Wei Tang
CSE353

Psuedocode For ID3

Entropy Pseudocode:

```
get_entropy(data)
        pos,neg
        for entry in data:
                if data[target_attribute] == target_attribute_value:
                        pos++
                else:
                        neg++
        return -pos/len(data)*log2(pos/len(data))-neg/len(data)*log2(neg/len(data))
```

Choose Attribute with Best Entrypy:

```
get_best_attribute(data,attributes)
        info_gain_array []
        for attribute in attributes:
                info_gain_array.append(get_infomation_gain(data,attribtue))

        best_attribute = attributes[info_gain_array.index(max(info_gain_array))]

        if best_attribute is discrete value
                return best_attribute ,False, -1
        else
                return best_arrtibute, True, max(infomation_gain)
```

Information Gain:

```
get_information_gain(data,attributes):
        attribute is continuous:
                sort data by attribute
                generate thresholdlist{}
                for threshold in thresholdList
                        left_data,right_data =
spread_data_by_threshold(data,attribtues,threshold)

                        for each entry in data:
```

```
                        if entry[attribute] <= threshold:
                                pos++
                        else
                                neg++

                threshold_info_gain = get_entropy(data) -
pos/len(data)*get_entropy(left_data)-neg/len(data)*getentrypy(right_data)
                thresholdlist{threshold} = threshold_info_gain

        return max(thresholdlist,key = get),threshold


    attribtue not continuous:
        for each value in attribute:
            calculate the frequency of its apparence

        for value in frequence:
            seperate data by value
            entrypy -=value_frequence[value]/len(data)*get_entrypy(sub_data)

        return get_entrypy(data) - entrypy


Make Tree:

maketree(data,attributes):
    if no data left or run out of attribute:
        return majority of the classification
    if all data has the same classification:
        return the classification


    best_attribute = get_best_attribtue(data,attribute)

    tree = {best_attribute:{}}

    if best_attribtue is continuous:
        divide the tree in half with threshold
        left_sub_tree = maketree(left_data,attribtue without best_attribute)
        right_sub_tree = maketree(right_data,attribtue without best_attribute)

    else:
        for value in attribute:
            sub_data = divide data by value
```

```
                    sub_tree = maketree(sub_data,attribtue withour best_attribtue)
        return


Test Data:

testing(data,tree):
        for entry in dta:
                while not reach leaf :
                        root_index = tree.key()[0]
                        if root_index is continuous:
                                if tree[root_index][threshold] == target_value:
                                        reach leaf
                                threhold = tree.value[root_index]
                                if entry[attribute] < threshold
                                        goto leftsub tree
                                else:
                                        goto rightsub tree
                        else:
                                if tree[root_index][entry[root_index]] == target_value
                                        reah leaf
                                else:
                                        goto tree[root_index][entry[toot_index]]
```