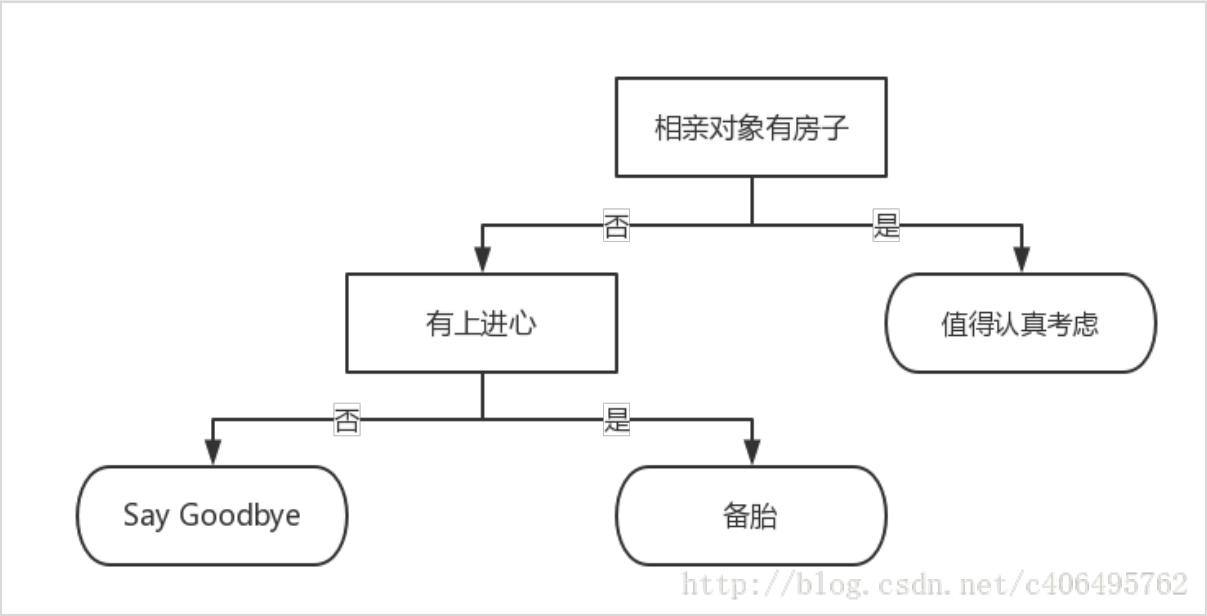
数据科学 8 机器学习： 决策树

决策树(decision tree)是一种基本的分类与回归方法。



流程图就是一个决策树，长方形代表判断模块(decision block)，椭圆形成代表终止模块(terminating block)，表示已经得出结论，可以终止运行。从判断模块引出的左右箭头称作为分支(branch)，它可以达到另一个判断模块或者终止模块。

长方形和椭圆形都是结点。长方形的结点属于内部结点，椭圆形的结点属于叶结点，从结点引出的左右箭头就是有向边。而最上面的结点就是决策树的根结点(root node)。

路径上内部结点的特征对应着规则的条件，而叶结点的类对应着规则的结论。

决策树的路径或其对应的if-then规则集合具有一个重要的性质：互斥并且完备。这就是说，每一个实例都被一条路径或一条规则所覆盖，而且只被一条路径或一条规则所覆盖。这里所覆盖是指实例的特征与路径上的特征一致或实例满足规则的条件。

决策树构建

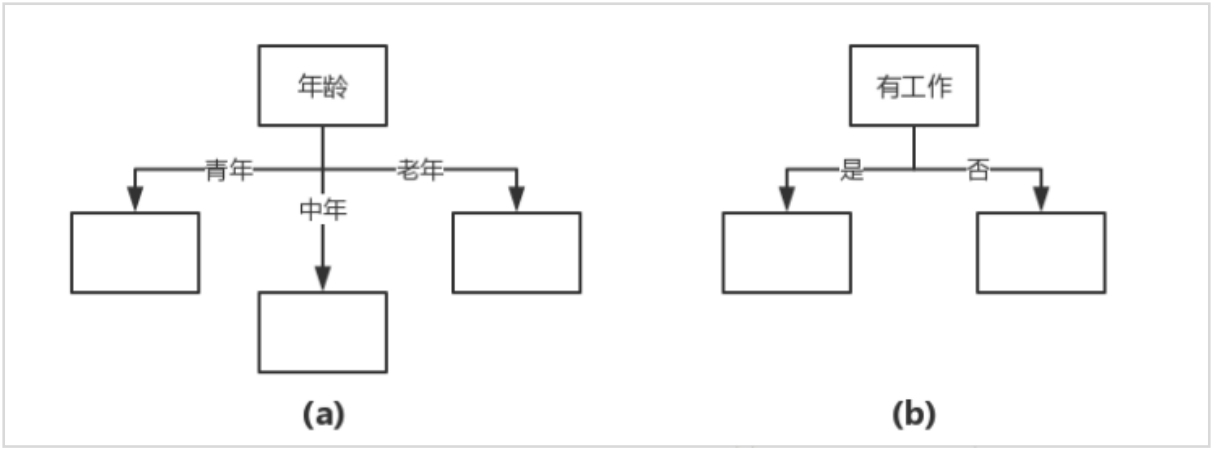
构建过程： 特征选择、决策树的生成和决策树的修剪

贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别(是否个给贷款)
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

特征选择在于选取对训练数据具有分类能力的特征。

希望通过所给的训练数据学习一个贷款申请的决策树，用以对未来的贷款申请进行分类，即当新的客户提出贷款申请时，根据申请人的特征利用决策树决定是否批准贷款申请。



图(a)所示的根结点的特征是年龄，有3个取值，对应于不同的取值有不同的子结点。图(b)所示的根节点的特征是工作，有2个取值，对应于不同的取值有不同的子结点。

通常特征选择的标准是信息增益(information gain)或信息增益比

在划分数据集之前之后信息发生的变化成为信息增益

知道如何计算信息增益，我们就可以计算每个特征值划分数据集获得的信息增益，获得信息增益最高的特征就是最好的选择

集合信息的度量方式成为香农熵或者简称为熵(entropy)，这个名字来源于信息论之父克劳德·香农。约翰·冯·诺依曼建议使用"熵"这个术语

熵定义为信息的期望值。在信息论与概率统计中，熵是表示随机变量不确定性的度量。如果待分类的事务可能划分在多个分类之中，则符号xi的信息定义为

$$l(x_i) = -log_2 p(x_i)$$

p(xi)是选择该分类的概率

为了计算熵，我们需要计算所有类别所有可能值包含的信息期望值

我们定义贷款申请样本数据表中的数据为训练数据集D，则训练数据集D的经验熵为H(D)，|D|表示其样本容量，及样本个数。设有K个类Ck，k = 1,2,3,⋯,K，|Ck|为属于类Ck的样本个数，这经验熵公式可以写为：

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

根据此公式计算经验熵H(D)，分析贷款申请样本数据表中的数据。最终分类结果只有两类，即放贷和不放贷。根据表中的数据统计可知，在15个数据中，9个数据的结果为放贷，6个数据的结果为不放贷。所以数据集D的经验熵H(D)为：

$$H(D) = -\frac{9}{15} \log_2 \frac{9}{15} - \frac{6}{15} \log_2 \frac{6}{15} = 0.971$$

编写代码计算经验熵

先对数据集进行属性标注。

- 年龄：0代表青年，1代表中年，2代表老年；
- 有工作：0代表否，1代表是；
- 有自己的房子：0代表否，1代表是；
- 信贷情况：0代表一般，1代表好，2代表非常好；
- 类别(是否给贷款)：no代表否，yes代表是

```
from math import log

"""
函数说明:创建测试数据集

Parameters:
    无
Returns:
    dataSet - 数据集
    labels - 分类属性
Modify:
    2017-07-20
"""
def createDataSet():
    dataSet = [[0, 0, 0, 0, 'no'],         #数据集
            [0, 0, 0, 1, 'no'],
            [0, 1, 0, 1, 'yes'],
            [0, 1, 1, 0, 'yes'],
            [0, 0, 0, 0, 'no'],
            [1, 0, 0, 0, 'no'],
            [1, 0, 0, 1, 'no'],
            [1, 1, 1, 1, 'yes'],
            [1, 0, 1, 2, 'yes'],
            [1, 0, 1, 2, 'yes'],
            [2, 0, 1, 2, 'yes'],
            [2, 0, 1, 1, 'yes'],
            [2, 1, 0, 1, 'yes'],
            [2, 1, 0, 2, 'yes'],
```

```
            [2, 0, 0, 0, 'no']]
    labels = ['年龄', '有工作', '有自己的房子', '信贷情况']        #分类属性
    return dataSet, labels                 #返回数据集和分类属性
```

```
"""
函数说明:计算给定数据集的经验熵(香农熵)

Parameters:
    dataSet - 数据集
Returns:
    shannonEnt - 经验熵(香农熵)
"""
def calcShannonEnt(dataSet):
    numEntires = len(dataSet)                        #返回数据集的行数
    labelCounts = {}                                 #保存每个标签(Label)出现次数的
字典
    for featVec in dataSet:                             #对每组特征向量进行统计
        currentLabel = featVec[-1]                   #提取标签(Label)信息
        if currentLabel not in labelCounts.keys():    #如果标签(Label)没有放入统计
次数的字典,添加进去
            labelCounts[currentLabel] = 0
        labelCounts[currentLabel] += 1               #Label计数
    shannonEnt = 0.0                                 #经验熵(香农熵)
    for key in labelCounts:                             #计算香农熵
        prob = float(labelCounts[key]) / numEntires    #选择该标签(Label)的概率
        shannonEnt -= prob * log(prob, 2)            #利用公式计算
    return shannonEnt                                #返回经验熵(香农熵)

if __name__ == '__main__':
    dataSet, features = createDataSet()
    print(dataSet)
    print(calcShannonEnt(dataSet))
```

信息增益

应该选择对最终分类结果影响最大的那个特征作为我们的分类特征

条件熵H(Y|X)表示在已知随机变量X的条件下随机变量Y的不确定性，随机变量X给定的条件下随机变量Y的条件熵(conditional entropy) H(Y|X)，定义X给定条件下Y的条件概率分布的熵对X的数学期望

特征A对训练数据集D的信息增益g(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即

$$g(D,A) = H(D) - H(D|A)$$

熵H(D)与条件熵H(D|A)之差成为互信息(mutual information)。决策树学习中的信息增益等价于训练数据集中类与特征的互信息

设特征A有n个不同的取值{a1,a2,···,an}，根据特征A的取值将D划分为n个子集D1,D2，···,Dn，|Di|为Di的样本个数。记子集Di中属于Ck的样本的集合为Dik，即Dik = Di ∩ Ck，|Dik|为Dik的样本个数。于是经验条件熵的公式可以些为:

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

比较特征的信息增益，由于特征A3(有自己的房子)的信息增益值最大，所以选择A3作为最优特征

```
"""
函数说明:按照给定特征划分数据集

Parameters:
    dataSet - 待划分的数据集
    axis - 划分数据集的特征
    value - 需要返回的特征的值
Returns:
```

```
    无
"""
def splitDataSet(dataSet, axis, value):
    retDataSet = []                                        #创建返回的数据集列表
    for featVec in dataSet:                                #遍历数据集
        if featVec[axis] == value:
            reducedFeatVec = featVec[:axis]                #去掉axis特征
            reducedFeatVec.extend(featVec[axis+1:])      #将符合条件的添加到返回的数
据集
            retDataSet.append(reducedFeatVec)
    return retDataSet
```

```
"""
函数说明:选择最优特征

Parameters:
    dataSet - 数据集
Returns:
    bestFeature - 信息增益最大的(最优)特征的索引值
"""
def chooseBestFeatureToSplit(dataSet):
    numFeatures = len(dataSet[0]) - 1                      #特征数量
    baseEntropy = calcShannonEnt(dataSet)                  #计算数据集的香农熵
    bestInfoGain = 0.0                                     #信息增益
    bestFeature = -1                                       #最优特征的索引值
    for i in range(numFeatures):                           #遍历所有特征
        #获取dataSet的第i个所有特征
        featList = [example[i] for example in dataSet]
        uniqueVals = set(featList)                         #创建set集合{},元素不
可重复
        newEntropy = 0.0                                   #经验条件熵
        for value in uniqueVals:                           #计算信息增益
            subDataSet = splitDataSet(dataSet, i, value)         #subDataSet划
分后的子集
            prob = len(subDataSet) / float(len(dataSet))           #计算子集的概
率
            newEntropy += prob * calcShannonEnt(subDataSet)     #根据公式计算经验
```

```
条件熵
        infoGain = baseEntropy - newEntropy                    #信息增益
        print("第%d个特征的增益为%.3f" % (i, infoGain))          #打印每个特征的信
息增益
        if (infoGain > bestInfoGain):                          #计算信息增益
            bestInfoGain = infoGain                            #更新信息增益，找
到最大的信息增益
            bestFeature = i                                    #记录信息增益最大
的特征的索引值
    return bestFeature                                         #返回信息增益
最大的特征的索引值
```

```
dataSet, features = createDataSet()
print("最优特征索引值:" + str(chooseBestFeatureToSplit(dataSet)))
```

splitDataSet函数是用来选择各个特征的子集的，比如选择年龄(第0个特征)的青年(用0代表)的自己，我们可以调用splitDataSet(dataSet,0,0)这样返回的子集就是年龄为青年的5个数据集
最优特征的索引值为2，也就是特征A3(有自己的房子)