3. Decision Trees

Prep

이번 시간엔 decision tree를 구현해봅니다. 먼저 필요한 모듈을 import하여주고, Google Drive도 마운트 합니다.

1 cell hidden

3.1. Computing Entorpy and Information Gain

9 cells hidden

3.2. Splitting a node

11 cells hidden

3.3 Putting Things Together - Tree Building via Recursions

4 cells hidden

3.4. Make Predictions with Our Decision Tree

4 cells hidden

3.5. Application to a real dataset

2 cells hidden

3.6. Visualize Decision Trees

이번 절에서는 학습된 decision tree를 화면에 그려보면서, 간단히 -- 하지만 코드는 쉽지 않습니다 :(-- Matplotlib이라는 라이브러리를 사용해보는 시간을 갖도록 하겠습니다.

다음의 code snippet은 matplotlib 라이브러리를 사용하여 간단히 화살표와 노드를 나타내는 도형을 그려봅니다. 이어지는 섹션에서 부품처럼 사용하기 위한 추상화(abstraction) 과정입니다. 즉, `plotNode()` 함수가 palette와 유사한 형태로 정의된 `decisionNode`, `leafNode`, `arrow_args` 등을 instantiate 할 수 있도록 구현하고 있습니다.

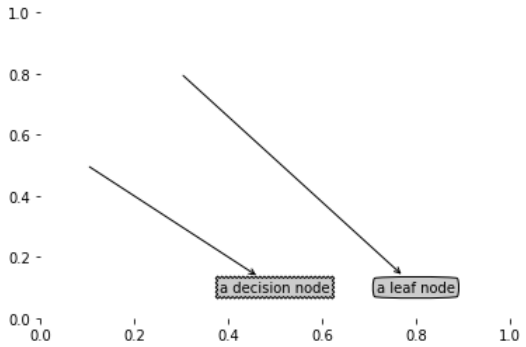
```
import matplotlib.pyplot as plt

# shape palette
decisionNode = dict(boxstyle="sawtooth", fc="0.8")
leafNode = dict(boxstyle="round4", fc="0.8")
arrow_args = dict(arrowstyle="<-")

def plotNode(nodeTxt, centerPt, parentPt, nodeType):
  createPlot.ax1.annotate(nodeTxt, xy=parentPt, xycoords='axes fraction',
                          xytext=centerPt, textcoords='axes fraction',
                          va="center", ha="center", bbox=nodeType, arrowprops=arrow_args)

def createPlot():
  fig = plt.figure(1, facecolor='white')
  fig.clf()
  createPlot.ax1 = plt.subplot(111, frameon=False)
  plotNode('a decision node', (0.5, 0.1), (0.1, 0.5), decisionNode)
  plotNode('a leaf node', (0.8, 0.1), (0.3, 0.8), leafNode)
  plt.show()

createPlot()
```



Identifying the number of leaves in a tree & the depth of the tree

다음은 학습된 tree를 받았을 때, tree의 속성을 파악하기 위해 구현하는 utility 함수들 입니다.

```
def getNumLeafs(myTree):
  numLeafs = 0
  firstStr = list(myTree)[0]
  secondDict = myTree[firstStr]
  for key in secondDict.keys():
    if type(secondDict[key]).__name__=='dict':
      numLeafs += getNumLeafs(secondDict[key])
    else:
      numLeafs += 1
  return numLeafs

def getTreeDepth(myTree):
  maxDepth = 0
  firstStr = list(myTree)[0]
  secondDict = myTree[firstStr]
  for key in secondDict.keys():
    if type(secondDict[key]).__name__=='dict':
      thisDepth = 1 + getTreeDepth(secondDict[key])
    else:
      thisDepth = 1
    if thisDepth > maxDepth:
      maxDepth = thisDepth
  return maxDepth
```

간단히 데모를 해보면서 코드가 잘 짜여졌는지 확인해 봅니다. 먼저, 가상의 트리를 만드는 함수 `retrieveTree()`를 만들어서 쉽게 트리 생성이 가능하도록 합니다.

`retrieveTree()`완성되었으면, 바로 위에서 작성한 `getNumLeafs()`, `getTreeDepth()`를 테스트해보도록 합니다.

```
# Create a demo tree
def retrieveTree(i):
  listOfTrees =[{'no surfacing': {0: 'no', 1: {'flippers': {0: 'no', 1: 'yes'}}}},
                {'no surfacing': {0: 'no', 1: {'flippers': {0: {'head': {0: 'no', 1: 'yes'}}, 1: 'no'}}}}
               ]
  return listOfTrees[i]

demoTree = retrieveTree(1)
print(demoTree)
print('getNumLeafs(demoTree)=', getNumLeafs(demoTree))
print('getTreeDepth(demoTree)=', getTreeDepth(demoTree))

demoTree = retrieveTree(0)
print(demoTree)
print('getNumLeafs(demoTree)=', getNumLeafs(demoTree))
print('getTreeDepth(demoTree)=', getTreeDepth(demoTree))

```

```
{'no surfacing': {0: 'no', 1: {'flippers': {0: {'head': {0: 'no', 1: 'yes'}}, 1: 'no'}}}}
getNumLeafs(demoTree)= 4
getTreeDepth(demoTree)= 3
{'no surfacing': {0: 'no', 1: {'flippers': {0: 'no', 1: 'yes'}}}}
getNumLeafs(demoTree)= 3
getTreeDepth(demoTree)= 2
```

이제 중요한 함수들입니다. Recursive 방식으로 트리를 읽고 구조를 따라가면서, 트리의 형태를 visualize 합니다. 해당 기능들은 plotTree()가 구현하고 있습니다.

뒤에 이어지는 plotMidText() 함수는 시각화 된 tree 위에 text를 기입해주는 utility 함수 입니다.

```
def plotTree(myTree, parentPt, nodeTxt):
  numLeafs = getNumLeafs(myTree)
  getTreeDepth(myTree)
  firstStr = list(myTree)[0]
  cntrPt = (plotTree.xOff + (1.0 + float(numLeafs))/2.0/plotTree.totalW, plotTree.yOff)
  plotMidText(cntrPt, parentPt, nodeTxt)
  plotNode(firstStr, cntrPt, parentPt, decisionNode)

  secondDict = myTree[firstStr]
```

```
  plotTree.yOff = plotTree.yOff - 1.0/plotTree.totalD

  for key in secondDict.keys():
    if type(secondDict[key]).__name__=='dict':
      plotTree(secondDict[key],cntrPt,str(key))
    else:
      plotTree.xOff = plotTree.xOff + 1.0/plotTree.totalW
      plotNode(secondDict[key], (plotTree.xOff, plotTree.yOff), cntrPt, leafNode)
      plotMidText((plotTree.xOff, plotTree.yOff), cntrPt, str(key))
  plotTree.yOff = plotTree.yOff + 1.0/plotTree.totalD

def plotMidText(cntrPt, parentPt, txtString):
  xMid = (parentPt[0]-cntrPt[0])/2.0 + cntrPt[0]
  yMid = (parentPt[1]-cntrPt[1])/2.0 + cntrPt[1]
  createPlot.ax1.text(xMid, yMid, txtString)

```

이제 `createPlot()` 함수를 새로 정의하여 지금까지 작성한 함수들로 tree를 그려냅니다.

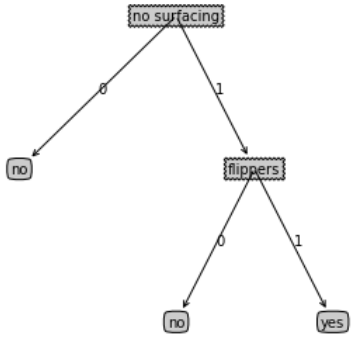
```
def createPlot(inTree):
  fig = plt.figure(1, facecolor='white')
  fig.clf()
  axprops = dict(xticks=[], yticks=[])
  createPlot.ax1 = plt.subplot(111, frameon=False, **axprops)
  plotTree.totalW = float(getNumLeafs(inTree))
  plotTree.totalD = float(getTreeDepth(inTree))
  plotTree.xOff = -0.5/plotTree.totalW
  plotTree.yOff = 1.0
  plotTree(inTree, (0.5,1.0), '')
  plt.show()

```

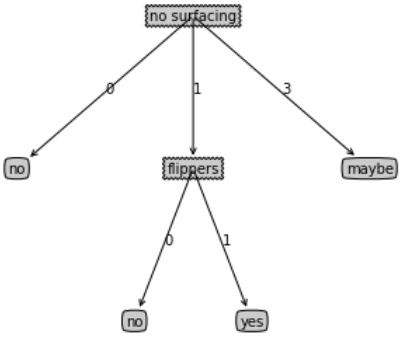
데모용 트리를 먼저 출력해봅니다.

```
# try out
demoTree = retrieveTree(0)
createPlot(demoTree)

```



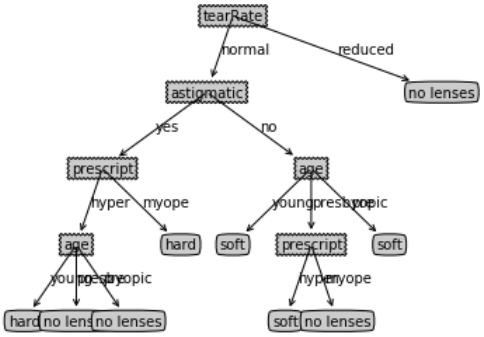

```
demoTree['no surfacing'][3]='maybe'
demoTree
createPlot(demoTree)
```



마지막으로 우리가

```
file = open('drive/My Drive/2019-Summer/camp_ML/data/lenses.txt')
lenses = [inst.strip().split('\t') for inst in file.readlines()]
# Input feature names
lensesLabels = ['age', 'prescript', 'astigmatic', 'tearRate']

lensesTree = train_DT(lenses, lensesLabels)
createPlot(lensesTree)
```


tearRate
normal
reduced
no lenses
astigmatic
yes
no
prescript
age
hyper
myope
young
hard
soft
prescript
soft
age
hard
no lens
no lenses
soft
no lenses