7. DecisionTree

March 28, 2022

[1]:
```python
import pandas as pd
import numpy as np
```

[2]:
```python
weather_features = ['outlook','temperature','humidity','windy','play']
weather_data = [
    ['overcast','hot','high','false','yes'],
    ['overcast','cool','normal','true','yes'],
    ['overcast','mild','high','true','yes'],
    ['overcast','hot','normal','false','yes'],
    ['rainy','mild','high','false','yes'],
    ['rainy','cool','normal','false','yes'],
    ['rainy','cool','normal','true','no'],
    ['rainy','mild','normal','false','yes'],
    ['rainy','mild','high','true','no'],
    ['sunny','hot','high','false','no'],
    ['sunny','hot','high','true','no'],
    ['sunny','mild','high','false','no'],
    ['sunny','cool','normal','false','yes'],
    ['sunny','mild','normal','true','yes']
    ]

train_data = pd.DataFrame(weather_data, columns = weather_features)
train_data=train_data.values.tolist()
train_data
```

[2]:
```
[['overcast', 'hot', 'high', 'false', 'yes'],
 ['overcast', 'cool', 'normal', 'true', 'yes'],
 ['overcast', 'mild', 'high', 'true', 'yes'],
 ['overcast', 'hot', 'normal', 'false', 'yes'],
 ['rainy', 'mild', 'high', 'false', 'yes'],
 ['rainy', 'cool', 'normal', 'false', 'yes'],
 ['rainy', 'cool', 'normal', 'true', 'no'],
 ['rainy', 'mild', 'normal', 'false', 'yes'],
 ['rainy', 'mild', 'high', 'true', 'no'],
 ['sunny', 'hot', 'high', 'false', 'no'],
 ['sunny', 'hot', 'high', 'true', 'no'],
 ['sunny', 'mild', 'high', 'false', 'no'],
```

```
 ['sunny', 'cool', 'normal', 'false', 'yes'],
 ['sunny', 'mild', 'normal', 'true', 'yes']]
```

[3]:
```
#train_data = pd.read_csv("weather.csv",header=0)
#features=train_data.columns.values.tolist()
#train_data = train_data.values.tolist()
#features = train_data[0:4]
#features
```

[4]:
```
from math import log
def entropy(dataset):
    numEntries=len(dataset)  # number of samples in dataset
    # empty dictionary to maintain count of each label in dataset
    labelCounts={}

    for eachRow in dataset:
        #print(eachRow)
        rowLabel=eachRow[-1]
        #print(rowLabel)
        if rowLabel not in labelCounts.keys():
            labelCounts[rowLabel]=1
        else:
            labelCounts[rowLabel]+=1
    print(labelCounts)

    ent=0.0
    for key in labelCounts:
        prob=float(labelCounts[key])/numEntries # probability of current label
        ent -= prob*log(prob,2)                 # -p1logp1-p2logp2
    return ent

#e=entropy(train_data)
#e
```

[5]:
```
def splitDataSet(dataset,feature,value):
    retdataset=[]
    for featVec in dataset:
        if featVec[feature]==value:
            reducedFeatVec=featVec[:feature]
            reducedFeatVec.extend(featVec[feature+1:])
            retdataset.append(reducedFeatVec)
    return retdataset

def chooseBestFeatureToSplit(dataset):
    numFeatures = len(dataset[0])

    baseEntropy=entropy(train_data)
```

```
    print("Base Entropy:",baseEntropy)
    bestInfoGain=0.0
    bestFeature=-1

    print("Range of numFeatures: ",numFeatures)
    for i in range(numFeatures-1):
        print("Iteration:",i)
        print("------------Feature:",features[i],"----------------")
        # extract columns one-by-one
        featList=[sample[i] for sample in dataset]
        # set of unique values in current column
        uniqueVals=set(featList)

        newEntropy=0.0

        for value in uniqueVals:
            print("----Value =",value,"--------")
            subdataset=splitDataSet(dataset,i,value)
            #subdataset=np.array(subdataset)
            prob=len(subdataset)/float(len(dataset))
            e=entropy(subdataset)
            print("Entropy=",e)
            newEntropy+=prob*e

        infoGain=baseEntropy-newEntropy
        print("InfoGain=",infoGain)

        if(infoGain > bestInfoGain):
            bestInfoGain = infoGain
            bestValue = value
            bestFeature = i

        print("Best Feature: ", bestFeature)

    return bestFeature
#index=chooseBestFeatureToSplit(train_data)
#bf=features[index]
#print()
#print("Best feature to split is: Feature -> ",bf)
#print("Best value to split is: Value = ",value)
```

[6]:
```
def majorityCnt(classList):
    classCount={}
    for vote in classList:
        if vote not in classCount.keys():
            classCount[vote]=0
        else:
```

```python
            classCount[vote]+=1
    sortedClassCount=sorted(classCount.iteritems(),key=operator.
↪itemgetter(1),reverse=True)
    return sortedClassCount[0][0]

def createTree(dataset,features):

    # extract last column (column with labels)
    classList = [sample[-1] for sample in dataset]
    print("-----------------Class List----------------------")
    print(classList)

    ## Stopping conditions
    ## If the (sub)dataset has all labels belonging to same class
    if classList.count(classList[0])==len(classList):
        print("-------Exit current path (same class)--------")
        print()
        return classList[0]
    ## If only one feature is left in the (sub)dataset
    if len(dataset[0])==1:
        print("-------Exit current path (one feature)--------")
        print()
        return majorityCnt(classList)

    bestFeat = chooseBestFeatureToSplit(dataset)
    print("Final Best Feature: ",bestFeat)
    bestFeatLabel = features[bestFeat]
    print("###############################################################")
    print("BestFeature:",bestFeatLabel)
    print("###############################################################")
    print()

    # Store the tree in a dictionary
    # the key of dictionary will be best feature,
    # and value can also be a dictionary
    myTree = {bestFeatLabel: {}}
    # we delete the best feature from list of features
    del(features[bestFeat])
    print("Remaining Features: ", features)

    # extract values from best feature column
    featValues = [sample[bestFeat] for sample in dataset]
    # identify unique values from extracted column
    uniqueVals = set(featValues)

    for value in uniqueVals:
        print("Feature=",bestFeatLabel,", Branch=",value)
```

```
        # copy list of features to a new modifiable list
        subLabels = features[:]
        myTree[bestFeatLabel][value] =␣
↪createTree(splitDataSet(dataset,bestFeat,value),subLabels)

    return myTree

#createTree(train_data,features)
```

[7]:
```
def classify(inputTree,featLabels,testVec):
    firstStr = list(inputTree.keys())[0]
    featIndex = featLabels.index(firstStr)
    secondDict = inputTree[firstStr]
    for key in secondDict.keys():
        if testVec[featIndex]==key:
            if type(secondDict[key]).__name__=='dict':
                classLabel = classify(secondDict[key],featLabels,testVec)
        else:
            classLabel = secondDict[key]
    return classLabel

features=['outlook', 'temperature', 'humidity', 'windy']
mytree = createTree(train_data,features)

features=['outlook', 'temperature', 'humidity', 'windy']
x=classify(mytree,features,['sunny','hot','low','false'])
print("The predicted label is:",x)
```

```
-----------------Class List----------------------
['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'no',
'yes', 'yes']
{'yes': 9, 'no': 5}
Base Entropy: 0.9402859586706309
Range of numFeatures:  5
Iteration: 0
------------Feature: outlook ----------------
----Value = rainy --------
{'yes': 3, 'no': 2}
Entropy= 0.9709505944546686
----Value = overcast --------
{'yes': 4}
Entropy= 0.0
----Value = sunny --------
{'no': 3, 'yes': 2}
Entropy= 0.9709505944546686
InfoGain= 0.2467498197744391
Best Feature:  0
```

```
Iteration: 1
------------Feature: temperature ----------------
----Value = hot --------
{'yes': 2, 'no': 2}
Entropy= 1.0
----Value = cool --------
{'yes': 3, 'no': 1}
Entropy= 0.8112781244591328
----Value = mild --------
{'yes': 4, 'no': 2}
Entropy= 0.9182958340544896
InfoGain= 0.029222565658954647
Best Feature:  0
Iteration: 2
------------Feature: humidity ----------------
----Value = high --------
{'yes': 3, 'no': 4}
Entropy= 0.9852281360342516
----Value = normal --------
{'yes': 6, 'no': 1}
Entropy= 0.5916727785823275
InfoGain= 0.15183550136234136
Best Feature:  0
Iteration: 3
------------Feature: windy ----------------
----Value = true --------
{'yes': 3, 'no': 3}
Entropy= 1.0
----Value = false --------
{'yes': 6, 'no': 2}
Entropy= 0.8112781244591328
InfoGain= 0.04812703040826927
Best Feature:  0
Final Best Feature:  0
################################################################
BestFeature: outlook
################################################################

Remaining Features:  ['temperature', 'humidity', 'windy']
Feature= outlook , Branch= rainy
-----------------Class List----------------------
['yes', 'yes', 'no', 'yes', 'no']
{'yes': 9, 'no': 5}
Base Entropy: 0.9402859586706309
Range of numFeatures:  4
Iteration: 0
------------Feature: temperature ----------------
----Value = cool --------
```

```
{'yes': 1, 'no': 1}
Entropy= 1.0
----Value = mild --------
{'yes': 2, 'no': 1}
Entropy= 0.9182958340544896
InfoGain= -0.010691541762062773
Best Feature:  -1
Iteration: 1
------------Feature: humidity ----------------
----Value = high --------
{'yes': 1, 'no': 1}
Entropy= 1.0
----Value = normal --------
{'yes': 2, 'no': 1}
Entropy= 0.9182958340544896
InfoGain= -0.010691541762062773
Best Feature:  -1
Iteration: 2
------------Feature: windy ----------------
----Value = true --------
{'no': 2}
Entropy= 0.0
----Value = false --------
{'yes': 3}
Entropy= 0.0
InfoGain= 0.9402859586706309
Best Feature:  2
Final Best Feature:  2
###############################################################
BestFeature: windy
###############################################################

Remaining Features:  ['temperature', 'humidity']
Feature= windy , Branch= true
-----------------Class List----------------------
['no', 'no']
-------Exit current path (same class)--------

Feature= windy , Branch= false
-----------------Class List----------------------
['yes', 'yes', 'yes']
-------Exit current path (same class)--------

Feature= outlook , Branch= overcast
-----------------Class List----------------------
['yes', 'yes', 'yes', 'yes']
-------Exit current path (same class)--------
```

```
Feature= outlook , Branch= sunny
-----------------Class List----------------------
['no', 'no', 'no', 'yes', 'yes']
{'yes': 9, 'no': 5}
Base Entropy: 0.9402859586706309
Range of numFeatures:  4
Iteration: 0
------------Feature: temperature ----------------
----Value = hot --------
{'no': 2}
Entropy= 0.0
----Value = cool --------
{'yes': 1}
Entropy= 0.0
----Value = mild --------
{'no': 1, 'yes': 1}
Entropy= 1.0
InfoGain= 0.5402859586706309
Best Feature:  0
Iteration: 1
------------Feature: humidity ----------------
----Value = high --------
{'no': 3}
Entropy= 0.0
----Value = normal --------
{'yes': 2}
Entropy= 0.0
InfoGain= 0.9402859586706309
Best Feature:  1
Iteration: 2
------------Feature: windy ----------------
----Value = true --------
{'no': 1, 'yes': 1}
Entropy= 1.0
----Value = false --------
{'no': 2, 'yes': 1}
Entropy= 0.9182958340544896
InfoGain= -0.010691541762062773
Best Feature:  1
Final Best Feature:  1
###############################################################
BestFeature: humidity
###############################################################

Remaining Features:  ['temperature', 'windy']
Feature= humidity , Branch= high
-----------------Class List----------------------
['no', 'no', 'no']
```

```
-------Exit current path (same class)--------

Feature= humidity , Branch= normal
-----------------Class List----------------------
['yes', 'yes']
-------Exit current path (same class)--------

The predicted label is: yes
```

```
[ ]:
```