Machine Learning

Assignment #3 DT

Submitted by:	Sec.	B.N.
Alaa Allah Essam Abdrabo	1	13

- This assignment is composed of 2 Problems.
- Part1 :
  - Recommend another type of trees than ID3 that would build a less deep tree than ID3, Assume that you are building a complete ID3 tree. Justify your choice
    - types of Decision trees
      1. ID3 (Iterative Dichotomiser 3)(It is not an ideal algorithm as it generally overfits the data and on continuous variables, splitting the data can be time consuming.)
      2. C4.5 (successor of ID3)(The splitting is done based on the normalized information gain and the feature having the highest information gain makes the decision. Unlike ID3, it can handle both continuous and discrete attributes very efficiently and after building a tree, it undergoes pruning by removing all the branches having low importance.)
      3. CART (Classification And Regression Tree)
      4. CHAID (CHi-squared Automatic Interaction Detector). ...
      5. MARS(Multivariate adaptive regression splines): extends decision trees to handle numerical data better.
    - CHAID (Chi-Square automatic interaction detector)

    this method could give a less deep tree as uses a statistical rule to stop tree growth called the Chi-Square test. The Chi-Square test qualifies the values observed, to those in theory. Values which are far off from theory, independent, are cut off and the tree stops at that branch.

    - a techinque that can reduce size of tree :

Pruning is the process of reducing the size of the tree by turning some branch nodes into leaf nodes, and removing the leaf nodes under the original branch.

- pre-prunning: Stop splitting the current node if it does not improve the entropy by at least some pre-set(threshold) value. Stop partitioning if the number of datapoints are less then some preset(Threshold) values. Restricting the depth of the tree to some pre-set(Threshold) value.
- Post-prunning: It can be done by first allowing the tree to grow to its full potential and then prunning the tree at each level after calculating the cross-validation accuracy at each level.

• Part2 :

- Implement the CART decision tree learning algorithm
- Required functions

Entropy

```
def entropy(x):

    elements,counts = np.unique(x,return_counts = True)     # returns the number of times each unique item appears in col
    #print(counts)
    entropy = np.sum([(-counts[i]/np.sum(counts))*np.log2(counts[i]/np.sum(counts)) for i in range(len(elements))])
    #print(entropy)
    return entropy
```

Information gain

```
# root and different nodes are chosen based on the  higher information gain

# information gain will be calculated for all attributes in all features

def IG(attribute,y):

  c1= sum(attribute)
  c2= attribute.shape[0] - c1
  return entropy(y)-c1/(c1+c2)*entropy(y[attribute])-c2/(c1+c2)*entropy(y[-attribute])
```

Every feature in data has diffierent attributes that are but in collections to calculate information gain

```
def categorical_options(x):

  x = x.unique()

  combinations = []
  for i in range(0, len(x)+1):
      for subset in itertools.combinations(x, i):
          subset = list(subset)
          combinations.append(subset)

  return combinations[1:-1]
```

splitting is based on highest information gain

```
def split(x, y):


  split_value = []
  ig = []
  options = categorical_options(x)

  for val in options:
    attribute =   x < val
    val_ig = IG(attribute, y)
    ig.append(val_ig)
    split_value.append(val)
    max_ig = max(ig)
    ig_index = ig.index(max_ig)
    best_split = split_value[ig_index]
    return(max_ig,best_split)
```