Parashar Parikh

HW#1

Pxp190016

Note: I have implemented Decision tree using Entropy heuristic and Variance heuristic. I have also implemented Pre-pruning (depth based). But my algorithm took so long for post pruning, so I have not included post pruning in the documentation.

Outputs:

Data-set	Entropy heuristic (%)	Variance heuristic (%)	Pre-pruning (%)		Randomforest (%)
			Entropy	Variance	
test_c300_d100	57.65	57.65	57.78	57.78	57.28
test_c300_d1000	65.06	65.03	64.03	64.03	64.63
test_c300_d5000	70.71	69.86	70.75	70.01	70.86
test_c500_d100	64.32	60.30	63.33	63.33	60.30
test_c500_d1000	68.68	66.73	65.78	64.55	71.83
test_c500_d5000	74.67	73.69	75.78	72.89	77.68
test_c1000_d100	68.84	68.84	69.56	69.56	83.91
test_c1000_d1000	79.63	79.58	78.88	78.96	89.89
test_c1000_d5000	83.85	83.26	83.87	83.84	93.41
test_c1500_d100	81.40	82.91	83.33	83.01	97.98
test_c1500_d1000	89.69	88.99	90.01	89.86	98.54
test_c1500_d5000	94.41	93.97	95.44	95.43	99.17
test_c1800_d100	90.95	90.95	89.98	90.10	97.98
test_c1800_d1000	96.64	95.79	97.77	96.45	99.69
test_c1800_d5000	98.47	97.65	98.87	97.78	99.89

1. Which impurity heuristic (Entropy/Variance) yields the best classification accuracy? How does increasing the number of examples and/or the number of clauses impact the (accuracy of the) two impurity heuristics. Explain your answer.

   I have applied both Entropy heuristic and Variance heuristic. In most of the case I got higher accuracy using Entropy heuristic. But in some case, I got same accuracy or higher in Variance compare to Entropy.

   These are screen shots showing these results.

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1500_d1000.csv Data/valid_c1500_d1000.csv Data/test_c1500_d1000.csv Varience 0
accuracy is  88.99449724862431
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1500_d5000.csv Data/valid_c1500_d5000.csv Data/test_c1500_d5000.csv Varience 0
accuracy is  93.97939793979398
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c500_d5000.csv Data/valid_c500_d5000.csv Data/test_c500_d5000.csv Entropy 0
accuracy is  74.66746674667468
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1500_d5000.csv Data/valid_c1500_d5000.csv Data/test_c1500_d5000.csv Varience 0
accuracy is  93.97939793979398
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d100.csv Data/valid_c1800_d100.csv Data/test_c1800_d100.csv Entropy 0
accuracy is  90.95477386934674
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d100.csv Data/valid_c1800_d100.csv Data/test_c1800_d100.csv Varience 0
accuracy is  90.95477386934674
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d1000.csv Data/valid_c1800_d1000.csv Data/test_c1800_d1000.csv Entropy 0
accuracy is  96.64832416208104
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d1000.csv Data/valid_c1800_d1000.csv Data/test_c1800_d1000.csv Varience 0
accuracy is  95.79789894947474
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d5000.csv Data/valid_c1800_d5000.csv Data/test_c1800_d5000.csv Entropy 0
accuracy is  98.47984798479848
```

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d5000.csv Data/valid_c1800_d5000.csv Data/test_c1800_d5000.csv Varience 0
accuracy is  97.65976597659765
```

2. Which overfitting avoidance method (reduced error pruning/ depth-based pruning) yields the best accuracy? Again, how does increasing the number of examples and/or the number of clauses impact the (accuracy of the) two overfitting avoidance methods. Explain your answer.

Since I used only one overfitting method depth-based pruning, which gave more accuracy sometimes. But complexity wise depth-based pruning is faster than reduced error pruning.
As examples increased accuracy increases. Because when we prune tree based on depth, we put general target as target value. So as number of examples increases, we have more pool of target to generalized. (general target: count no of 0's and 1's and then put 0 or 1 depending on higher count).

Pruning on test_c300_d100 using depth-based pruning and Entropy heuristic.

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py Data/train_c300_d100.csv Data/valid_c300_d100.csv
 Data/test_c300_d100.csv Entropy preprune
Depth= 100  accuracy is=  59.2964824120603
Depth= 5  accuracy is=  24.623115577889447
Depth= 10  accuracy is=  59.2964824120603
Depth= 15  accuracy is=  59.2964824120603
Depth= 50  accuracy is=  59.2964824120603
Depth= 20  accuracy is=  59.2964824120603
final accuracy is  57.78894472361809
```

For pruning I have checked all the level accuracy using validation set as showed in above picture. Then select depth (10) which is having more accuracy and used this depth as hyperparameter for testing set.

3. Are random forests much better in terms of classification accuracy than your decision tree learners? Why? Explain your answer.

Partially yes. Because as no of example increases accuracy increases in random forest compare to my decision tree.

For small data set I got little higher accuracy compare to random forest.

My implementation
Test_c300_d100

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c300_d100.csv Data/valid_c300_d100.csv Data/test_c300_d100.csv Varience 0
accuracy is  57.78894472361809
```

Random forest
Test_c300_d100

```
In [34]: import pandas as pd
         import numpy as np
         from sklearn.ensemble import RandomForestClassifier

         import os
         cwd = os.getcwd()
         #cwd
         #c300_d100
         train_c300_d100=pd.read_csv(r'C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1\Data\train_c300_d100.csv')
         #train_c300_d100
         valid_c300_d100=pd.read_csv(r'C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1\Data\valid_c300_d100.csv')
         #valid_c300_d100
         test_c300_d100=pd.read_csv(r'C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1\Data\test_c300_d100.csv')
         train_c300_d100
         valid_c300_d100
         train_c300_d100_target=train_c300_d100.iloc[:,-1]
         train_c300_d100.drop(train_c300_d100.columns[-1], axis=1, inplace=True)
         train_c300_d100.shape
         train_c300_d100_target.shape
         rf = RandomForestClassifier()
         rf.fit(train_c300_d100,train_c300_d100_target)

         #valid_c300_d100_target=valid_c300_d100.iloc[:,-1]
         test_c300_d100_target=test_c300_d100.iloc[:,-1]
         #valid_c300_d100.drop(valid_c300_d100.columns[-1], axis=1, inplace=True)
         test_c300_d100.drop(test_c300_d100.columns[-1], axis=1, inplace=True)
         #valid_c300_d100_target.shape
         #valid_c300_d100.shape
         #test_c300_d100.shape
         test_c300_d100_target.shape
         #rf.score(valid_c300_d100,valid_c300_d100_target)
         #rf.score(train_c300_d100,train_c300_d100_target)
         score_c300_d100=rf.score(test_c300_d100,test_c300_d100_target)
         score_c300_d100

Out[34]: 0.5728643216080402
```

But for a greater number of inputs my tree got less accuracy compare to random forest.
My implementation

Test_c1800_d5000

```
(C:\ProgramData\Anaconda3) C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1>python Decision_tree.py
 Data/train_c1800_d5000.csv Data/valid_c1800_d5000.csv Data/test_c1800_d5000.csv Entropy 0
accuracy is  98.47984798479848
```

Random forest
Test_c1800_d5000

```
In [37]: #c1800_d5000
train_c1800_d5000=pd.read_csv(r'C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1\Data\train_c1800_d5000.csv',nrows=10
#train_c1800_d5000
valid_c1800_d5000=pd.read_csv(r'C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1\Data\valid_c1800_d5000.csv',nrows=10
#valid_c1800_d5000
test_c1800_d5000=pd.read_csv(r'C:\Users\Parashar Parikh\Desktop\UTD Sem1\Machie Learning\hw1\Data\test_c1800_d5000.csv',nrows=1000
#test_c1800_d5000


train_c1800_d5000_target=train_c1800_d5000.iloc[:,-1]
train_c1800_d5000.drop(train_c1800_d5000.columns[-1], axis=1, inplace=True)
train_c1800_d5000.shape
train_c1800_d5000_target.shape

rf = RandomForestClassifier()
rf.fit(train_c1800_d5000,train_c1800_d5000_target)

valid_c1800_d5000_target=valid_c1800_d5000.iloc[:,-1]
test_c1800_d5000_target=test_c1800_d5000.iloc[:,-1]
valid_c1800_d5000.drop(valid_c1800_d5000.columns[-1], axis=1, inplace=True)
test_c1800_d5000.drop(test_c1800_d5000.columns[-1], axis=1, inplace=True)

valid_c1800_d5000_target.shape

valid_c1800_d5000.shape

test_c1800_d5000.shape

test_c1800_d5000_target.shape

rf.score(valid_c1800_d5000,valid_c1800_d5000_target)
rf.score(train_c1800_d5000,train_c1800_d5000_target)
score_c1800_d5000=rf.score(test_c1800_d5000,test_c1800_d5000_target)
score_c1800_d5000

Out[37]: 0.998999899989999
```

Random forest generates tree randomly. And give output as average of all the tress. Every time you run random forest algorithm accuracy may vary. But in our implementation accuracy remains same. So, some time Random forest gives more accuracy and some time less. But in general, Random forest gives more accuracy compare to my algorithm.