

graphviz插件的安装及使用 – 决策树模型可...

graphviz插件的安装及使用 – 决策树可视化

by 华能信托 王宇韬

更多内容可以参考笔者书籍：

《Python大数据分析与机器学习商业案例实战》第五章

1、决策树模型搭建

首先通过书上第五章的相关代码搭建决策树模型：

```
1 # 模型搭建代码汇总
2 import pandas as pd
3 # 1.读取数据与简单预处理
4 df = pd.read_excel('员工离职预测模型.xlsx')
5 df = df.replace({'工资': {'低': 0, '中': 1, '高': 2}})
6
7 # 2.提取特征变量和目标变量
8 X = df.drop(columns='离职')
9 y = df['离职']
10
11 # 3.划分训练集和测试集
12 from sklearn.model_selection import train_test_split
13 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
14 random_state=123)
15
16 # 4.模型训练及搭建
17 from sklearn.tree import DecisionTreeClassifier
18 model = DecisionTreeClassifier(max_depth=3, random_state=123)
19 model.fit(X_train, y_train)
```

其中用到的数据文件如下：



员工离职预测模型.xlsx
0.5MB

所有代码文件如下：

2、graphviz插件安装与环境变量部署

本小节主要讲解一下graphviz插件的安装与环境变量部署，为之后将决策树模型可视化做准备。

2.1 graphviz插件下载

搭建完决策树模型后，我们可以通过graphviz插件将其可视化呈现出来。首先需要安装一下graphviz插件，其下载地址为：<https://graphviz.gitlab.io/download/>，以Windows版本为例，在下载网站上选择下图框中内容：Stable 2.38 Windows install packages。

Windows

- [Development Windows install packages](#)
- [Stable 2.38 Windows install packages](#)
- [Cygwin Ports](#)* provides a port of Graphviz to Cygwin.
- [WinGraphviz](#)* Win32/COM object (dot/neato library for Visual Basic and ASP).

Mostly correct notes for building Graphviz on Windows can be found [here](#).

然后下载下图所示的msi文件：

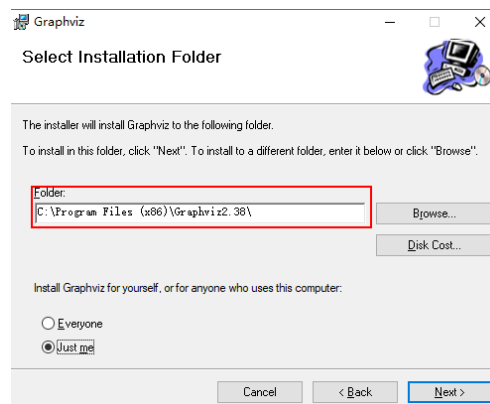
Windows Packages

Note: These Visual Studio packages do not alter the PATH variable or access the registry at all. If you wish to use the command-line interface to Graphviz or are using some other program that calls a Graphviz program, you will need to set the PATH variable yourself.

2.38 Stable Release

- [graphviz-2.38.msi](#)
- [graphviz-2.38.zip](#)

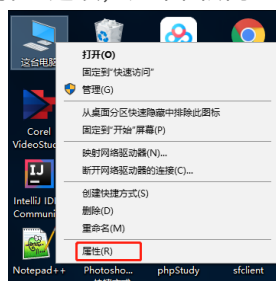
下载完该msi文件后进行安装，注意，要记住下图所示的安装的文件路径，之后进行环境变量部署的时候会用到。



2.2 环境变量部署

安装完graphviz后我们需要进行环境变量部署，所谓环境变量部署，就是把安装的软件部署到整个电脑系统环境中，这样在电脑的各个地方都可以调用配置好的软件。其配置方法如下：

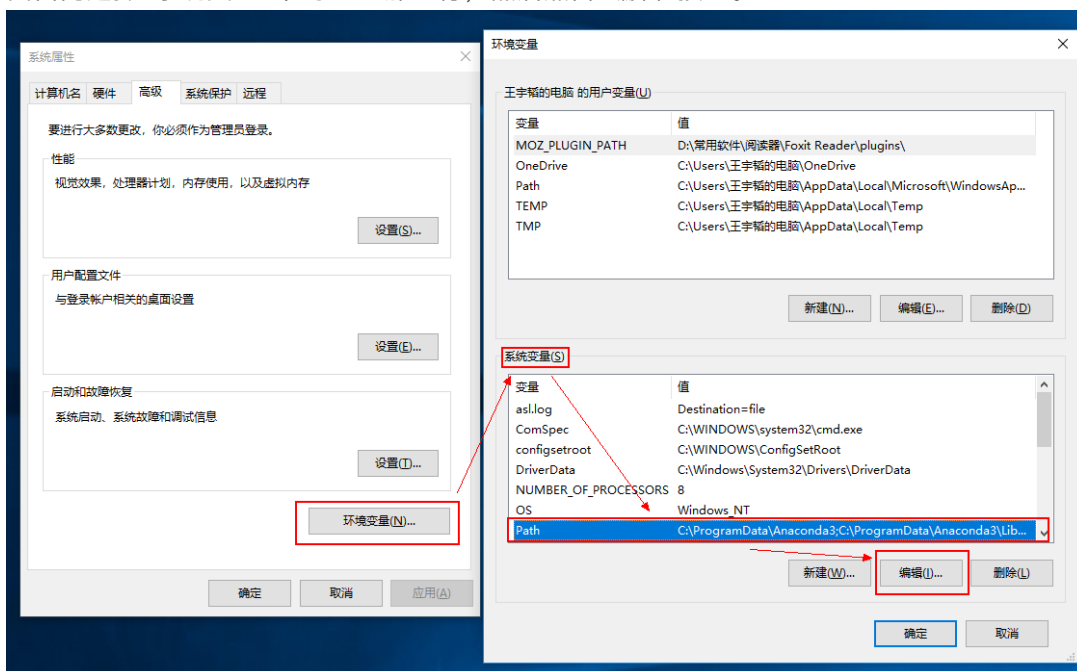
1.右键点击我的电脑，选择“属性”选项，如下图所示：



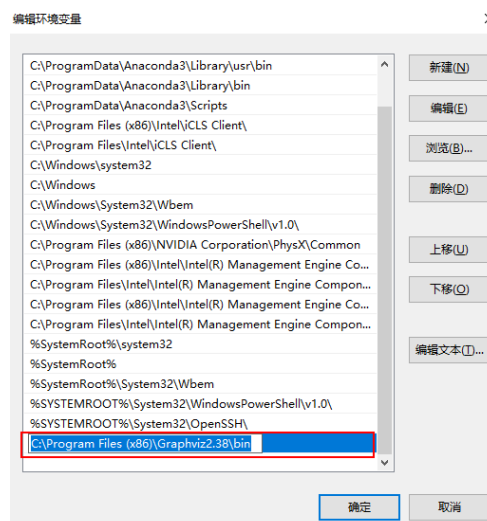
2.在弹出的界面中选择“高级系统设置”，如下图所示：



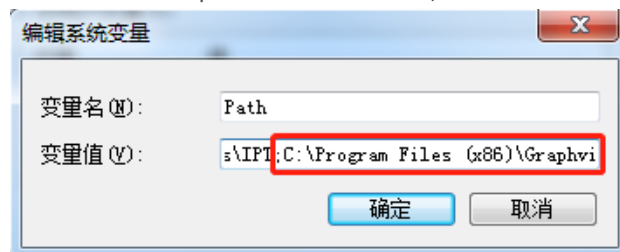
3.如下图所示，在弹出的系统属性界面中，选择“环境变量”，然后在弹出的“环境变量”弹出窗内选择“系统变量”中的Path那一行，然后点击“编辑”按钮。



4.如下图所示，在弹出的界面中，点击“新建”，然后将graphviz安装所在文件夹中bin文件夹添加到其中，最后点击确定即可。环境变量部署就是将软件所在文件路径中的bin文件夹，如这里的“C:\Program Files (x86)\Graphviz2.38\bin”添加到上图所示的Path中，所以当完成这一步后，环境变量也就部署完毕了。



上面演示的是Win10系统的弹出界面，如果是Win7系统，在上一步骤点击“编辑”按钮后，直接在弹出的弹窗变量值那一栏目中，加一个**英文格式**下的分号“;”中，然后添加该bin文件夹路径“C:\Program Files (x86)\Graphviz2.38\bin”即可，如下图所示：

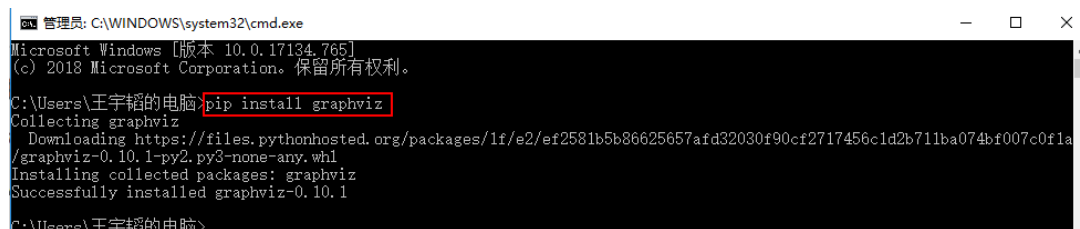


3、在Python中使用graphviz

上面已经安装好graphviz插件，并完成环境变量部署后，我们就可以在Python中使用graphviz将决策树模型可视化输出出来了。

3.1 graphviz库安装

想要在Python中使用graphviz插件，首先还需要在Python中安装相关的库，这里同样是通过PIP安装法进行安装，以Windows系统为例，Win + R组合键调出系统运行框，输入cmd后点击确定，在弹出框内输入pip install graphviz，按一下回车键等待安装结束即可。如下图所示：



3.2 graphviz库的使用

通过如下代码就可以生成一个可视化的决策树模型：

```
1 from sklearn.tree import export_graphviz
2 import graphviz
3 dot_data = export_graphviz(model, out_file=None, class_names=['0', '1'])
4 graph = graphviz.Source(dot_data)
5 graph.render('决策树可视化')
```

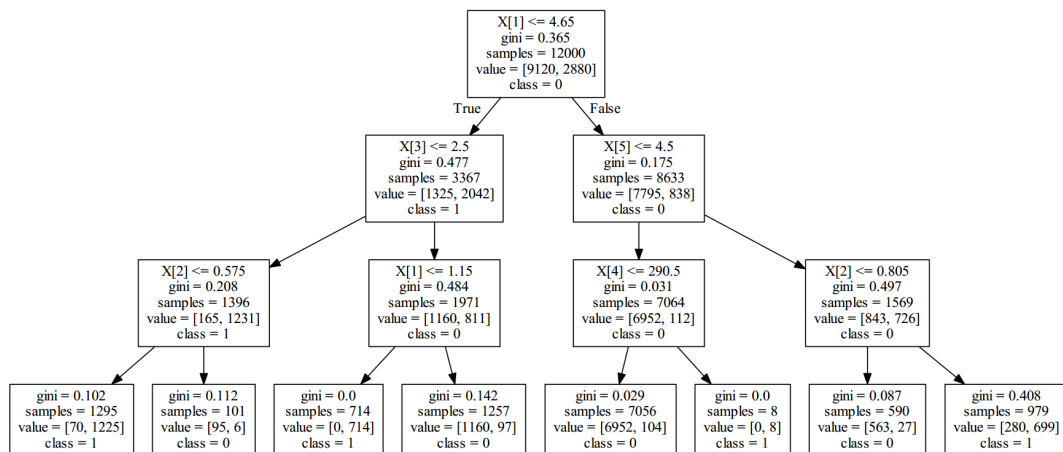
前两行引入使用graphviz的相关库，第三行通过export_graphviz()方法将之前搭建的决策树模型model转换为字符串格式并赋值给dot_data，其中注意需要设定out_file参数为None，这样获得的才是字符串格式，感兴趣的读者可以将dot_data打印出来看下，获得的dot_data如下图所示，里面的内容其实就是之后要可视化的内容。

```

digraph Tree {
node [shape=box] ;
0 [label="X[1] <= 4.65\ngini = 0.365\nsamples = 12000\nvalue = [9120, 2880]\nclass = 0"] ;
1 [label="X[3] <= 2.5\ngini = 0.477\nsamples = 3367\nvalue = [1325, 2042]\nclass = 1"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[2] <= 0.575\ngini = 0.208\nsamples = 1396\nvalue = [165, 1231]\nclass = 1"] ;
1 -> 2 ;
3 [label="gini = 0.102\nsamples = 1295\nvalue = [70, 1225]\nclass = 1"] ;
2 -> 3 ;
4 [label="gini = 0.112\nsamples = 101\nvalue = [95, 6]\nclass = 0"] ;
2 -> 4 ;
5 [label="X[1] <= 1.15\ngini = 0.484\nsamples = 1971\nvalue = [1160, 811]\nclass = 0"] ;
1 -> 5 ;
6 [label="gini = 0.0\nsamples = 714\nvalue = [0, 714]\nclass = 1"] ;
5 -> 6 ;
7 [label="gini = 0.142\nsamples = 1257\nvalue = [1160, 97]\nclass = 0"] ;
5 -> 7 ;

```

第四行代码则是将dot_data转换成可视化的格式，第五行代码则是通过render()方法将图像输出出来，通过上述代码默认是输出一个PDF文件，PDF文件如下图所示：



可以看到里面的内容的确就是上面dot_data文本内容的可视化呈现了。其中X[1]就表示第2个特征变量：满意度，X[3]则表示第4个特征变量：工程数量，X[5]则表示第6个特征变量：工龄；gini则表示该节点的基尼系数；samples则表示该节点中的样本数，比如说第一个节点，也即根节点中的12000也即训练集中的样本数量；value则表示不同种类所占的个数，比如说根节点中value左边的9120则表示非流失客户的数量，2880则表示流失客户的数量，class=0则是认为该节点为未流失节点。

如果通过运行上述代码产生下图所示的报错信息，说明之前的环境变量部署没有部署成功。

```

Traceback (most recent call last):
  File "E:\华小智金融科技团队\丛书\Python金融大数据挖掘与分析全流程实战\第十六章\第十六章汇总\源代码汇总\16.3.3 模型可视化呈现.py", line 31, in <module>
    graph.render("result")
  File "D:\Anaconda\Anaconda\lib\site-packages\graphviz\files.py", line 188, in render
    rendered = backend.render(self.engine, format, filepath, renderer, formatter)
  File "D:\Anaconda\Anaconda\lib\site-packages\graphviz\backend.py", line 183, in render
    run(cmd, capture_output=True, check=True, quiet=quiet)
  File "D:\Anaconda\Anaconda\lib\site-packages\graphviz\backend.py", line 150, in run
    raise ExecutableNotFound(cmd)
graphviz.backend.ExecutableNotFound: failed to execute ['dot', '-Tpdf', '-O', 'result'], make sure the Graphviz executables are on your systems' PATH

```

那么此时我们可以手动在程序里部署环境变量，只要在代码最上方添加如下两行代码即可。其中添加字母r的原因是用来去除文件路径中反斜杠的特殊含义，另一种方法是用两个反斜杠“\\”代替单个斜杠“\”。如果graphviz的安装路径不是下面所示，那么自行修改即可。

```

1 import os
2 os.environ['PATH'] = os.pathsep + r'C:\Program Files
  (x86)\Graphviz2.38\bin'

```

完整代码如下：

```

1 from sklearn.tree import export_graphviz
2 import graphviz
3 import os # 以下这两行是手动进行环境变量配置，防止在本机的环境变量部署失败

```

```

4 os.environ['PATH'] = os.pathsep + r'C:\Program Files
  (x86)\Graphviz2.38\bin'
5 dot_data = export_graphviz(model, out_file=None)
6 graph = graphviz.Source(dot_data)
7 graph.render('决策树可视化')

```

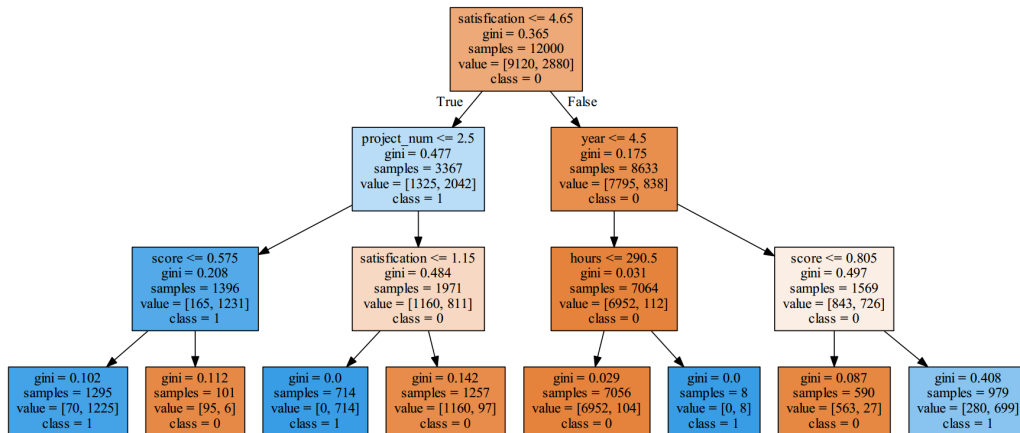
如果想让可视化的图片里的内容更丰富些，可以在`export_graphviz()`的括号中增添一些参数，比如增加`feature_names`参数可以显示特征变量的名称，不过直接使用`graphviz`识别不了中文，所以我们等会先用英文来做演示；增添`class_names`参数则可以显示最后的分类结果，因为中文可能会出现乱码问题，所以我们用字符串'0'和'1'来代替'流失'和'非流失'，注意不能写成数字格式的0和1，因为`class_names`参数中只能设置字符串格式的数据；将`filled`参数设置为`True`还可以给可视化的决策树添加颜色，演示代码如下：

```

1 # 添加名称（feature_names）和填充颜色（filled=True）
2 dot_data = export_graphviz(model, out_file=None, feature_names=['income',
  'satisfaction', 'score', 'project_num', 'hours', 'year'], class_names=
  ['0', '1'], filled=True)

```

此时生成的可视化图形如下图所示：



此时的特征变量就不再试通过`X[4]`这种形式来表示了，而是通过具体我们设置的名称来显示了。又因为我们设置了`class_names`，所以每一个节点多出一个内容叫作`class`，其中0代表着非流失，1代表着流失，其判断依据为`value`中哪个类别占的数量多，则判定为该类别，比如根节点中的`value`，非流失客户为475人，流失客户为325人，那么则判断该节点的类别为0，及非流失。在实际应用中，我们更关心最终叶子节点的分类型别，比如左下角的叶子节点的分类型别为1，及流失，那么如果新来一个客户，通过该决策树模型分到左下角的叶子节点的话，那么则判断其为流失客户。

上面我们生成的可视化图形已经比较完善了，但是`graphviz`本身是没有办法识别中文的，所以如果把上面代码中的`feature_names`或者`class_names`参数中的内容写成中文的话，在最后生成图形中会出现中文乱码的问题。关于中文乱码的问题我们将在下一小节解决。

3.3 中文乱码问题解决办法

这一小节我们重点解决`graphviz`因为不能识别中文，导致生成的中文乱码问题。步骤如下：

1.引入相关库并在程序中部署好环境变量

```

1 from sklearn.tree import export_graphviz
2 import os # 以下这两行是手动进行环境变量配置，防止在本机的环境变量部署失败
3 os.environ['PATH'] = os.pathsep + r'C:\Program Files
  (x86)\Graphviz2.38\bin'

```

这里不需要引入graphviz库，因为等会生成图形的方式不再是利用上一小节graphviz.Source(dot_data)来进行生成了，这个稍后再讲。此外，这里通过手动的方式进行了环境变量部署，也是为了防止之前在本机的环境变量部署失败。










2.生成dot_data

```
1 dot_data = export_graphviz(model, out_file=None,  
    feature_names=X_train.columns, class_names=['不流失', '流失'],  
    rounded=True, filled=True)
```

这里通过export_graphviz()方法生成dot_data，此时的dot_data是一个字符串类型的数据，里面的内容则是之后要进行可视化的内容，这个在上一小节也提到过。这里注重讲下export_graphviz()方法里设置的参数：首先model表示之前训练的模型，我们的目的就是将其转换成字符串的格式然后再转换成图形；out_file参数是用来设置生成的内容为字符串的；feature_names则是用来设置特征变量的名称，其中X_train.columns则是测试训练集的表头名称，也即这个决策树模型的特征变量名称们，感兴趣的读者可以将其打印出来，效果如下，可以看到这里的表头全是中文，如果还用之前的方式进行可视化的话，其中的中文会出现乱码现象。

```
1 Index(['收入', '年龄', '性别', '历史授信额度', '历史流失次数'],  
    dtype='object')
```

class_names则是设定最后的分类结果，这里我们直接设置成中文的‘不流失’和‘流失’，之后会讲如何处理中文乱码问题；rounded参数需要设置成True，这样之后设置中文格式才会生效；最后的filled参数设置为True可以使得生产的决策树有颜色。

solid	dashed	dotted	bold	rounded	diagonals	filled	striped	wedged
								

3.将dot_data写入到txt文件中

通过如下代码可以将之前的dot_data字符串类型的数据写到txt文件中。

```
1 f = open('dot_data.txt', 'w')  
2 f.write(dot_data)  
3 f.close()
```

之后我们将利用这里生成的dot_data.txt文件来生成新的决策树文本文件并修改其中的中文配置信息从而实现通过graphviz能够输出中文的目的。此时生成dot_data.txt文件如下图所示，可以看到里面的内容的确是之后可视化图形要展示的内容了。


```
dot_data.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
digraph Tree {
node [shape=box, style="filled, rounded", color="black", fontname=helvetica];
edge [fontname=helvetica];
0 [label="满意度 <= 4.65\ngini = 0.365\nsamples = 12000\nvalue = [9120,
2880]\nnclass = 不离职", fillcolor="#eda978"];
1 [label="工程数量 <= 2.5\ngini = 0.477\nsamples = 3367\nvalue = [1325,
2042]\nnclass = 离职", fillcolor="#b9ddf6"];
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"];
2 [label="考核得分 <= 0.575\ngini = 0.208\nsamples = 1396\nvalue = [165,
1231]\nnclass = 离职", fillcolor="#54aae8"];
```

4.修改字体设置

通过如下代码，我们可以修改原来dot_data.txt中的字体及编码格式，并将其另存为一个新的dot_data_new.txt文件，为之后解决中文乱码问题做铺垫。

```
1 import re
2 f_old = open('dot_data.txt', 'r')
3 f_new = open('dot_data_new.txt', 'w', encoding='utf-8')
4 for line in f_old:
5     if 'fontname' in line:
6         font_re = 'fontname=(.*)'
7         old_font = re.findall(font_re, line)[0]
8         line = line.replace(old_font, 'SimHei')
9     f_new.write(line)
10 f_old.close()
11 f_new.close()
```

这里首先引入正则表达式库re，为之后的替换字体做准备；然后通过open()函数读取原来的dot_data.txt中的文本内容，其中'r'表示以读取模式打开txt文件，我们将文件内容赋值给f_old变量；之后通过open()函数新建一个dot_data_new.txt文件，其中'w'表示以写入模式打开txt文件，并且每次写入时都会把原有内容清除，encoding参数为编码方式，这里设置为能够支持中文显示的utf-8编码，这个编码方式的设置对解决中文乱码问题而言很重要。

下面的几行代码这里单独拎出来讲解一下：

```
1 for line in f_old:
2     if 'fontname' in line:
3         font_re = 'fontname=(.*)'
4         old_font = re.findall(font_re, line)[0]
5         line = line.replace(old_font, 'SimHei')
6     f_new.write(line)
```

这几行代码的目的就是用来替换原来dot_data.txt的默认的字体格式，替换成SimHei也就是黑体格式，如下图所示：

```
dot_data.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
digraph Tree {
node [shape=box, style="filled, rounded", color="black", fontname=helvetica];
edge [fontname=helvetica];
0 [label="历史违约次数 <= 0.5\ngini = 0.482\nsamples = 800\nvalue = [475,
325]\nnclass = 不违约", fillcolor="#e5813951"];
```


这里首先通过for循环语句遍历f_old，也即dot_data.txt中的每一行，然后通过if判断语句来查看每一行中是否有fontname内容，其中fontname的中文意思就是字体名字，对于含有fontname的行，我们通过正则表达式的非贪婪匹配(.*?)将默认的字体找到，并通过replace()函数将旧的字体替换成我们想设置的字体，比如这里设置的SimHei，也即黑体字体。最后将这些替换完的新的行写入到新的txt文件f_new中。

最后生成完新的f_new文件后，我们就可以通过close()函数将f_old和f_new关闭。

补充知识点：这里的SimHei是黑体的英文翻译，如果想采用其他字体，可参考下面的字体英文对照表：

黑体	SimHei
微软雅黑	Microsoft YaHei
新宋体	NSimSun
新细明体	PMingLiU
细明体	MingLiU
仿宋	FangSong
楷体	KaiTi

5.生成可视化文件

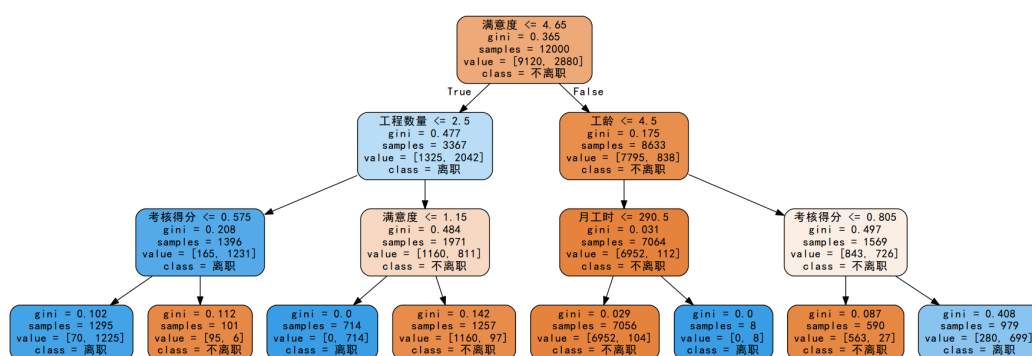
生成完新的重新设置编码格式及字体的dot_data_new.txt文件后，我们就可以通过graphviz插件生成可视化的文件了，代码如下：

```
1 os.system('dot -Tpdf dot_data_new.txt -o 决策树模型.pdf')
```

这里是通过调用os.system()系统函数来使用graphviz插件，其中-Tpdf表示生成PDF格式的文件，如果将其改成-Tpng生成PNG格式的图片文件，代码如下：

```
1 os.system('dot -Tpng dot_data_new.txt -o 决策树模型.png')
```

最后的生成效果如下图所示，可以看到已经可以显示中文了。



所有代码整理如下：

```

1 from sklearn.tree import export_graphviz
2 import graphviz
3 import os # 以下这两行是手动进行环境变量配置，防止在本机环境的变量部署失败
4 os.environ['PATH'] = os.pathsep + r'C:\Program Files
  (x86)\Graphviz2.38\bin'
5
6 # 生成dot_data
  
```

```
7 dot_data = export_graphviz(model, out_file=None,
  feature_names=X_train.columns, class_names=['不流失', '流失'],
  rounded=True, filled=True) # rounded和字体有关, filled设置颜色填充
8
9 # 将生成的dot_data内容导入到txt文件中
10 f = open('dot_data.txt', 'w')
11 f.write(dot_data)
12 f.close()
13
14 # 修改字体设置, 避免中文乱码!
15 import re
16 f_old = open('dot_data.txt', 'r')
17 f_new = open('dot_data_new.txt', 'w', encoding='utf-8')
18 for line in f_old:
19     if 'fontname' in line:
20         font_re = 'fontname=(.*)?'
21         old_font = re.findall(font_re, line)[0]
22         line = line.replace(old_font, 'SimHei')
23     f_new.write(line)
24 f_old.close()
25 f_new.close()
26
27 # 以PNG的图片形式存储生成的可视化文件
28 os.system('dot -Tpng dot_data_new.txt -o 决策树模型.png')
29 print('决策树模型.png已经保存在代码所在文件夹! ')
30
31 # 以PDF的形式存储生成的可视化文件
32 os.system('dot -Tpdf dot_data_new.txt -o 决策树模型.pdf')
33 print('决策树模型.pdf已经保存在代码所在文件夹! ')

```