# 使用决策树算法在印第安人糖尿病数据实现分类器

## 一、准备

安装python + numpy + scipy

安装scikit-learn： sudo pip install scikit-learn

测试scikit-learn安装：

Python

>>> from sklearn import tree

>>> X = [[0, 0], [1, 1]]

>>> Y = [0, 1]

>>> clf = tree.DecisionTreeClassifier()

>>> clf = clf.fit(X, Y)

>>> clf.predict([[2., 2.]])

array([1])

>>>

实验数据：皮马印第安人糖尿病数据集（UCI）<https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data>

一共768条数据，每一条数据包括8个健康指标（年龄血压等）和一个类别信息（{0, 1}）.

示例：

6,148,72,35,0,33.6,0.627,50,1

1,85,66,29,0,26.6,0.351,31,0

8,183,64,0,0,23.3,0.672,32,1

...

## 二、实现

scikit-learn的[DecisionTreeClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html" \l "sklearn.tree.DecisionTreeClassifier" \o "sklearn.tree.DecisionTreeClassifier)能够进行多重分类。输入数据为两个数组：第一个参数X[m, n]包含所有训练样本的特征值，其中m为样本数、n为特征数；第二个参数Y[m]为对应的类别标签。scikit-learn uses an optimised version of the CART algorithm: <http://scikit-learn.org/stable/modules/tree.html>

### 2.1 训练分类器

使用前650个数据作为训练集，训练分类器：

from sklearn import tree

# use the first 650 samples as training set

x = []

y = []

# use sample 651- as testing set

xc = []

yc = []

#import data

i = 0

with open(*'pima-indians-diabetes.data'*,*'r'*) as f:

for line in f:

arr = map(float,line.split(*','*))

if i < 650 :

y.append(int(arr[8]))

del arr[8]

x.append(arr)

else :

yc.append(int(arr[8]))

del arr[8]

xc.append(arr)

i = i+1

f.close()

# train the classifier

clf = tree.DecisionTreeClassifier()

clf = clf.fit(x, y)

## 2.2 测试分类效果

以后118条数据作为测试集。计算分类的准确度：

# predict

predicts = clf.predict(xc)

# calculate the precision

total = 0

match = 0

for predict in predicts :

if predict == yc[total] :

match = match + 1

total = total + 1

print *'predict precision: '*

print(float(match)/total)

结果：

predict precision:

0.703389830508

## 三、可视化

下面为了直观地观察decision tree的形态，使用pydot将分类器绘制到文件查看。

安装pydot：sudo apt-get install python-pydot

将训练出的分类器输出为pdf文件：

from sklearn.externals.six import StringIO

import pydot

...

#visualize decision tree

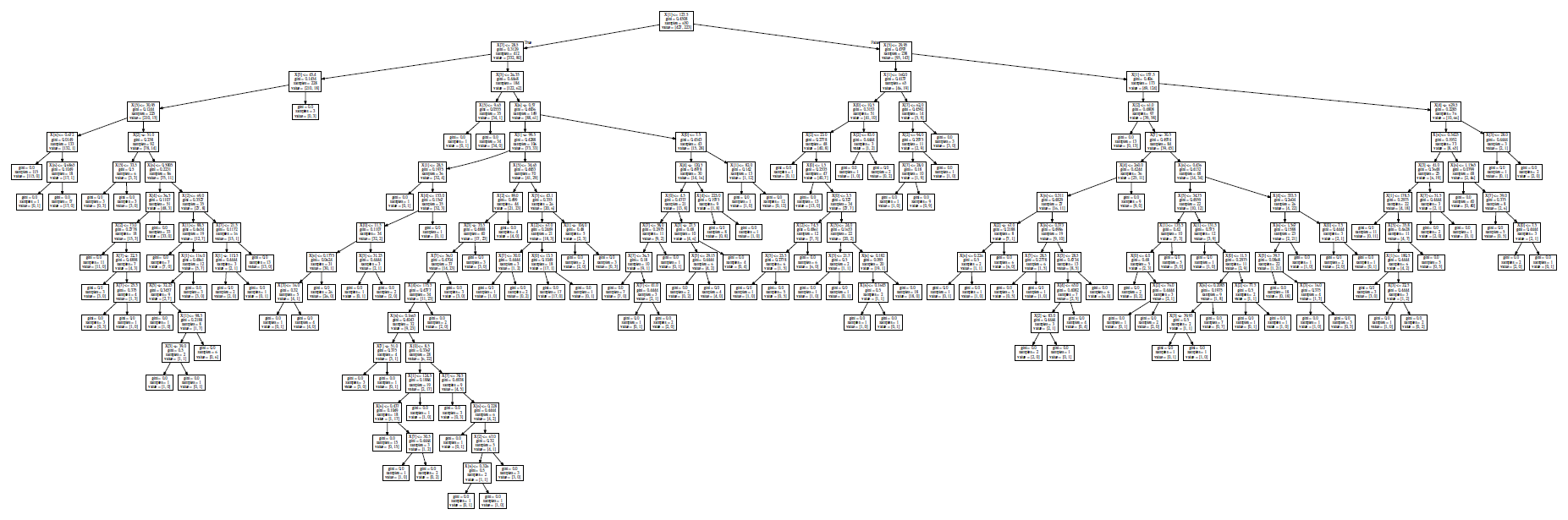
dot\_data = StringIO()

tree.export\_graphviz(clf, out\_file=dot\_data)

graph = pydot.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_pdf(*"indians-diabetes.pdf"*)

得到文件indians-diabetes.pdf。分类器的总体形状：



细节：



## 四、Refinement

控制树的深度，分别测试最大深度为3和5的分类器

from sklearn.tree import DecisionTreeRegressor

...

def **trainRegressionModel**(x, y, maxDep):

# train the classifier

regr = DecisionTreeRegressor(max\_depth=maxDep)

regr = regr.fit(x, y)

return regr

def **testRegression**(x, y, regr):

# predict

predicts = regr.predict(x)

# calculate the precision

total = 0

match = 0

for predict in predicts :

if ((y[total] - predict)< 0.5) & ((y[total] - predict)> -0.5):

match = match + 1

total = total + 1

print *'predict precision: '*

print(float(match)/total)

def **testIndianData**():

...

print *'max-dep = 3:'*

reg1 = trainRegressionModel(x, y, 3)

testRegression(xc, yc, reg1)

print *'max-dep = 5:'*

reg2 = trainRegressionModel(x, y, 5)

testRegression(xc, yc, reg2)

输出结果：

max-dep = 3:

predict precision:

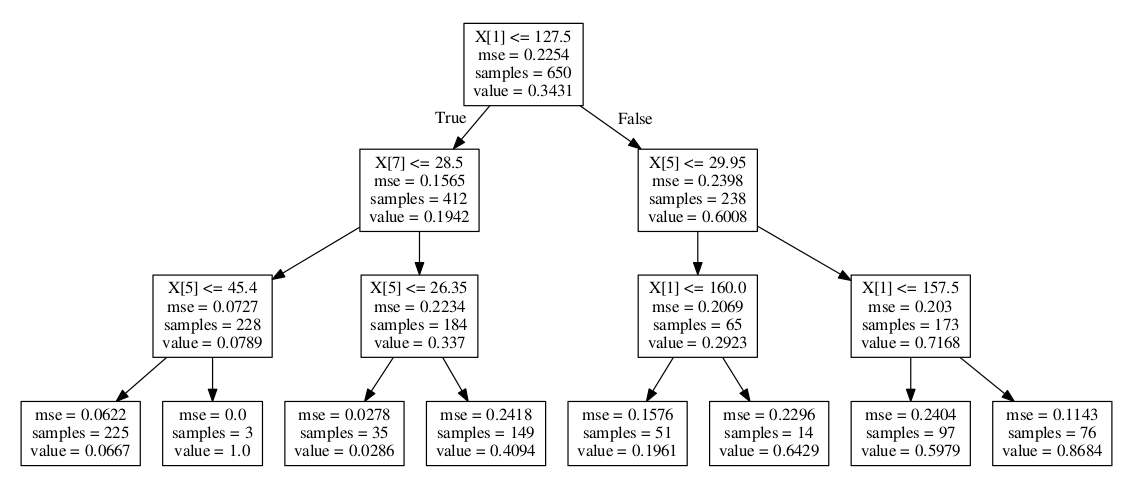
0.762711864407

max-dep = 5:

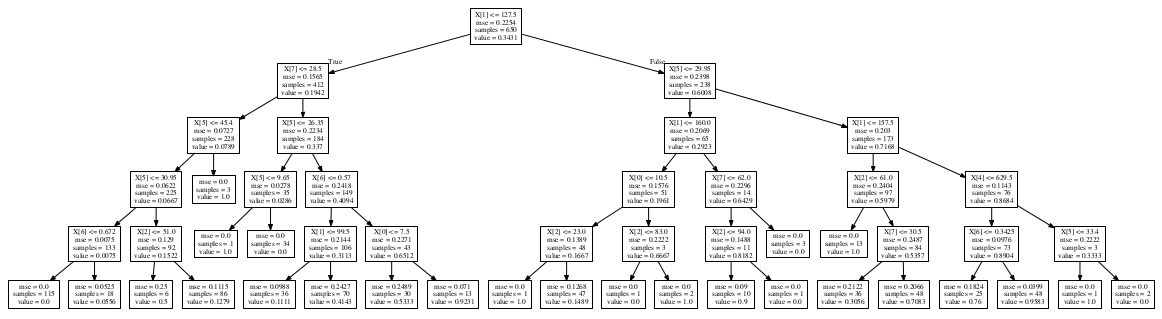
predict precision:

0.745762711864

说明最大深度为3时结果更好（达到76.27%）。可能的原因是限制最大深度避免了模型的过度拟合。



max-depth = 3



max-depth = 5

附：完整代码

from sklearn import tree

from sklearn.externals.six import StringIO

from sklearn.tree import DecisionTreeRegressor

import pydot

def trainModel(x, y):

# train the classifier

clf = tree.DecisionTreeClassifier()

clf = clf.fit(x, y)

return clf

def trainRegressionModel(x, y, maxDep):

# train the classifier

regr = DecisionTreeRegressor(max\_depth=maxDep)

regr = regr.fit(x, y)

return regr

def test(x, y, clf):

# predict

predicts = clf.predict(x)

# calculate the precision

total = 0

match = 0

for predict in predicts :

if predict == y[total] :

match = match + 1

total = total + 1

print 'predict precision: '

print(float(match)/total)

def testRegression(x, y, regr):

# predict

predicts = regr.predict(x)

# calculate the precision

total = 0

match = 0

for predict in predicts :

if ((y[total] - predict)< 0.5) & ((y[total] - predict)> -0.5):

match = match + 1

total = total + 1

print 'predict precision: '

print(float(match)/total)

def plotTree(clf, file):

#visualize decision tree

dot\_data = StringIO()

tree.export\_graphviz(clf, out\_file=dot\_data)

graph = pydot.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_pdf(file)

def testIndianData():

# use the first 650 samples as training set

x = []

y = []

# use sample 651- as testing set

xc = []

yc = []

#import data

i = 0

with open('pima-indians-diabetes.data','r') as f:

for line in f:

arr = map(float,line.split(','))

if i < 650 :

y.append(int(arr[8]))

del arr[8]

x.append(arr)

else :

yc.append(int(arr[8]))

del arr[8]

xc.append(arr)

i = i+1

f.close()

# print x

# print y

print 'max-dep = 3:'

reg1 = trainRegressionModel(x, y, 3)

testRegression(xc, yc, reg1)

plotTree(reg1, "indians-diabetes-maxDep3.pdf")

print 'max-dep = 5:'

reg2 = trainRegressionModel(x, y, 5)

testRegression(xc, yc, reg2)

plotTree(reg2, "indians-diabetes-maxDep5.pdf")