4=n = 0
— — — — — firstName2=h = 1: -
— — — — — firstName2=h = 0: +
— — lastName2=l = 0
— — — lastName2=o = 1
— — — — firstName0=b = 1: -
— — — — firstName0=b = 0: +
— — — lastName2=o = 0
— — — — firstName3=f = 1: +
— — — — firstName3=f = 0
— — — — — lastName4=l = 1
— — — — — — firstName2=h = 1: -
— — — — — — firstName2=h = 0
— — — — — — — lastName0=l = 1: -
— — — — — — — lastName0=l = 0
— — — — — — — — lastName0=q = 1: -
— — — — — — — — lastName0=q = 0: +
— — — — — lastName4=l = 0
— — — — — — firstName1=o = 1
— — — — — — — lastName0=f = 1: -
— — — — — — — lastName0=f = 0
— — — — — — — — firstName2=e = 1: -
— — — — — — — — firstName2=e = 0
— — — — — — — — — firstName3=a = 1
— — — — — — — — — — lastName0=h = 1: +
— — — — — — — — — — lastName0=h = 0: -
— — — — — — — — — firstName3=a = 0
— — — — — — — — — — firstName2=n = 1: +
— — — — — — — — — — firstName2=n = 0
— — — — — — — — — — — lastName2=n = 1: +
— — — — — — — — — — — lastName2=n = 0
— — — — — — — — — — — — lastName1=a = 1: -
— — — — — — — — — — — — lastName1=a = 0
— — — — — — — — — — — — — firstName3=g = 1: -
— — — — — — — — — — — — — firstName3=g = 0: +
— — — — — — firstName1=o = 0
— — — — — — — lastName0=l = 1
— — — — — — — — firstName1=a = 1
— — — — — — — — — firstName0=d = 1: +
— — — — — — — — — firstName0=d = 0: -
— — — — — — — — firstName1=a = 0: +
— — — — — — — lastName0=l = 0
— — — — — — — — lastName3=m = 1
— — — — — — — — — firstName2=r = 1: -
— — — — — — — — — firstName2=r = 0: +
— — — — — — — — lastName3=m = 0
— — — — — — — — — firstName1=e = 1
— — — — — — — — — — firstName2=n = 1: +
— — — — — — — — — — firstName2=n = 0
— — — — — — — — — — — firstName2=o = 1
— — — — — — — — — — — — lastName0=b = 1: -
— — — — — — — — — — — — lastName0=b = 0: +
— — — — — — — — — — — firstName2=o = 0
— — — — — — — — — — — — lastName2=r = 1
— — — — — — — — — — — — — firstName0=m = 1: -
— — — — — — — — — — — — — firstName0=m = 0: +
— — — — — — — — — — — — lastName2=r = 0: -
— — — — — — — — — firstName1=e = 0
— — — — — — — — — — firstName0=t = 1
— — — — — — — — — — — lastName4=e = 1: -
— — — — — — — — — — — lastName4=e = 0
— — — — — — — — — — — — lastName0=s = 1: -
— — — — — — — — — — — — lastName0=s = 0: +
— — — — — — — — — — firstName0=t = 0
— — — — — — — — — — — firstName3=o = 1
— — — — — — — — — — — — firstName0=a = 1: +
— — — — — — — — — — — — firstName0=a = 0
— — — — — — — — — — — — — firstName0=m = 1: +
— — — — — — — — — — — — — firstName0=m = 0: -
— — — — — — — — — — — firstName3=o = 0
— — — — — — — — — — — — firstName4=o = 1
— — — — — — — — — — — — — firstName0=s = 1: -
— — — — — — — — — — — — — firstName0=s = 0: +
— — — — — — — — — — — — firstName4=o = 0
— — — — — — — — — — — — — lastName0=s = 1
— — — — — — — — — — — — — — firstName0=d = 1: +
— — — — — — — — — — — — — — firstName0=d = 0
— — — — — — — — — — — — — — — lastName4=h = 1
— — — — — — — — — — — — — — — — firstName0=s = 1: -
— — — — — — — — — — — — — — — — firstName0=s = 0: +
— — — — — — — — — — — — — — — lastName4=h = 0: -
— — — — — — — — — — — — — lastName0=s = 0
— — — — — — — — — — — — — — lastName3=l = 1
— — — — — — — — — — — — — — — firstName0=d = 1: -
— — — — — — — — — — — — — — — firstName0=d = 0: +
— — — — — — — — — — — — — — lastName3=l = 0: -
Correctly Classified Instances 35          76.087 %
Incorrectly Classified Instances 11          23.913 %
Kappa statistic 0.5199
Mean absolute error 0.2391
```

Root mean squared error 0.489
Relative absolute error 48.1752 %
Root relative squared error 98.1732 %
Total Number of Instances 46

ID3 with depth of 8,fold1
firstName3=f = 1: +
firstName3=f = 0
— lastName0=c = 1: -
— lastName0=c = 0
— — lastName4=l = 1
— — — lastName0=q = 1: -
— — — lastName0=q = 0: +
— — lastName4=l = 0
— — — firstName0=r = 1
— — — — firstName1=o = 1: +
— — — — firstName1=o = 0
— — — — — firstName1=a = 1: +
— — — — — firstName1=a = 0
— — — — — — firstName1=e = 1: +
— — — — — — firstName1=e = 0: -
— — — firstName0=r = 0
— — — — lastName0=m = 1
— — — — — firstName2=n = 1: -
— — — — — firstName2=n = 0
— — — — — — firstName0=p = 1: -
— — — — — — firstName0=p = 0
— — — — — — — lastName2=t = 1
— — — — — — — — firstName0=t = 1: +
— — — — — — — — firstName0=t = 0: -
— — — — — — — lastName2=t = 0: +
— — — — lastName0=m = 0
— — — — — lastName0=l = 1
— — — — — — firstName1=a = 1
— — — — — — — firstName0=d = 1: +
— — — — — — — firstName0=d = 0: -
— — — — — — firstName1=a = 0: +
— — — — — lastName0=l = 0
— — — — — — lastName3=m = 1
— — — — — — — firstName2=r = 1: -
— — — — — — — firstName2=r = 0: +
— — — — — — lastName3=m = 0
— — — — — — — lastName2=l = 1
— — — — — — — — firstName2=r = 1: -
— — — — — — — — firstName2=r = 0: +

— — — — — — — lastName2=l = 0
— — — — — — — — lastName3=l = 1: +
— — — — — — — — lastName3=l = 0: -

Correctly Classified Instances 48 73.8462 %
Incorrectly Classified Instances 17 26.1538 %
Kappa statistic 0.4785
Mean absolute error 0.3371
Root mean squared error 0.4722
Relative absolute error 67.4285 %
Root relative squared error 94.4514 %
Total Number of Instances 65

ID3 with depth of 4,fold4
lastName2=l = 1
— firstName2=r = 1: -
— firstName2=r = 0
— — firstName2=m = 1: -
— — firstName2=m = 0: +
lastName2=l = 0
— lastName2=o = 1
— — firstName0=d = 1: -
— — firstName0=d = 0
— — — firstName2=l = 1: -
— — — firstName2=l = 0: +
— lastName2=o = 0
— — firstName3=f = 1: +
— — firstName3=f = 0
— — — lastName0=m = 1
— — — — firstName0=n = 1: -
— — — — firstName0=n = 0: +
— — — lastName0=m = 0
— — — — lastName1=l = 1: +
— — — — lastName1=l = 0: -

Correctly Classified Instances 48 72.7273 %
Incorrectly Classified Instances 18 27.2727 %
Kappa statistic 0.409
Mean absolute error 0.3812
Root mean squared error 0.4582
Relative absolute error 78.7652 %
Root relative squared error 93.1911 %
Total Number of Instances 66