Main

September 27, 2019

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import cross_val_score,train_test_split
        from sklearn.metrics import accuracy_score
        import matplotlib.pyplot as plt
        import timeit
        import copy
        from MyDT import *
```

```
In [37]: def learn(train_data, impurity_measure='entropy', prune=False, pruning_sampels_indices
             if prune:
                 train_sampels_indices = np.delete(range(train_data.shape[0]-1),pruning_sampels
                 fulltree = grow_tree(train_data[train_sampels_indices, :], impurity_measure)
                 out_tree = copy.deepcopy(prune_tree(copy.deepcopy(fulltree), train_data[prunin
             else:
                 out_tree = grow_tree(train_data, impurity_measure)
             return out_tree
```

```
In [5]: def predict(x, tree):
            prediction = Tree_Predict(tree, x)
            return prediction
```

```
In [103]: # import data
          X = pd.read_csv('abalone.data',header=None)
          X = np.array(X)
          # data preprocessing (make the first attribute continious like others)
          X[:,0] = np.where(X[:,0]=='M', 0, X[:,0])
          X[:,0] = np.where(X[:,0]=='F', 1, X[:,0])
          X[:,0] = np.where(X[:,0]=='I', 2, X[:,0])
          np.random.shuffle(X)

          #split train and test sets
          Train_data = X[0:round(X.shape[0]*.7),:]
          Test_data = X[round(X.shape[0]*.7)-1:,:]
```

```
In [107]: start = timeit.default_timer() #  record time
          # learning
          the_learnt_tree = learn(Train_data, 'entropy')
```

```
         stop = timeit.default_timer() #  record time
         print ("Time elapsed for learning:", stop - start)
         # checking the accuracy
         print ("Accuracy :",accuracy_for_a_set(the_learnt_tree, Test_data))
         # visualization
         # tree_visualization(the_learnt_tree)  #uncomment if need visualization
```

```
Time elapsed for learning: 5.963340000000244
Accuracy : 0.17862838915470494
```

```
In [108]: Pruning_sampels_indices = np.random.randint(Train_data.shape[0], size=round(Train_da

          start = timeit.default_timer() #  record time
          # learning with pruning
          the_learnt_tree_pruned = learn(Train_data, 'entropy', True, Pruning_sampels_indices)
          stop = timeit.default_timer() #  record time
          print ("Time elapsed for learning:", stop - start)
          # checking the accuracy
          print ("Accuracy :",accuracy_for_a_set(the_learnt_tree_pruned, Test_data))
          # visualization
          # tree_visualization(the_learnt_tree_pruned)  #uncomment if need visualization
```

```
Time elapsed for learning: 4.125450800000181
Accuracy : 0.19856459330143542
```

```
In [106]: from sklearn.tree import DecisionTreeClassifier
          from sklearn.metrics import accuracy_score
          from sklearn import tree
          from sklearn.externals.six import StringIO
          import pydotplus
          from IPython.display import Image
          from graphviz import Digraph


          iny = np.array(Train_data[:,8])
          iny = iny.astype('int')
          inX = np.array(Train_data[:,0:7])
          clf = tree.DecisionTreeClassifier(random_state=0) # optional>> , max_depth=10
          start = timeit.default_timer() #  record time
          # learning
          clf.fit(inX, iny)
          stop = timeit.default_timer() #  record time
          print ("Time elapsed for learning:", stop - start)

          predicted_y = clf.predict(Test_data[:,0:7])
          print(accuracy_score(Test_data[:,8].astype('int'),predicted_y))
```

```
    # visualization
    # dot_data = StringIO()
    # mytree = tree.export_graphviz(clf, out_file=dot_data)
    # graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    # Image(graph.create_png())
```

```
Time elapsed for learning: 0.020507899999756773
0.19218500797448165
```