1. 前言

2. 问题起源

2.1. 对付数值型类别变量

2.2. 对付字符串型类别变量

2.3. 无用的尝试

3. 另一种解决方案

4. 参考资料

1. 前言

这几天埋头撰写『优雅高效地数据挖掘——基于Python的sklearn\_pandas库』 一文，其中有一部分涉及如何批量并行地进行特征二值化，在此过程中发现了

scikit-learn (以下简称 sklearn)中，二值化函数存在一些坑，跟 sklearn\_pandas 的作者在 github 上交流过，在此总结一下，做个记录

所涉及到的几种 sklearn 的二值化编码函数：OneHotEncoder(), LabelEncoder(), LabelBinarizer(), MultiLabelBinarizer()

2. 问题起源

首先造一个测试数据

import pandas as pdfrom sklearn.preprocessing import OneHotEncoderfrom sklearn.preprocessing import LabelEncoderfrom sklearn.preprocessing import LabelBinarizerfrom sklearn.preprocessing import MultiLabelBinarizertestdata = pd.DataFrame({'pet': ['cat', 'dog', 'dog', 'fish'],                         'age': [4 , 6, 3, 3],                         'salary':[4, 5, 1, 1]})

这里我们把 pet、age、salary 都看做类别特征，所不同的是 age 和 salary 都是数值型，而 pet 是字符串型。我们的目的很简单:

把他们全都二值化，进行 one-hot 编码

2.1. 对付数值型类别变量

对 age 进行二值化很简单，直接调用 OneHotEncoder

OneHotEncoder(sparse = False).fit\_transform( testdata.age ) # testdata.age 这里与 testdata[['age']]等价

然而运行结果是 array([[ 1.,  1.,  1.,  1.]])，这个结果是错的，从 Warning

信息中得知，原因是 sklearn 的新版本中，OneHotEncoder 的输入必须是 2-D

array，而 testdata.age 返回的 Series 本质上是 1-D array，所以要改成

OneHotEncoder(sparse = False).fit\_transform( testdata[['age']] )

我们得到了我们想要的：

array([[ 0.,  1.,  0.],       [ 0.,  0.,  1.],       [ 1.,  0.,  0.],       [ 1.,  0.,  0.]])

可以用同样的方法对 salary 进行 OneHotEncoder,

然后将结果用 numpy.hstack() 把两者拼接起来得到变换后的结果

a1 = OneHotEncoder(sparse = False).fit\_transform( testdata[['age']] )a2 = OneHotEncoder(sparse = False).fit\_transform( testdata[['salary']])final\_output = numpy.hstack((a1,a2))

不过这样的代码略显冗余，既然 OneHotEncoder() 可以接受 2-D array 输入，那我们可以写成这样

OneHotEncoder(sparse = False).fit\_transform( testdata['age', 'salary'])

结果为

array([[ 0.,  1.,  0.,  0.,  1.,  0.],       [ 0.,  0.,  1.,  0.,  0.,  1.],       [ 1.,  0.,  0.,  1.,  0.,  0.],       [ 1.,  0.,  0.,  1.,  0.,  0.]])

有时候我们除了得到最终编码结果，还想知道结果中哪几列属于 age 的二值化编码，哪几列属于 salary 的，这时候我们可以通过 OneHotEncoder() 自带的 feature\_indices\_ 来实现这一要求，比如这里 feature\_indices\_ 的值是[0,

3, 6]，表明 第[0:3]列是age的二值化编码，[3:6]是salary的。更多细节请参考

sklearn 文档，

2.2. 对付字符串型类别变量

遗憾的是OneHotEncoder无法直接对字符串型的类别变量编码，也就是说OneHotEncoder().fit\_transform(testdata[['pet']])这句话会报错(不信你试试)。已经有很多人在

stackoverflow 和 sklearn 的 github issue 上讨论过这个问题，但目前为止的 sklearn 版本仍没有增加OneHotEncoder对字符串型类别变量的支持，所以一般都采用曲线救国的方式：

方法一 先用 LabelEncoder() 转换成连续的数值型变量，再用 OneHotEncoder() 二值化

方法二 直接用 LabelBinarizer() 进行二值化

然而要注意的是，无论 LabelEncoder() 还是 LabelBinarizer()，他们在 sklearn 中的设计初衷，都是为了解决标签 y 的离散化，而非输入 X， 所以他们的输入被限定为 1-D array，这恰恰跟 OneHotEncoder() 要求输入 2-D array 相左。所以我们使用的时候要格外小心，否则就会出现上面array([[

1.,  1.,  1.,  1.]])那样的错误

# 方法一: LabelEncoder() + OneHotEncoder()a = LabelEncoder().fit\_transform(testdata['pet'])OneHotEncoder( sparse=False ).fit\_transform(a.reshape(-1,1)) # 注意: 这里把 a 用 reshape 转换成 2-D array# 方法二: 直接用 LabelBinarizer()LabelBinarizer().fit\_transform(testdata['pet'])

这两种方法得到的结果一致，都是

array([[ 1.,  0.,  0.],       [ 0.,  1.,  0.],       [ 0.,  1.,  0.],       [ 0.,  0.,  1.]])

正因为LabelEncoder和LabelBinarizer设计为只支持

1-D array，也使得它无法像上面 OneHotEncoder 那样批量接受多列输入，也就是说LabelEncoder().fit\_transform(testdata[['pet',

'age']])会报错。

2.3. 无用的尝试

然而执着如我怎会就此放弃，我又仔细翻了翻 sklearn 的 API 接口，果然发现有个叫 MultiLabelBinarizer() 的，看着似乎可以解决这个问题，于是尝试了一下

MultiLabelBinarizer().fit\_transform(testdata[['age','salary']].values)

输出结果如下

array([[0, 0, 1, 0, 0],       [0, 0, 0, 1, 1],       [1, 1, 0, 0, 0],       [1, 1, 0, 0, 0]])

结果咋一看毫无问题，再仔细一看，被打脸！MultiLabelBinarizer并没有分别对每列进行

one-hot 编码，而是将这几列的取值看做一个整体，每行样本都被去重了，所以结果中第一行只有一个 1 ，因为 age 和 salary 第一行取值都是 4， MultiLabelBinarizer 默认这行样本只有一个类别

4 。。。。。。

3. 另一种解决方案

其实如果我们跳出 scikit-learn， 在 pandas 中可以很好地解决这个问题，用 pandas 自带的get\_dummies函数即可

pd.get\_dummies(testdata,columns=testdata.columns)

结果正是我们想要的

age\_3   age\_4   age\_6   pet\_cat pet\_dog pet\_fish    salary\_1    salary\_4    salary\_50   0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0 0.01   0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.02   1.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.03   1.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 0.0

get\_dummies的优势在于:

本身就是 pandas 的模块，所以对 DataFrame 类型兼容很好

不管你列是数值型还是字符串型，都可以进行二值化编码

能够根据指令，自动生成二值化编码后的变量名

这么看来，我们找到最完美的解决方案了？ No！get\_dummies千般好，万般好，但毕竟不是 sklearn

里的transformer类型，所以得到的结果得手动输入到 sklearn 里的相应模块，也无法像

sklearn 的transformer一样可以输入到pipeline中

进行流程化地机器学习过程。更重要的一点

get\_dummies 不像 sklearn 的 transformer一样，有 transform方法，所以一旦测试集中出现了训练集未曾出现过的特征取值，简单地对测试集、训练集都用 get\_dummies 方法将导致数据错误

---------------------

本文来自 Han\_Sen 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/hshuihui/article/details/53259710?utm\_source=copy