决策树

【关键词】树，信息增益

决策树的优缺点

优点：计算复杂度不高，输出结果易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。既能用于分类，也能用于回归

缺点：可能会产生过度匹配问题

一、决策树的原理

predict()

【二十个问题的游戏】

游戏的规则很简单：参与游戏的一方在脑海里想某个事物，其他参与者向他提问题，只允许提20个问题，问题的答案也只能用对或错回答。问问题的人通过推断分解，逐步缩小待猜测事物的范围。决策树的工作原理与20个问题类似，用户输人一系列数据 ，然后给出游戏的答案。

我们经常使用决策树处理分类问题。近来的调查表明决策树也是最经常使用的数据挖掘算法。它之所以如此流行，一个很重要的原因就是使用者基本上不用了解机器学习算法，也不用深究它是如何工作的。

如果以前没有接触过决策树，完全不用担心，它的概念非常简单。即使不知道它也可以通过简单的图形了解其工作原理。

决策树分类的思想类似于找对象。现想象一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：

女儿：多大年纪了?

母亲：26。

女儿：长的帅不帅?

母亲：挺帅的。

女儿：收入高不?

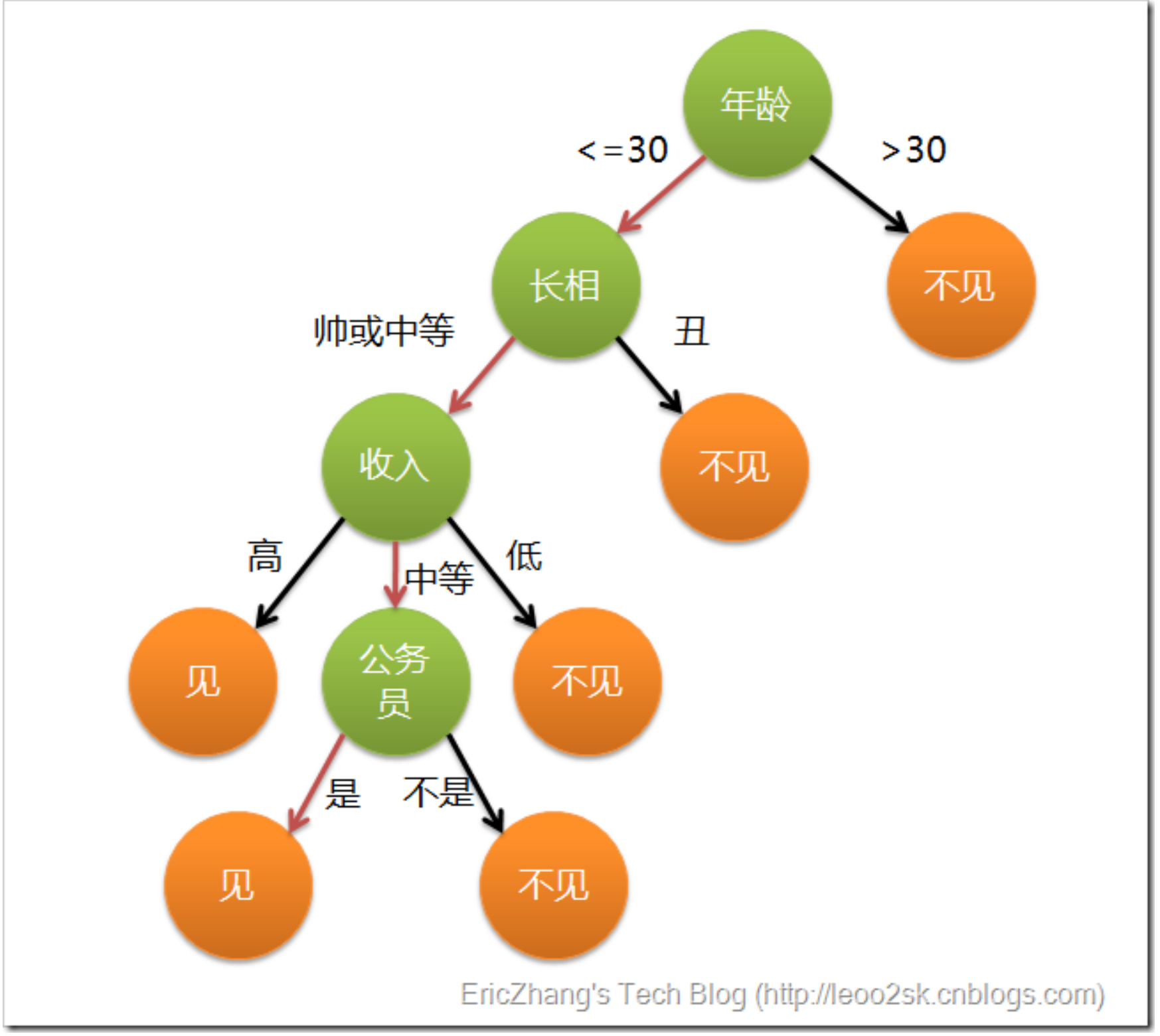
母亲：不算很高，中等情况。

女儿：是公务员不?

母亲：是，在税务局上班呢。

女儿：那好，我去见见。

这个女孩的决策过程就是典型的分类树决策。相当于通过年龄、长相、收入和是否公务员对将男人分为两个类别：见和不见。假设这个女孩对男人的要求是：30岁以下、长相中等以上并且是高收入者或中等以上收入的公务员，那么这个可以用下图表示女孩的决策逻辑:



上图完整表达了这个女孩决定是否见一个约会对象的策略，其中绿色节点表示判断条件，橙色节点表示决策结果，箭头表示在一个判断条件在不同情况下的决策路径，图中红色箭头表示了上面例子中女孩的决策过程。

这幅图基本可以算是一颗决策树，说它“基本可以算”是因为图中的判定条件没有量化，如收入高中低等等，还不能算是严格意义上的决策树，如果将所有条件量化，则就变成真正的决策树了。

有了上面直观的认识，我们可以正式定义决策树了：

决策树（decision tree）是一个树结构（可以是二叉树或非二叉树）。其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。

可以看到，决策树的决策过程非常直观，容易被人理解。目前决策树已经成功运用于医学、制造产业、天文学、分支生物学以及商业等诸多领域。

之前介绍的K-近邻算法可以完成很多分类任务，但是它最大的缺点就是无法给出数据的内在含义，决策树的主要优势就在于数据形式非常容易理解。

决策树算法能够读取数据集合，构建类似于上面的决策树。决策树很多任务都是为了数据中所蕴含的知识信息，因此决策树可以使用不熟悉的数据集合，并从中提取出一系列规则，机器学习算法最终将使用这些机器从数据集中创造的规则。专家系统中经常使用决策树，而且决策树给出结果往往可以匹敌在当前领域具有几十年工作经验的人类专家。

知道了决策树的定义以及其应用方法，下面介绍决策树的构造算法。

二、决策树的构造

分类解决离散问题，回归解决连续问题

- 决策树：信息论
- 逻辑斯底回归、贝叶斯：概率论

不同于逻辑斯蒂回归和贝叶斯算法，决策树的构造过程不依赖领域知识，它使用属性选择度量来选择将元组最好地划分成不同的类的属性。所谓决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构。

构造决策树的关键步骤是分裂属性。所谓分裂属性就是在某个节点处按照某一特征属性的不同划分构造不同的分支，其目标是让各个分裂子集尽可能地“纯”。尽可能“纯”就是尽量让一个分裂子集中待分类项属于同一类别。分裂属性分为三种不同的情况：

1、属性是离散值且不要求生成二叉决策树。此时用属性的每一个划分作为一个分支。

2、属性是离散值且要求生成二叉决策树。此时使用属性划分的一个子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支。

3、属性是连续值。此时确定一个值作为分裂点split_point，按照>split_point和<=split_point生成两个分支。

构造决策树的关键性内容是进行属性选择度量，属性选择度量是一种选择分裂准则，它决定了拓扑结构及分裂点split_point的选择。

属性选择度量算法有很多，一般使用自顶向下递归分治法，并采用不回溯的贪心策略。这里介绍常用的ID3算法。

ID3算法

划分数据集的大原则是：*将无序的数据变得更加有序。*

我们可以使用多种方法划分数据集，但是每种方法都有各自的优缺点。组织杂乱无章数据的一种方法就是使用信息论度量信息，信息论是量化处理信息的分支科学。我们可以在划分数据之前使用信息论量化度量信息的内容。

在划分数据集之前之后信息发生的变化称为信息增益，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。

在可以评测哪种数据划分方式是最好的数据划分之前，我们必须学习如何计算信息增益。集合信息的度量方式称为香农熵或者简称为熵，这个名字来源于信息论之父克劳德•香农。

entropy

熵定义为信息的期望值，在明晰这个概念之前，我们必须知道信息的定义。如果待分类的事务可能划分在多个分类之中，则符号x的信息定义为：

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

其中p(x)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，通过下面的公式得到：

$$H = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

其中n是分类的数目。

在决策树当中，设D为用类别对训练元组进行的划分，则D的熵（entropy）表示为：

$$info(D) = -\sum_{i=1}^{m} p_i log_2(p_i)$$

其中pi表示第i个类别在整个训练元组中出现的概率，可以用属于此类别元素的数量除以训练元组元素总数量作为估计。熵的实际意义表示是D中元组的类标号所需要的平均信息量。

现在我们假设将训练元组D按属性A进行划分，则A对D划分的期望信息为：

$$info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} info(D_j)$$

而信息增益即为两者的差值：

$$gain(A) = info(D) - info_A(D)$$

ID3算法就是在每次需要分裂时，计算每个属性的增益率，然后选择增益率最大的属性进行分裂。下面我们继续用SNS社区中不真实账号检测的例子说明如何使用ID3算法构造决策树。为了简单起见，我们假设训练集合包含10个元素：

日志密度	好友密度	是否使用真实头像	账号是否真实
s	s	no	no
s	l	yes	yes
l	m	yes	yes
m	m	yes	yes
l	m	yes	yes
m	l	no	yes
m	s	no	no
l	m	no	yes
m	s	no	yes
s	s	yes	no

其中s、m和l分别表示小、中和大。

设L、F和H表示日志密度、好友密度、是否使用真实头像，下面计算各属性的信息增益。

$$info(D) = -0.7log_20.7 - 0.3log_20.3 = 0.7 * 0.51 + 0.3 * 1.74 = 0.879$$

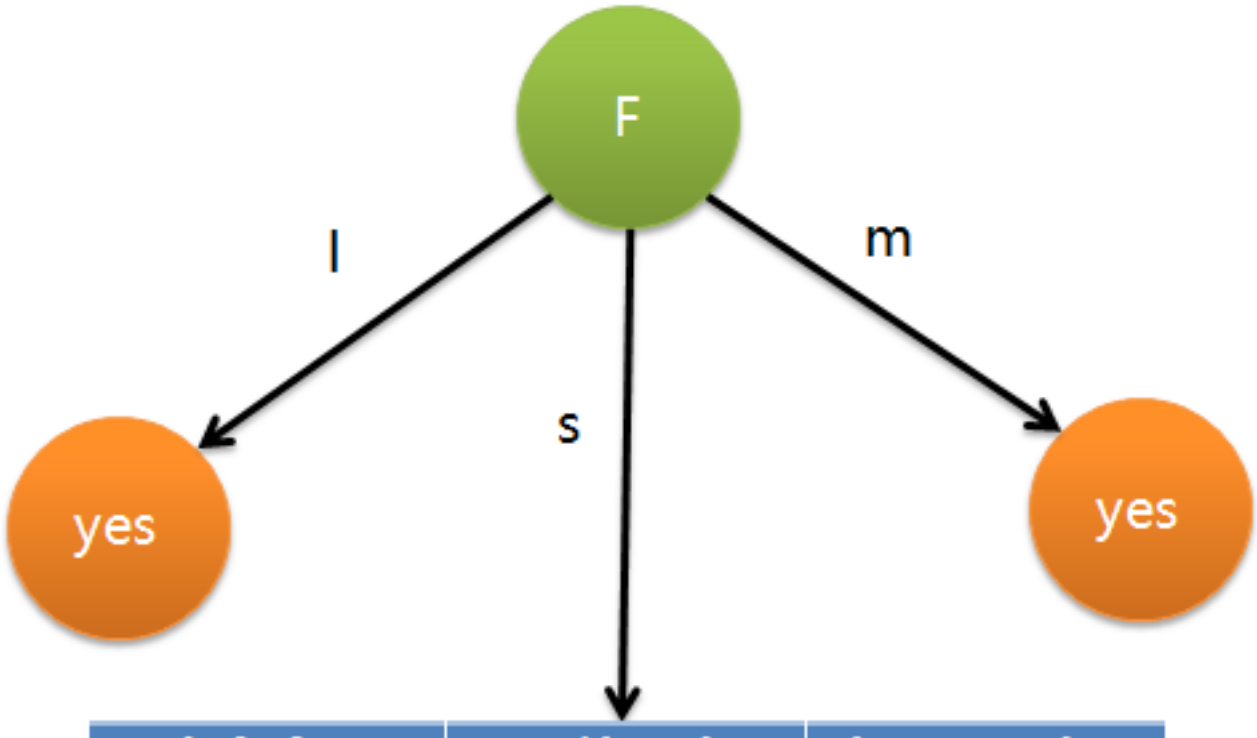
$$info_L(D) = 0.3 * (-\frac{0}{3}log_2\frac{0}{3} - \frac{3}{3}log_2\frac{3}{3}) + 0.4 * (-\frac{1}{4}log_2\frac{1}{4} - \frac{3}{4}log_2\frac{3}{4}) + 0.3 * (-\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}) = 0 + 0.326 + 0.277 = 0.603$$

$$gain(L) = 0.879 - 0.603 = 0.276$$

因此日志密度的信息增益是0.276。

用同样方法得到F和H的信息增益分别为0.553和0.033。

因为F具有最大的信息增益，所以第一次分裂选择F为分裂属性，分裂后的结果如下图表示：



日志密度	是否使用真实头像	账号是否真实
s	no	no
m	no	no
m	no	yes
s	yes	no

在上图的基础上，再递归使用这个方法计算子节点的分裂属性，最终就可以得到整个决策树。

练习

计算上图的信息熵，确定下一个分类的特征

In [1]:

```
import math
```

In [3]:

```
# 按照分类的类别划分，计算信息熵
# no  3/10
# yes 7/10
info_D = -0.3*math.log2(0.3) + (-0.7*math.log2(0.7))
info_D
```

Out[3]:

0.8812908992306927

In [4]:

```
# 按照L划分
# s 0.3 -->  no yes no  | no 2/3  yes 1/3
# m 0.4 -->  yes yes no yes  | no 1/4  yes/4
# l 0.3 -->  yes yes yes  |  no 0  yes  1

info_D_L = 0.3*(-2/3*math.log2(2/3) -1/3*math.log2(1/3)) + 0.4*(-1/4*math.log2(1/4) -3/4*math.log2(
#             0.3*(-0*math.log2(0) -1*math.log2(1))
info_D_L
```

Out[4]:

```
0.6
```

In [5]:

```
info_D - info_D_L
```

Out[5]:

```
0.2812908992306927
```

In [8]:

```
# s 4/10  ---> no no yes no  |no 3/4 yes 1/4
# m 4/10  ---> yes yes yes yes  |no 0  yes 1
# l 2/10  ---> yes yes  | no 0 yes 1
info_D_F = 0.4*(-3/4*math.log2(3/4) -1/4*math.log2(1/4))
info_D_F
```

Out[8]:

```
0.32451124978365314
```

In [9]:

```
info_D - info_D_F
```

Out[9]:

```
0.5567796494470396
```

In [11]:

```
# yes 0.5   --->  no yes no yes yes  |no 2/5 yes 3/5
# no 0.5    --->  yes yes yes yes no  | no 1/5  yes 4/5
info_D_H = 0.5*(-2/5*math.log2(2/5) -3/5*math.log2(3/5)) + 0.5*(-1/5*math.log2(1/5) -4/5*math.log2
info_D_H
```

Out[11]:

```
0.8464393446710154
```

In [12]:

```
info_D - info_D_H
```

Out[12]:

0.034851554559677256

三、实战

【注意】 参数max_depth越大，越容易过拟合

In [13]:

```
# 决策树分类模型
from sklearn.tree import DecisionTreeClassifier
```

1、使用自带的iris数据集

In [14]:

```
import sklearn.datasets as datasets

iris = datasets.load_iris()
```

In [18]:

```
data = iris.data
target = iris.target
samples = data[:,:2]
```

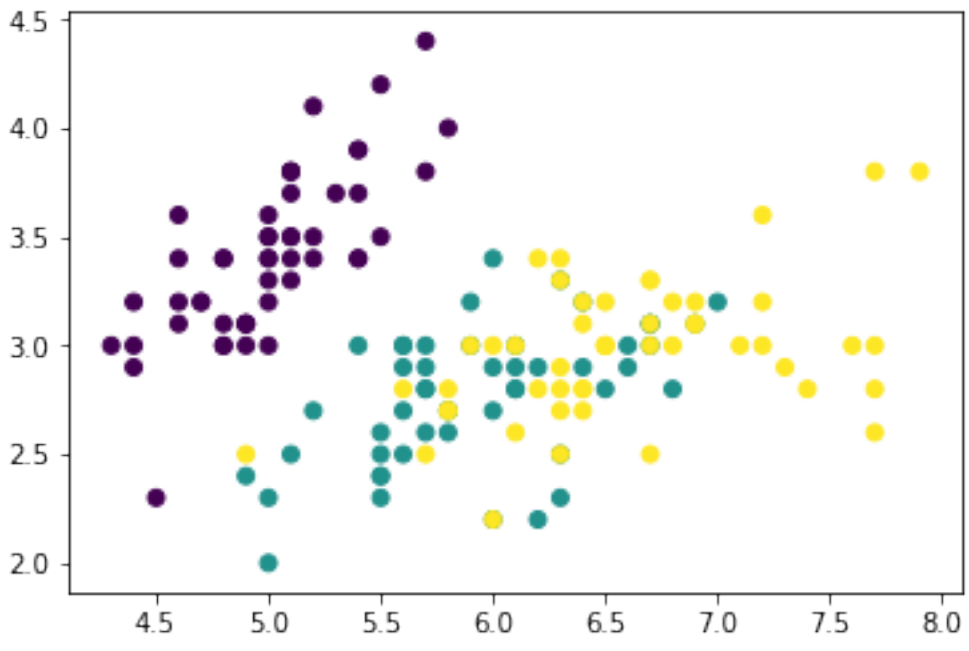
In [19]:

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.scatter(samples[:,0],samples[:,1],c=target)
```

Out[19]:

<matplotlib.collections.PathCollection at 0x16af7ab0>



使用决策树算法

In [21]:

```
import numpy as np
xmin,xmax = samples[:,0].min()-0.5,samples[:,0].max()+0.5
ymin,ymax = samples[:,1].min()-0.5,samples[:,1].max()+0.5

x = np.linspace(xmin,xmax,300)
y = np.linspace(ymin,ymax,300)

xx,yy = np.meshgrid(x,y)

X_test = np.c_[xx.ravel(),yy.ravel()]
```

In [28]:

```
from matplotlib.colors import ListedColormap
colormatp = ListedColormap(['r','g','b'])
```

In [33]:

```
# max_depth决策树深度，值越大越精确，但容易出现过拟合
decision = DecisionTreeClassifier(max_depth=5)
decision.fit(samples,target)
y_ = decision.predict(X_test)

plt.scatter(X_test[:,0],X_test[:,1],c=y_)
plt.scatter(samples[:,0],samples[:,1],c=target,cmap=colormatp)
```

. . .

使用KNN算法

In [30]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(samples,target)

y1_ = knn.predict(X_test)
plt.scatter(X_test[:,0],X_test[:,1],c=y1_)
plt.scatter(samples[:,0],samples[:,1],c=target,cmap=colormatp)
```

. . .

使用逻辑斯蒂回归算法

In [31]:

```
from sklearn.linear_model import LogisticRegression

logistic = LogisticRegression()
logistic.fit(samples,target)
y2_ = logistic.predict(X_test)
plt.scatter(X_test[:,0],X_test[:,1],c=y2_)
plt.scatter(samples[:,0],samples[:,1],c=target,cmap=colormatp)
```

. . .

2、使用回归预测一个椭圆

使用RandomState生成固定随机数
创建-100到100之间的随机弧度
生成正弦值和余弦值
添加噪声

In [126]:

```
# 固定随机数
np.random.seed(1)

radians = np.sort(np.random.random(200))*200-100

X = np.cos(radians)
y = np.sin(radians)

plt.scatter(X,y)
```

. . .

In [127]:

```
noise = np.random.random(50) - 0.5
noise
```

. . .

In [128]:

```
y[::4] += noise
```

In [129]:

```
plt.scatter(X,y)
```

. . .

撒盐操作，让圆上的20个点不规则的显示在圆的周围

创建不同深度的决策树
进行数据训练

In [141]:

```
from sklearn.tree import DecisionTreeRegressor

tree1 = DecisionTreeRegressor(max_depth=1)
tree2 = DecisionTreeRegressor(max_depth=5)
tree3 = DecisionTreeRegressor(max_depth=50)

X_train = radians.reshape(-1,1)
y_train = np.c_[X,y]

tree1.fit(X_train,y_train)
tree2.fit(X_train,y_train)
tree3.fit(X_train,y_train)
```

Out[141]:

```
DecisionTreeRegressor(criterion='mse', max_depth=50, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

创建-100到100的预测数据，间隔为0.01
对数据进行预测

In [142]:

```
X_test = np.arange(-100,100,0.01).reshape(-1,1)

y1_ = tree1.predict(X_test)
y2_ = tree2.predict(X_test)
y3_ = tree3.predict(X_test)
```

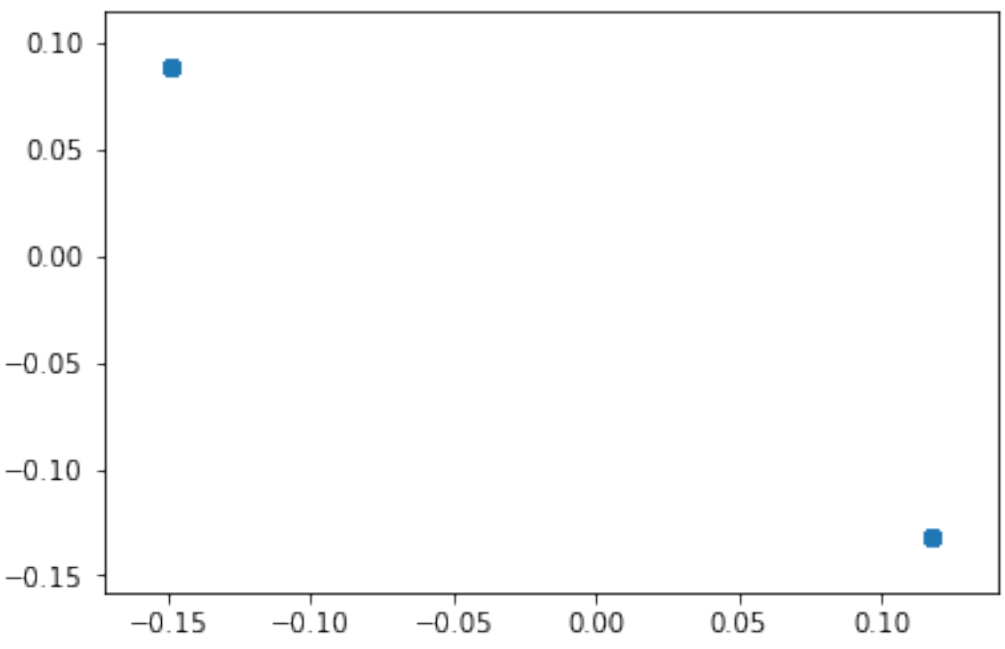
显示图片

In [143]:

```
plt.scatter(y1_[:,0],y1_[:,1])
```

Out[143]:

<matplotlib.collections.PathCollection at 0x1bded1f0>

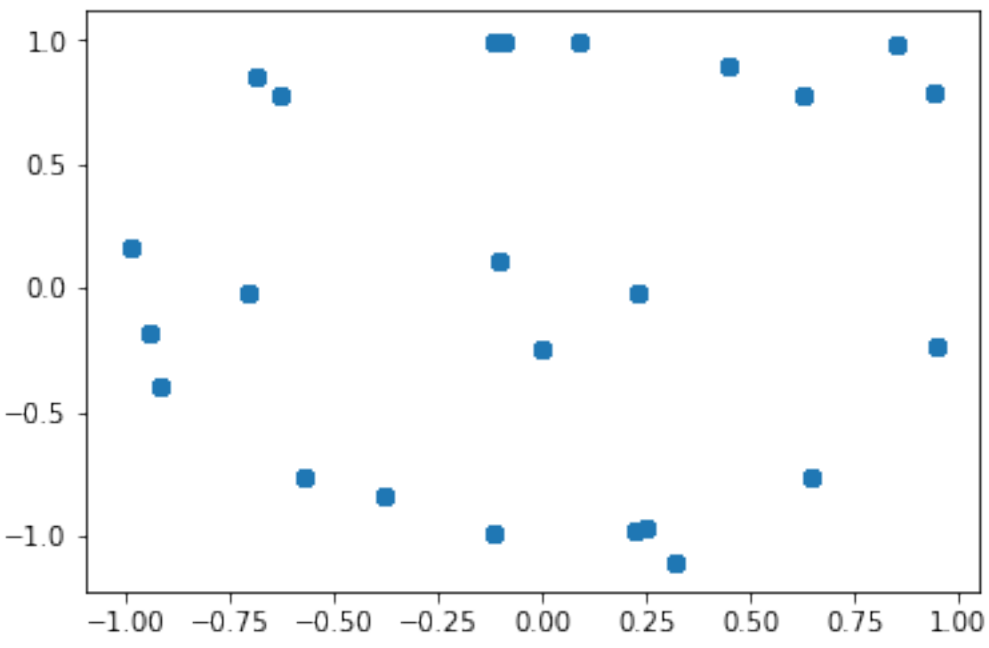


In [144]:

```
plt.scatter(y2_[:,0],y2_[:,1])
```

Out[144]:

<matplotlib.collections.PathCollection at 0x1be28270>

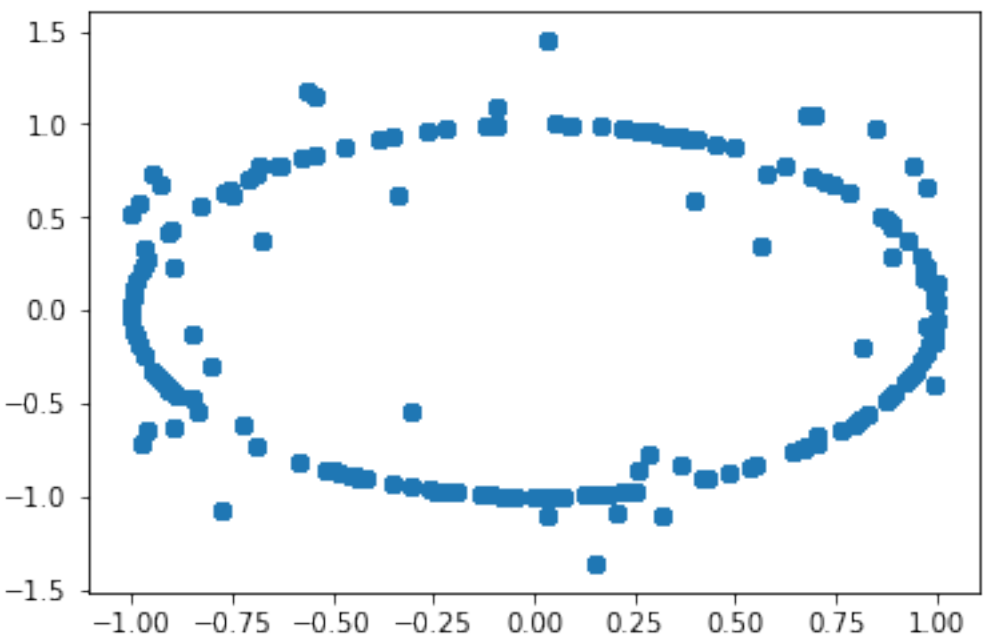


In [145]:

```
plt.scatter(y3_[:,0],y3_[:,1])
```

Out[145]:

<matplotlib.collections.PathCollection at 0x1be628b0>



In [111]:

```
X = np.sort(200 * np.random.random(200) - 100,axis = 0)
#根据角度生成正弦值和余弦值，这些值就是圆上面的点
y = np.array([np.pi * np.sin(X).ravel(),np.pi * np.cos(X).ravel()]).transpose()
```

四、作业

1、预测隐形眼镜的类型

分析lenses.txt文件

In [2]:

```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame

import matplotlib.pyplot as plt
%matplotlib inline
```

In [16]:

```
data = pd.read_csv('../data/lenses.txt',sep='\t',header=None)
train = data.iloc[:,:4].copy()
target = data[4]
```

In [8]:

```
target.unique()
```

Out[8]:

```
array(['no lenses', 'soft', 'hard'], dtype=object)
```

In [10]:

```
train[0].unique()
```

Out[10]:

```
array(['young', 'pre', 'presbyopic'], dtype=object)
```

In [11]:

```
train[1].unique()
```

Out[11]:

```
array(['myope', 'hyper'], dtype=object)
```

In [26]:

```
train[3].unique()
```

Out[26]:

```
array(['reduced', 'normal'], dtype=object)
```

In [27]:

```
map_dic = {
    'young':0,
    'pre':1,
    'presbyopic':2,
    'myope':0,
    'hyper':1,
    'no':0,
    'yes':1,
    'reduced':0,
    'normal':1
}
```

In [28]:

```
train.replace(map_dic,inplace=True)
```

In [39]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(train,target,test_size=0.2,random_state=1)
```

In [21]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

In [41]:

```
knn = KNeighborsClassifier(n_neighbors=5)
logistic = LogisticRegression()
tree = DecisionTreeClassifier(max_depth=3)
gaussion = GaussianNB()

print("knn score is %f"%knn.fit(X_train,y_train).score(X_test,y_test))
print("logisitci score is %f"%logistic.fit(X_train,y_train).score(X_test,y_test))
print("tree score is %f"%tree.fit(X_train,y_train).score(X_test,y_test))
print("gaussion score is %f"%gaussion.fit(X_train,y_train).score(X_test,y_test))
```

```
knn score is 1.000000
logisitci score is 0.800000
tree score is 1.000000
gaussion score is 1.000000
```

2、使用make_blobs产生数据，训练模型，并画出类别边界。

决策树

【关键词】树，信息增益

决策树的优缺点

优点：计算复杂度不高，输出结果易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。既能用于分类，也能用于回归

缺点：可能会产生过度匹配问题

一、决策树的原理

predict()

【二十个问题的游戏】

游戏的规则很简单：参与游戏的一方在脑海里想某个事物，其他参与者向他提问题，只允许提20个问题，问题的答案也只能用对或错回答。问问题的人通过推断分解，逐步缩小待猜测事物的范围。决策树的工作原理与20个问题类似，用户输人一系列数据 ，然后给出游戏的答案。

我们经常使用决策树处理分类问题。近来的调查表明决策树也是最经常使用的数据挖掘算法。它之所以如此流行，一个很重要的原因就是使用者基本上不用了解机器学习算法，也不用深究它是如何工作的。

如果以前没有接触过决策树，完全不用担心，它的概念非常简单。即使不知道它也可以通过简单的图形了解其工作原理。

决策树分类的思想类似于找对象。现想象一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：

女儿：多大年纪了?

母亲：26。

女儿：长的帅不帅?

母亲：挺帅的。

女儿：收入高不?

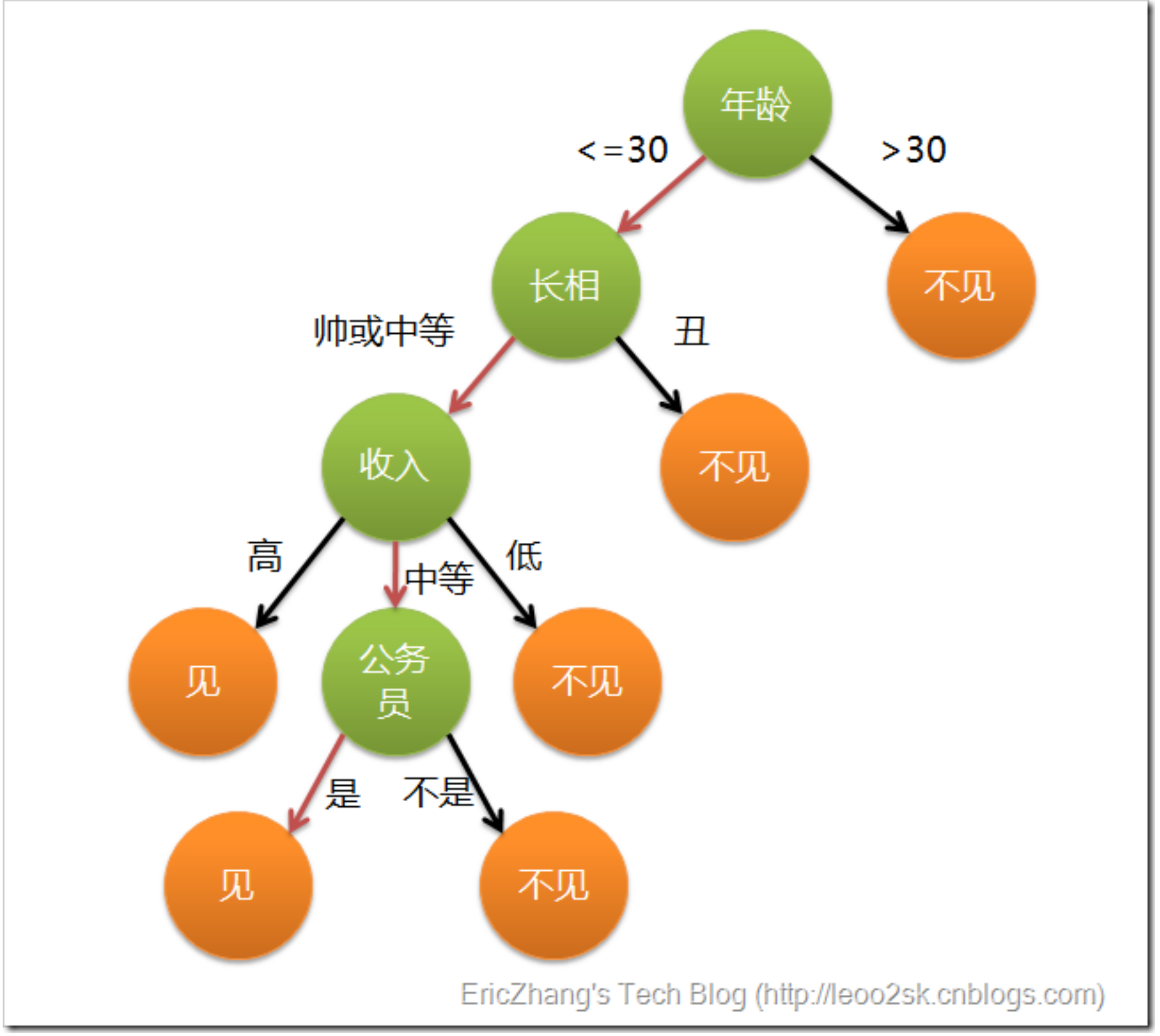
母亲：不算很高，中等情况。

女儿：是公务员不?

母亲：是，在税务局上班呢。

女儿：那好，我去见见。

这个女孩的决策过程就是典型的分类树决策。相当于通过年龄、长相、收入和是否公务员对将男人分为两个类别：见和不见。假设这个女孩对男人的要求是：30岁以下、长相中等以上并且是高收入者或中等以上收入的公务员，那么这个可以用下图表示女孩的决策逻辑:



上图完整表达了这个女孩决定是否见一个约会对象的策略，其中绿色节点表示判断条件，橙色节点表示决策结果，箭头表示在一个判断条件在不同情况下的决策路径，图中红色箭头表示了上面例子中女孩的决策过程。

这幅图基本可以算是一颗决策树，说它“基本可以算”是因为图中的判定条件没有量化，如收入高中低等等，还不能算是严格意义上的决策树，如果将所有条件量化，则就变成真正的决策树了。

有了上面直观的认识，我们可以正式定义决策树了：

决策树（decision tree）是一个树结构（可以是二叉树或非二叉树）。其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。

可以看到，决策树的决策过程非常直观，容易被人理解。目前决策树已经成功运用于医学、制造产业、天文学、分支生物学以及商业等诸多领域。

之前介绍的K-近邻算法可以完成很多分类任务，但是它最大的缺点就是无法给出数据的内在含义，决策树的主要优势就在于数据形式非常容易理解。

决策树算法能够读取数据集合，构建类似于上面的决策树。决策树很多任务都是为了数据中所蕴含的知识信息，因此决策树可以使用不熟悉的数据集合，并从中提取出一系列规则，机器学习算法最终将使用这些机器从数据集中创造的规则。专家系统中经常使用决策树，而且决策树给出结果往往可以匹敌在当前领域具有几十年工作经验的人类专家。

知道了决策树的定义以及其应用方法，下面介绍决策树的构造算法。

二、决策树的构造

分类解决离散问题，回归解决连续问题

- 决策树：信息论
- 逻辑斯底回归、贝叶斯：概率论

不同于逻辑斯蒂回归和贝叶斯算法，决策树的构造过程不依赖领域知识，它使用属性选择度量来选择将元组最好地划分成不同的类的属性。所谓决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构。

构造决策树的关键步骤是分裂属性。所谓分裂属性就是在某个节点处按照某一特征属性的不同划分构造不同的分支，其目标是让各个分裂子集尽可能地“纯”。尽可能“纯”就是尽量让一个分裂子集中待分类项属于同一类别。分裂属性分为三种不同的情况：

1、属性是离散值且不要求生成二叉决策树。此时用属性的每一个划分作为一个分支。

2、属性是离散值且要求生成二叉决策树。此时使用属性划分的一个子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支。

3、属性是连续值。此时确定一个值作为分裂点split_point，按照>split_point和<=split_point生成两个分支。

构造决策树的关键性内容是进行属性选择度量，属性选择度量是一种选择分裂准则，它决定了拓扑结构及分裂点split_point的选择。

属性选择度量算法有很多，一般使用自顶向下递归分治法，并采用不回溯的贪心策略。这里介绍常用的ID3算法。

ID3算法

划分数据集的大原则是：*将无序的数据变得更加有序。*

我们可以使用多种方法划分数据集，但是每种方法都有各自的优缺点。组织杂乱无章数据的一种方法就是使用信息论度量信息，信息论是量化处理信息的分支科学。我们可以在划分数据之前使用信息论量化度量信息的内容。

在划分数据集之前之后信息发生的变化称为信息增益，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。

在可以评测哪种数据划分方式是最好的数据划分之前，我们必须学习如何计算信息增益。集合信息的度量方式称为香农熵或者简称为熵，这个名字来源于信息论之父克劳德•香农。

entropy

熵定义为信息的期望值，在明晰这个概念之前，我们必须知道信息的定义。如果待分类的事务可能划分在多个分类之中，则符号x的信息定义为：

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

其中p(x)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，通过下面的公式得到：

$$H = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

其中n是分类的数目。

在决策树当中，设D为用类别对训练元组进行的划分，则D的熵（entropy）表示为：

$$info(D) = -\sum_{i=1}^{m} p_i log_2(p_i)$$

其中pi表示第i个类别在整个训练元组中出现的概率，可以用属于此类别元素的数量除以训练元组元素总数量作为估计。熵的实际意义表示是D中元组的类标号所需要的平均信息量。

现在我们假设将训练元组D按属性A进行划分，则A对D划分的期望信息为：

$$info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} info(D_j)$$

而信息增益即为两者的差值：

$$gain(A) = info(D) - info_A(D)$$

ID3算法就是在每次需要分裂时，计算每个属性的增益率，然后选择增益率最大的属性进行分裂。下面我们继续用SNS社区中不真实账号检测的例子说明如何使用ID3算法构造决策树。为了简单起见，我们假设训练集合包含10个元素：

日志密度	好友密度	是否使用真实头像	账号是否真实
s	s	no	no
s	l	yes	yes
l	m	yes	yes
m	m	yes	yes
l	m	yes	yes
m	l	no	yes
m	s	no	no
l	m	no	yes
m	s	no	yes
s	s	yes	no

其中s、m和l分别表示小、中和大。

设L、F和H表示日志密度、好友密度、是否使用真实头像，下面计算各属性的信息增益。

$$info(D) = -0.7log_20.7 - 0.3log_20.3 = 0.7 * 0.51 + 0.3 * 1.74 = 0.879$$

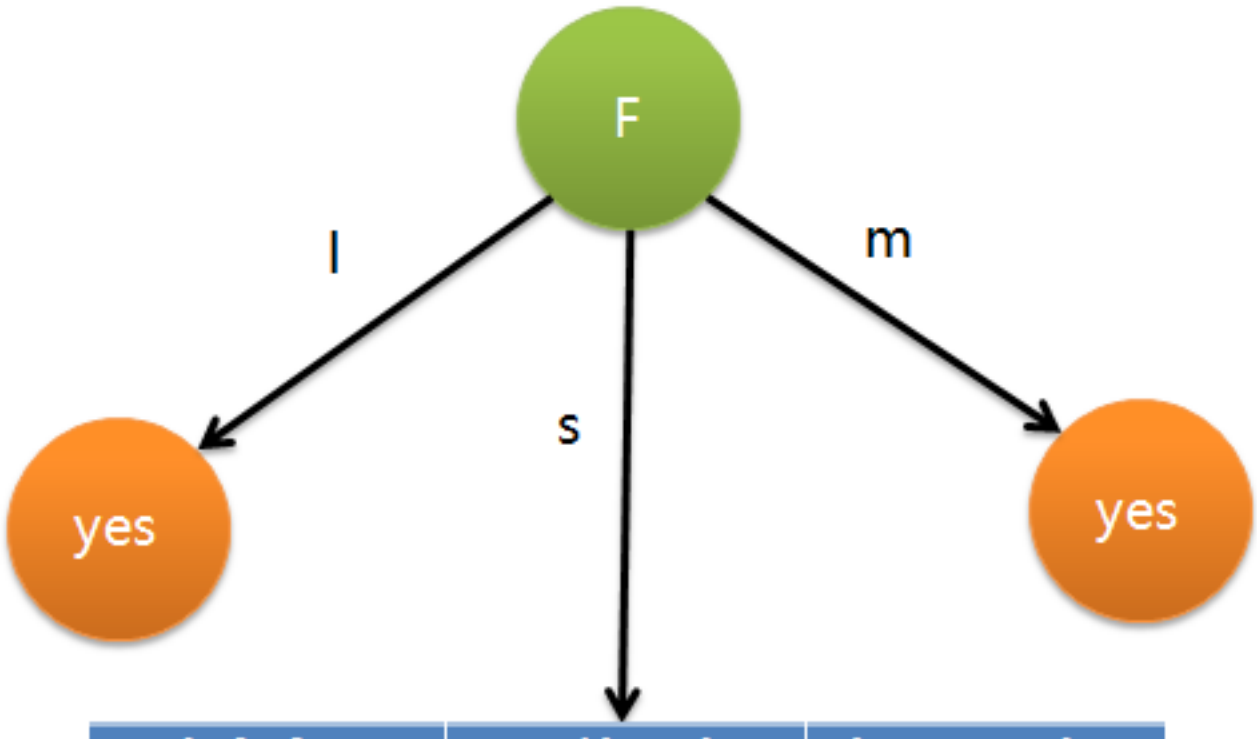
$$info_L(D) = 0.3 * (-\frac{0}{3}log_2\frac{0}{3} - \frac{3}{3}log_2\frac{3}{3}) + 0.4 * (-\frac{1}{4}log_2\frac{1}{4} - \frac{3}{4}log_2\frac{3}{4}) + 0.3 * (-\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}) = 0 + 0.326 + 0.277 = 0.603$$

$$gain(L) = 0.879 - 0.603 = 0.276$$

因此日志密度的信息增益是0.276。

用同样方法得到F和H的信息增益分别为0.553和0.033。

因为F具有最大的信息增益，所以第一次分裂选择F为分裂属性，分裂后的结果如下图表示：



日志密度	是否使用真实头像	账号是否真实
s	no	no
m	no	no
m	no	yes
s	yes	no

在上图的基础上，再递归使用这个方法计算子节点的分裂属性，最终就可以得到整个决策树。

练习

计算上图的信息熵，确定下一个分类的特征

In [1]:

In [3]:

Out[3]:

0.8812908992306927

In [4]:

Out[4]:

0.6

In [5]:

Out[5]:

0.2812908992306927

In [8]:

Out[8]:

0.32451124978365314

In [9]:

Out[9]:

0.5567796494470396

In [11]:

Out[11]:

0.8464393446710154

In [12]:

Out[12]:

0.0348515545596772 56

三、实战

【注意】 参数max_depth越大，越容易过拟合

In [13]:

1、使用自带的iris数据集

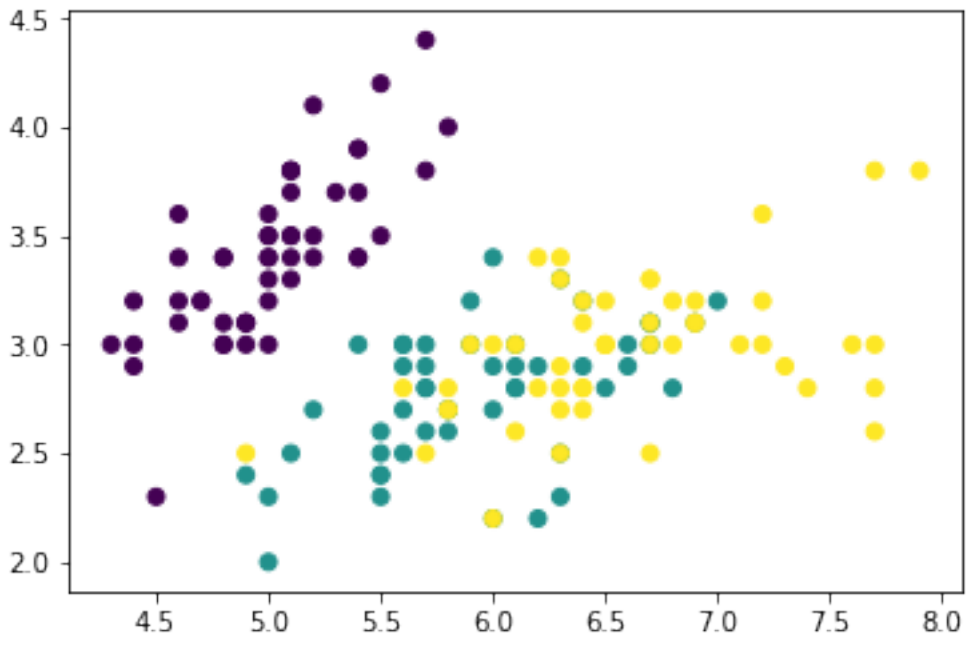
In [14]:

In [18]:

In [19]:

Out[19]:

<matplotlib.collections.PathCollection at 0x16af7ab0>



使用决策树算法

In [21]:

In [28]:

In [33]:

. . .

使用KNN算法

In [30]:

. . .

使用逻辑斯蒂回归算法

In [31]:

. . .

2、使用回归预测一个椭圆

使用RandomState生成固定随机数
创建-100到100之间的随机弧度
生成正弦值和余弦值
添加噪声

In [126]:

```
...
```

In [127]:

```
...
```

In [128]:

```

```

In [129]:

```
...
```

撒盐操作，让圆上的20个点不规则的显示在圆的周围

创建不同深度的决策树
进行数据训练

In [141]:

```

```

Out[141]:

```
DecisionTreeRegressor(criterion='mse', max_depth=50, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

创建-100到100的预测数据，间隔为0.01
对数据进行预测

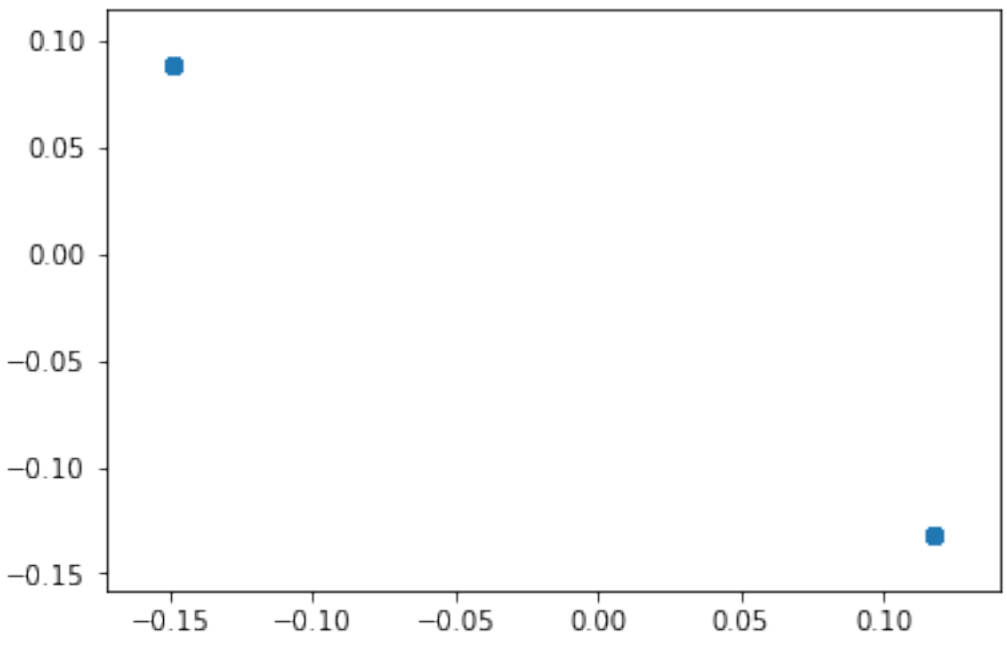
In [142]:

显示图片

In [143]:

Out[143]:

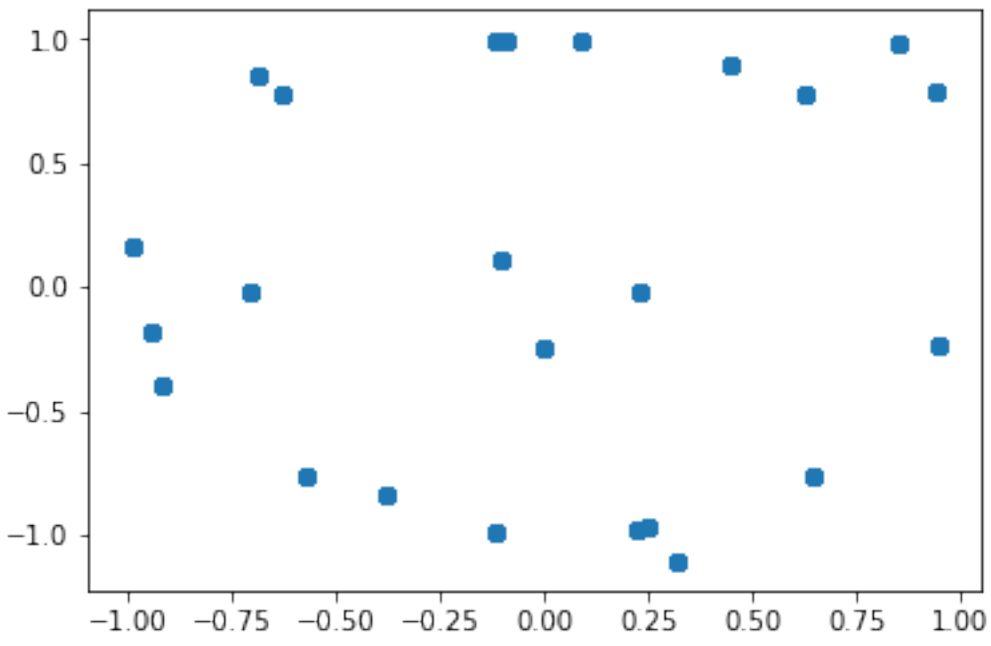
<matplotlib.collections.PathCollection at 0x1bded1f0>



In [144]:

Out[144]:

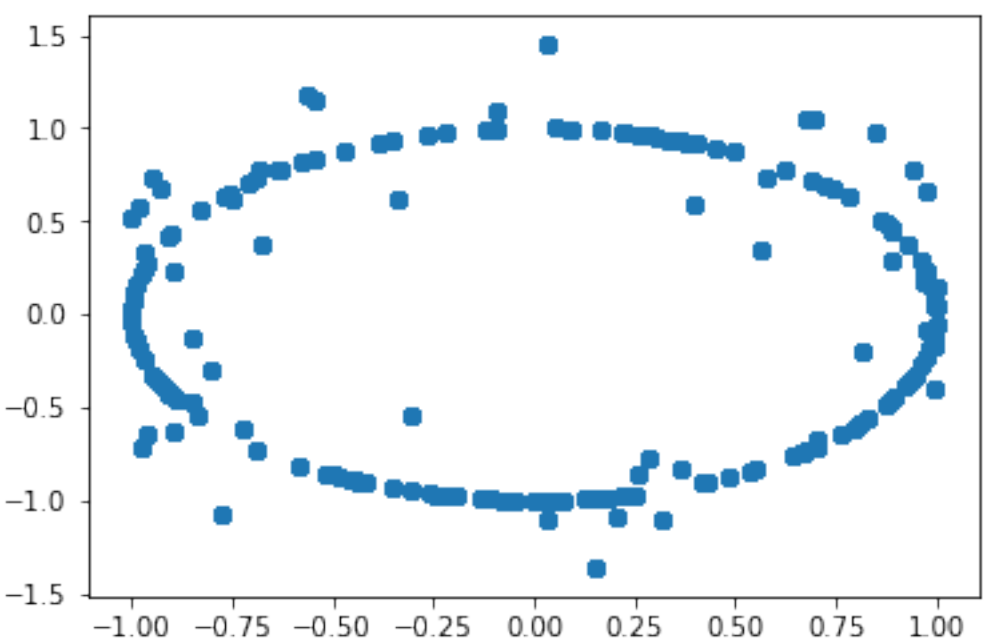
<matplotlib.collections.PathCollection at 0x1be28270>



In [145]:

Out[145]:

<matplotlib.collections.PathCollection at 0x1be628b0>



In [111]:

四、作业

1、预测隐形眼镜的类型

分析lenses.txt文件

In [2]:

In [16]:

In [8]:

Out[8]:

array(['no lenses', 'soft', 'hard'], dtype=object)

In [10]:

Out[10]:

array(['young', 'pre', 'presbyopic'], dtype=object)

In [11]:

Out[11]:

array(['myope', 'hyper'], dtype=object)

In [26]:

Out[26]:

array(['reduced', 'normal'], dtype=object)

In [27]:

In [28]:

In [39]:

In [21]:

In [41]:

```
knn score is 1.000000
logisitci score is 0.800000
tree score is 1.000000
gaussion score is 1.000000
```

2、使用make_blobs产生数据，训练模型，并画出类别边界。