

第17课：决策树——告诉你 Hello Kitty 是人是猫

### Hello Kitty 的种族问题

Hello Kitty，一只以无嘴造型40年来风靡全球的萌萌猫，在其40岁生日时，居然被其形象拥有者宣称：Hello Kitty 不是猫！

2014年八月，研究 Hello Kitty 多年的人类学家 Christine R. Yano 在写展品解说时，却被 Hello Kitty 持有商三丽鸥纠正：Hello Kitty 是一个卡通人物，她是一个小女孩，是一位朋友，但她“绝不”是一只猫。



粉了快半个世纪的世界萌猫，你说是人就是人啦？！就算是形象持有者，也没权利下这个定论啊!

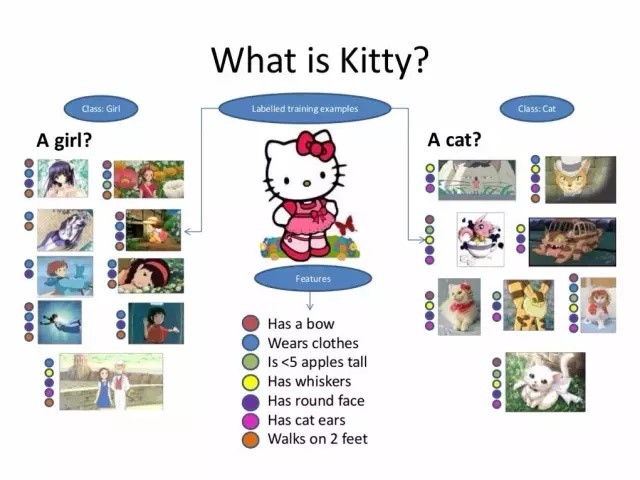
谁有权认定 Hello Kitty 是人是猫呢？我们把裁决权交给世界上最公正无私的裁判—— 计算机。让机器来决定。

机器如何具备区分一个形象属于哪个物种的知识呢？让它学习呀！机器是可以学习的。我们用计算机编个程序，再输入一堆数据，等着这个程序运行一个算法来处理这些数据。最后，我们需要的结论就显示在屏幕上啦。就是这么简单！

那么来看看我们需要的数据和算法吧。

### 训练数据

如下图所示，左边一堆是一群小女孩，右边一堆是一群猫。



### 特征选取

我们提取七个特征，用来判断一个形象，是人是猫。这七个特征包括：有否蝴蝶结；是否穿衣服；是否高过5个苹果；是否有胡子；是否圆脸；是否有猫耳朵；是否两脚走路。

用一个表格来表现这七个特征则，如下图所示（第一列为 Label，第二至八列为7个特征，每个特征只有两个取值，Yes 或者 No）：



Table-1

### 用 ID3 算法构造分类树

本例中，我们选用最简单的 ID3 算法，代入数据进行计算。

（1）根据****信息熵****的概念，我们先来计算 Entropy(S)。因为总共只有两个类别：人和猫，因此 n==2。

Entropy(S)=−∑ni=1pilog(pi)=−pGirllog(pGirl)−pCatlog(pCat)=−9/17⋅log(9/17)−8/17⋅log(8/17)=0.69Entropy(S)=−∑i=1npilog⁡(pi)=−pGirllog⁡(pGirl)−pCatlog⁡(pCat)=−9/17⋅log⁡(9/17)−8/17⋅log⁡(8/17)=0.69

（2）然后我们再分别计算各个特征的：

Entropy(S|T)=∑value(T)|Sv||S|Entropy(Sv)Entropy(S|T)=∑value(T)|Sv||S|Entropy(Sv)

因为无论哪个特征，都只有两个特征值：Yes 或者 No，因此 value(T)value(T) 总共只有两个取值。

下面以“Has a bow”为例来示意其计算过程。

Entropy(S|HasABow)=pYes(−p(Girl|Yes)log(p(Girl|Yes))–p(Cat|Yes)log(p(Cat|Yes)))+pNo(−p(Girl|No)log(p(Girl|No))–p(Cat|No)log(p(Cat|No)))=8/17⋅(−4/8⋅log(4/8)–4/8⋅log(4/8))+9/17⋅(−5/9⋅log(5/9)–4/9⋅log(4/9))=0.69Entropy(S|HasABow)=pYes(−p(Girl|Yes)log⁡(p(Girl|Yes))–p(Cat|Yes)log⁡(p(Cat|Yes)))+pNo(−p(Girl|No)log⁡(p(Girl|No))–p(Cat|No)log⁡(p(Cat|No)))=8/17⋅(−4/8⋅log⁡(4/8)–4/8⋅log(4/8))+9/17⋅(−5/9⋅log⁡(5/9)–4/9⋅log⁡(4/9))=0.69

InformationGain(T)=Entropy(S)−∑value(T)|Sv||S|Entropy(Sv)InformationGain(T)=Entropy(S)−∑value(T)|Sv||S|Entropy(Sv)

依次计算其他几项，得出如下结果：

Entropy(S|Wear Clothes) = 0.31

Entropy(S|Less than 5 apples tall) = 0.60

Entropy(S|Has whiskers) = 0.36

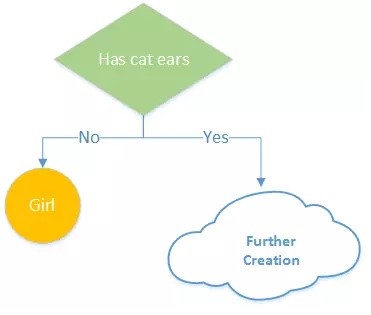
Entropy(S|Has round face) = 0.61

Entropy(S|Has cat ears) = 0.18

Entropy(S|Walks on 2 feet) = 0.36

（3）进一步计算，得出 InfoGain(Has cat ears) 最大，因此“Has cat ears”是第一个分裂节点。

而从这一特征对应的类别也可以看出，所有特征值为 No 的都一定是 Girl；特征值为 Yes，可能是 Girl 也可能是 Cat，那么第一次分裂，我们得出如下结果：



现在“Has cat ears”已经成为了分裂点，则下一步将其排除，用剩下的6个 Feature 继续分裂成树：



Table-2

Table-2 为第二次分裂所使用的训练数据，相对于 Table-1，“Has cat ears”列，和前7行对应“Has cat ears”为 No 的数据都已经被移除，剩下部分用于第二次分裂。

如此反复迭代，最后使得7个特征都成为分裂点。

需要****注意****的是，如果某个特征被选为当前轮的分裂点，但是它在现存数据中只有一个值，另一个值对应的记录为空，则这个时候针对不存在的特征值，将它标记为该特征在所有训练数据中所占比例最大的类型。

对本例而言，当我们将“Wear Clothes”作为分裂点时，会发现该特征只剩下了一个选项——Yes（如下 Table-3 所示）。此时怎么给“Wear Clothes”为 No 的分支做标记呢？

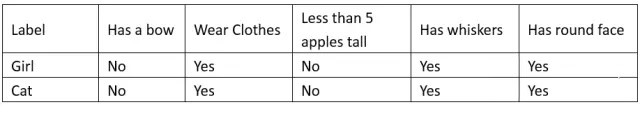
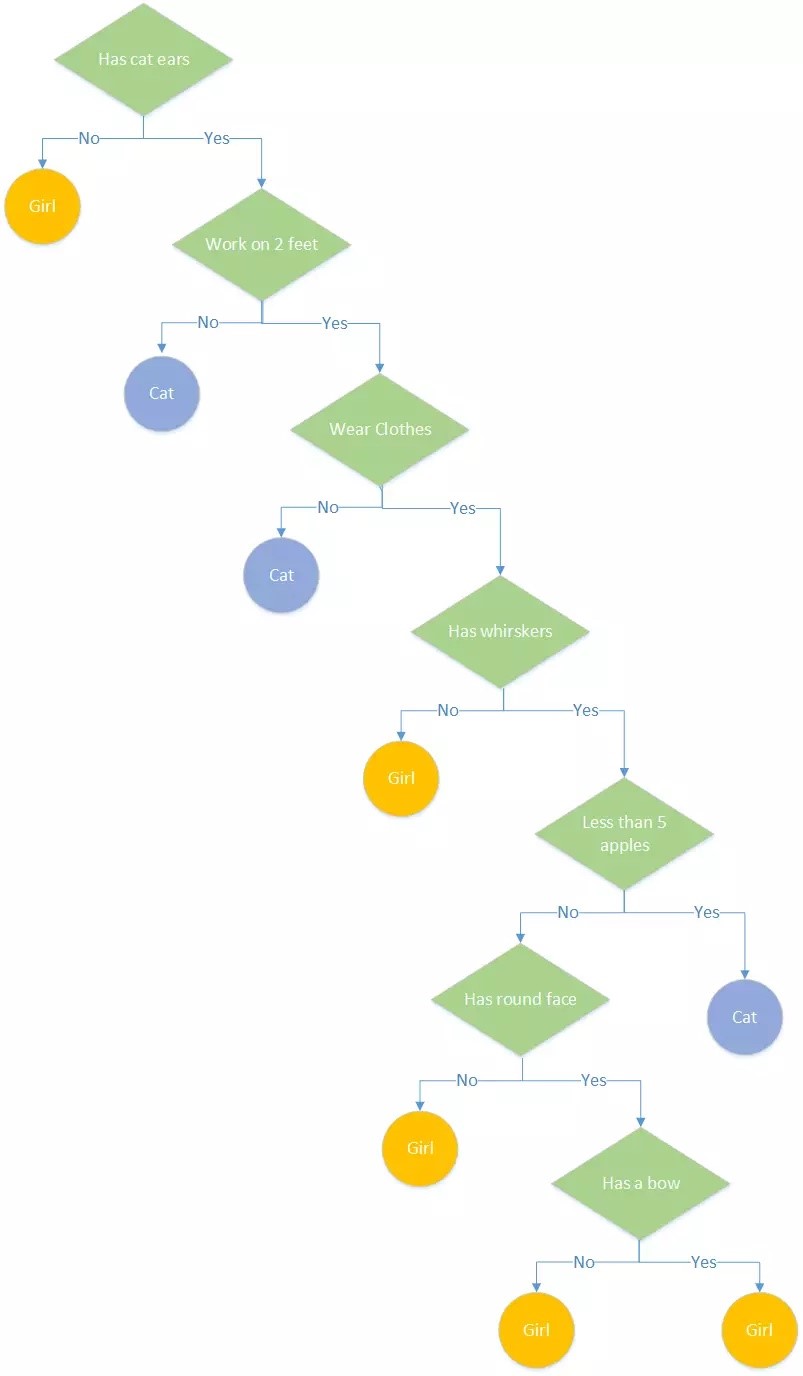


Table-3

这时就要看在 Table-1 中，“Wear Clothes”为 No 的记录中是 Girl 多还是 Cat 多。一目了然，在 Table-1 中这两种记录数量为 0：6，因此“Wear Clothes”为 No 的分支直接标志成 Cat。

根据上述方法，最终我们构建出了如下决策树：



决策树构建过程，如下代码所示：

DecisionTree **induceTree**(training\_set, features) {

If(training\_set中所有的输入项都被标记为同一个label){

**return** 一个标志位该label的叶子节点；

} **else** **if**(features为空) {

# 默认标记为在所有training\_set中所占比例最大的label

**return** 一个标记为默认label的叶子节点；

} **else** {

选取一个feature，F；

以F为根节点创建一棵树currentTree；

从Features中删除F；

**foreach**(**value** V of F) {

将training\_set中feature F的取值为V的元素全部提取出来，组成partition\_v；

branch\_v= induceTree(partition\_V, features);

将branch\_v添加为根节点的子树，根节点到branch\_v的路径为F的V值；

}

returncurrentTree；

}

}

### 后剪枝优化决策树

#### 决策树剪枝

剪枝是优化决策树的常用手段。剪枝方法大致可以分为两类：

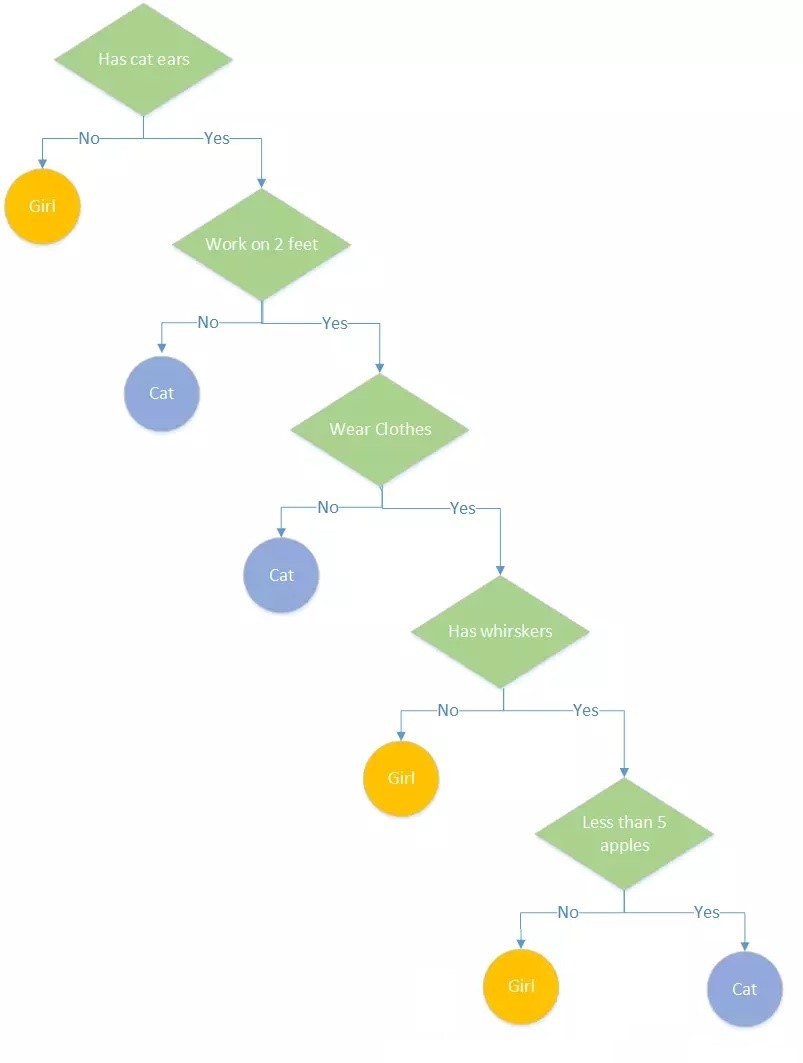
1. 先剪枝（局部剪枝）：在构造过程中，当某个节点满足剪枝条件，则直接停止此分支的构造。
2. 后剪枝（全局剪枝）：先构造完成完整的决策树，再通过某些条件遍历树进行剪枝。

#### 后剪枝优化 Hello Kitty 树

现在，决策树已经构造完成，所以我们采用后剪枝法，对上面决策树进行修剪。

如图中显示，最后两个分裂点“Has round face”和“Has a bow”存在并无意义——想想也是啊，无论人猫，都有可能是圆脸，也都可以戴蝴蝶结啊。

所以我们遍历所有节点，将没有区分作用的节点删除。完成后，我们的决策树变成了下面这样：



### 用决策树对 Hello Kitty 进行分类

我们将 Hello Kitty 的特征带入 Cat-Girl 决策树，发现 Hello Kitty：Has cat ears: Yes -> Work on 2 feet: Yes -> Wear Clothes: Yes -> Has whirskers: Yes -> Less than 5 apples: Yes -> Cat。

Bingo! Hello Kitty 是只猫！这是我们的 ID3 决策树告诉我们的！

### 代码实现

下面的代码就是用 numpy 和 sklearn 来实现例子中的训练分类树来判断 Hello Kitty 种族所对应的程序。

**from** sklearn **import** tree

**from** sklearn.model\_selection im

port train\_test\_split

**import** numpy **as** np

*#9个女孩和8只猫的数据，对应7个feature，yes取值为1，no为0*

features = np.array([

[1, 1, 0, 0, 1, 0, 1],

[1, 1, 1, 0, 0, 0, 1],

[0, 1, 0, 0, 0, 0, 1],

[1, 1, 0, 0, 1, 0, 1],

[0, 1, 0, 0, 1, 0, 0],

[0, 1, 0, 0, 1, 0, 1],

[1, 1, 0, 0, 1, 0, 0],

[0, 1, 0, 0, 1, 0, 1],

[0, 1, 0, 1, 1, 1, 1],

[1, 0, 1, 1, 1, 1, 0],

[0, 0, 0, 1, 1, 1, 0],

[1, 0, 1, 1, 1, 1, 0],

[0, 0, 0, 1, 1, 1, 0],

[1, 0, 0, 1, 1, 1, 0],

[0, 0, 1, 0, 1, 1, 0],

[1, 1, 1, 1, 1, 1, 0],

[1, 0, 1, 1, 1, 1, 0]

])

*#1 表示是女孩，0表示是猫*

labels = np.array([

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[1],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

[0],

])

*# 从数据集中取20%作为测试集，其他作为训练集*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

features,

labels,

test\_size=0.2,

random\_state=0,

)

*# 训练分类树模型*

clf = tree.DecisionTreeClassifier()

clf.fit(X=X\_train, y=y\_train)

*# 测试*

**print**(clf.predict(X\_test))

*# 对比测试结果和预期结果*

**print**(clf.score(X=X\_test, y=y\_test))

*# 预测HelloKitty*

HelloKitty = np.array([[1,1,1,1,1,1,1]])

**print**(clf.predict(HelloKitty))

最后输出为：

[1 1 0 0]

0.75

[0]

以上就是“有监督学习 Ⅰ”的内容，每天一个模型学习显然是不能做到完全学懂吃透的，这就需要同学们反复阅读文章，多思考和推算，记得打卡分享你们的学习笔记哦！

[邀请好友一起学，获得 25% 返现奖励](https://gitbook.cn/m/mazi/columns/5bc6ac7442d7d32f50f19a98/topics/5bf28251fd72950cafdccd95)

[IMG_265](https://gitbook.cn/m/mazi/comp/column?columnId=5bc6ac7442d7d32f50f19a98%26tag=2#catalog)