**智能风控学习笔记五-变量编码一**

大家好，这里是Python知识学堂。这次的推文我们一起来讨论一下变量的编码，并使用Python代码来实现其中的知识点。

**文章的大体结构**

1.为什么要进行编码

2.One-hot编码

3.哑变量编码

4.标签编码

5.WOE编码

**一、为什么要进行编码**

我们知道计算机只能处理一些机构化的数据，非结构化的数据需要转换为机构化的数据才能做进一步的处理。结构化的数据如一张表，表中的一行代表一个样本，表中的列表示变量；非结构化的数据比如说图像，文档等。一般来说我们在处理图像数据的时候将图像转换为像素点进行处理，也就是转换为可以让计算机处理的类型。另外比如说我们处理文本数据的时候需要进行向量化的处理。

​ 很显然我们会想到一种变量如性别变量，有男、女、未知三种可能。而计算机并不认识也不知道怎么处理这三种状态，因此就需要进行编码。

​ 另外更重要的就是建模算法只能处理数值型的变量，这也是为什么我们需要进行编码的原因。

**二、One-Hot编码**

​ 首先One-hot编码是针对离散变量（属性数据）进行编码的一种技术，其又叫做一位有效编码。假设离散变量有N中可能，那么One-Hot编码就使用N位状态码来进行编码。每一个被编码的变量只有一位数据有效，整体上我们可以把编码之后的结果看做是一个N维的向量，接下来我们举一个例子来看一下。假设我们对性别这个离散变量进行编码，那么就有：

|  |  |
| --- | --- |
| 性别取值 | One-Hot编码结果 |
| 男 | 1 0 0 |
| 女 | 0 1 0 |
| 未知性别 | 0 0 1 |

可以看出上述离散变量性别有3中取值可能，所以对应的M = 3, 所以可以使用3位二进制的向量0和1进行变量的可能取值表示即可。

比如现在有五种学历信息：小学，中学，大学，硕士，博士。我们可以这样进行one-hot编码：

|  |  |
| --- | --- |
| 学历取值 | One-Hot编码结果 |
| 小学 | 1 0 0 0 0 |
| 中学 | 0 1 0 0 0 |
| 大学 | 0 0 1 0 0 |
| 硕士 | 0 0 0 1 0 |
| 博士 | 0 0 0 0 1 |

我在学习的时候有一个疑问，表示成这种格式有什么好处，接下来我们总结一下One-Hot编码的优缺点：

**优点：**

将离散变量映射到欧式空间，会使得特征之间的距离（一般为欧氏距离或者其他距离）更加的合理性，这个合理性能用来做什么呢？比如可以运行在机器学习算法（聚类，分类）中计算两个向量之间的相似度（相似度通过距离来进行量化），欧式距离的计算公式如下:

​

根据公式可知上述的两两性别变量之间（去除未知性别变量）的举例为。

**缺点：**

缺点也是比较明显，就是离散特征的取值有很多种可能的时候，这种方式会造成大量的空间浪费。这个怎么理解呢？大家可以看到上述One-Hot编码的结果中存在很多的0，这就造成了一个新的问题-编码后数据的高度稀疏性。一般针对稀疏性的问题采用稀疏编码或降维的方式（PCA）。

使用Python进行One-Hot编码十分简单，我们可以直接使用Pandas中的get\_dummies（）函数。

import pandas as pd

df = pd.DataFrame(

[

['男'],

['女'],

['未知']

]

)

df.columns = ['gener']

after = pd.get\_dummies(df)

print(after)

# 输出如下

gener\_女 gener\_未知 gener\_男

0 0 0 1

1 1 0 0

2 0 1 0

```

当然了，我们也可以使用sklearn中的函数实现：

import pandas as pd

from sklearn.preprocessing import OneHotEncoder

df = pd.DataFrame(

[

['男'],

['女'],

['未知']

]

)

enc = OneHotEncoder()

print(enc.fit(df))

print(enc.transform(df).toarray())

# 输出如下：

OneHotEncoder()

[[0. 0. 1.]

[1. 0. 0.]

[0. 1. 0.]]

**三、哑变量编码**

知道了独热变量之后我们来了解哑变量编码（dummy编码）就非常简单了，哑变量变量和One-hot编码和其实在很大程度上是相似的，只是哑变量将其中一个的编码状态位去除。我们举例来看一下，我们以上述性别的例子来看：

|  |  |
| --- | --- |
| 性别取值 | dummy编码结果 |
| 男 | 1 0 |
| 女 | 0 1 |
| 未知性别 | 0 0 |

很显然使用dummy编码之后也能很好的进行编码。

至于他们之间的区别也非常的明显，这里我就给学习的小伙伴自己解决了。

现在我们来看一下使用Python怎么实现dummy编码：

import pandas as pd  
df=pd.DataFrame(  
 [  
 ['男'],  
 ['女'],  
 ['未知']  
 ],columns=['性别'])  
  
df1=pd.get\_dummies(df['性别'],drop\_first=True)  
print(df1)

#结果如下：

未知 男

0 0 1

1 0 0

2 1 0

**四、标签编码**

学习了上述几种编码之后，我们可以看出这种编码不能看出变量的一种等级关系。比如说学历关系中：博士>硕士>本科>高中。这个时候我们就需要使用到标签编码了。

我们以学历举例标签编码的结果为：

|  |  |
| --- | --- |
| 学历取值 | 标签编码结果 |
| 小学 | 1 |
| 中学 | 2 |
| 大学 | 3 |
| 硕士 | 4 |
| 博士 | 5 |

这里要注意的是：我们这里只是简单的说明标签编码的案例，所以就直接进行1,2,3,4,5个简单的编码结果。当然了在实际生产环境上进行编码的时候，要结合实际的业务进行合适的编码。比如：

|  |  |
| --- | --- |
| 学历取值 | 标签编码结果 |
| 小学 | 1 |
| 中学 | 10 |
| 大学 | 50 |
| 硕士 | 70 |
| 博士 | 100 |

我们来使用Python来实现这个标签编码：

from sklearn import preprocessing  
data=['小学','高中','本科','硕士','博士']  
encode=preprocessing.LabelEncoder()  
*#训练LabelEncoder*encode\_train=encode.fit(['小学','高中','本科','硕士','博士'])  
result=encode\_train.transform(data)  
  
print(result)

**五、WOE编码**

WOE编码是评分卡中最常见的一种数据编码方式，考虑到WOE的内容较多，我们将在下期推文中和大家一起学习。

**六、总结**

本期内容和大家一起学习了在智能风控中最为常见的数据编码方式，介绍了One-Hot编码、dummy编码、标签编码。严格意义上来说这些编码方式是属于无监督编码方式的，之后学习的WOE编码是一种有监督的编码方式。