# 第五章：分类与回归实验

## 5.1 处理混合类型的数据集

完成数据挖掘任务前，需要先处理数据集时往往会遇到混合类型的数据。一个简单的思路是：首先把不同类型的属性区分开，然后单独的进行处理，再合并创建一个统一的数据集。下面介绍几个例子，它们对分类属性进行one-hot编码处理，对数值属性进行规范化。重要的是，用**ColumnTransformer**将它们集中处理，非常简便。

1. **sklearn.preprocessing.OneHotEncoder**

对类别属性进行one-hot编码

class sklearn.preprocessing.OneHotEncoder(\*, categories='auto', drop=None, sparse='deprecated', sparse\_output=True, dtype=<class 'numpy.float64'>, handle\_unknown='error', min\_frequency=None, max\_categories=None)[source]

主要参数如下：

（1）categories：‘auto’或者一个属性列表, default=’auto’

（2）drop：{‘first’, ‘if\_binary’} or an array-like of shape (n\_features,), default=None

设定一种方法来丢弃one-hot编码后产生的新的属性中的某一个。因为存在完全多重共线性，可能会导致后面的任务出现偏差。有下面三种选择：

None：默认值，不删除。

‘first’：丢弃第一个新属性。

‘if\_binary’：如果原始属性有两个属性值，丢弃第一个属性值。其余情况，即原始属性有两个以上的属性值，保留所有新属性完整。

（3）sparse\_output：bool值，默认为True。为True时返回稀疏矩阵，否则返回数组

（4）handle\_unknown：遇到未知值的处理方法，有三种选择

‘error’：产生出错信息，这是默认值。

‘ignore’ : 设置为0。

‘infrequent\_if\_exist’：设置成最不频繁的少数属性值。

常用的两个函数是：

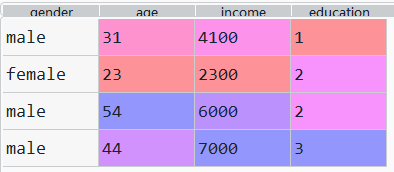
* fit(X[, y])：用一个数据集X去拟合一个OneHotEncoder的对象。其含义是，一个OneHotEncoder数据对象再数据集上拟合是在学习分类属性的特征，包括属性值等信息。然后就可以处理具有相同属性的其他数据集。
* transform(X)：在OneHotEncoder的对象拟合了一个数据集后，使用该方法对数据集X进行one-hot编码。返回的结果是ndarray的spare matrix数据对象。
* fit\_transform(X[, y])：用一个数据集X去拟合一个OneHotEncoder的对象，并把X进行one-hot编码。fit\_transform函数看作是把fit和transform两个函数集成到了一起。

使用OneHotEncoder的基本步骤是，

* 首先准备好要处理的数据集，它可以是numpy或pandas的数据对象。它应该都是类别属性。如果包含数值属性，会被当做类别属性处理。
* 创建一个OneHotEncoder的对象。后面对数据集进行one-hot编码时，就是使用该对象。
* 用创建的OneHotEncoder的对象去拟合一个数据集。其目的是了解一个数据集的类别属性和属性值。然后，后面就可以用改对象，去对其他相同结果的数据集进行one-hot编码。一个典型的应用场景是在对训练集中的类别属性进行拟合，然后可以对测试集进行相同的处理。拟合可以用fit函数，或fit\_transform函数。
* 用拟合后的OneHotEncoder的对象去处理数据集。采用transform函数。当然fit\_transform函数可以拟合完成后就转换数据集。

**练习5.1：基本的one-hot编码**

本节使用一个简单的职工信息数据集faculty.csv，只有四条记录。



其中属性gender和education是分类属性。属性age和income是数值属性。下面使用OneHotEncoder对分类属性进行one-hot编码时，需要单独对分类属性进行处理，例如：

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.preprocessing import OneHotEncoder  df = pd.read\_csv('data/faculty.csv')  cols = ['gender','education']  df2 = df[cols]  enc = OneHotEncoder()  enc.fit(df2)  X=enc.transform(df2).toarray() |

上述代码创建了一个OneHotEncoder对象，然后在只有分类属性的数据帧df2上拟合数据。再将该数据帧转换成ndarray数据结构。

当然，上面拟合和转换是两个步骤。也可以用fit\_transform函数一次性的拟合和转换。其返回的结果是一个scipy的sparse matrics数据结构。调用其toarry()函数可以转换成numpy的ndarray数据结构。

从上面的操作可以看出，如果要训练一个模型，还需要把得到的one-hot编码后的X和原先数据帧中的数值属性进行合并。这样的操作比较复杂，下面再介绍几个函数，通过创建pipeline函数可以方便进行数据集处理的操作。

1. **sklearn.compose.** **ColumnTransformer**

创建一个可以应用在pandas数据帧或者numpy的array数据集上的部分列上的转换器（Transformer）。它非常适合处理混合类型数据的数据集。

class sklearn.compose.ColumnTransformer(transformers, \*, remainder='drop', sparse\_threshold=0.3, n\_jobs=None, transformer\_weights=None, verbose=False, verbose\_feature\_names\_out=True)

创建转换器对象时，它的参数是包含多个元组的列表。每个元组代表一种处理，它的内容是

(name, transformer, columns)

Name是给当前的处理分配的名称

Transformers 当前的处理器

Columns表示在哪些列上处理，它是一个列名的列表。

和OneHotEncoder一样**ColumnTransformer**有三个主要方法。具体内容和OneHotEncoder一样。

fit(X[, y])，transform(X)，fit\_transform(X[, y])

**练习5.2： 使用ColumnTransformer处理混合类型数据集**

将练习5.1中的数据集数值属性进行规范化，类别属性进行one-hot编码。

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.compose import ColumnTransformer  from sklearn.preprocessing import OneHotEncoder, StandardScaler  from sklearn.pipeline import make\_pipeline  from sklearn.preprocessing import OneHotEncoder  df = pd.read\_csv('data/faculty.csv')  numeric\_features = ["age", "income"]  categorical\_features = ['gender','education']  preprocessor = ColumnTransformer(  [  ("num", StandardScaler(),numeric\_features),  ("cat",  OneHotEncoder(handle\_unknown="ignore"),  categorical\_features,  ),  ],  verbose\_feature\_names\_out=False,  )  X=preprocessor.fit\_transform(df) |

上面的程序创建了一个ColumnTransformer类的对象preprocessor。它包括了两个属性的处理：对数值属性进行零均值规范化，对类别属性one-hot编码。

1. **自动按照属性的类型分别进行不同的预处理**

例子5.2中，预先有用户给出了哪些属性是数值型，哪些是类别型。如果属性很多时如此会比较麻烦。此时可以用make\_column\_selector函数，根据正则表达式自动选择不同属性。

**sklearn.compose.make\_column\_selector** sklearn.compose.make\_column\_selector(pattern=None, \*, dtype\_include=None, dtype\_exclude=None)

patter是正则表达式，设定按照属性的名词来信选择属性。如果不给出，则不会基于正则表达式来选择属性。

dtype\_include：按照属性类型来选择一些属性

dtype\_exclude：按照属性类型来排除一些属性

在使用时，该函数通常作为创建ColumnTransformer类的对象时的参数。

**练习5.3：使用ColumnTransformer和make\_column\_selector处理混合类型数据集**

下面代码把make\_column\_selector嵌入到ColumnTransformer类创建对象的过程中参数，自动根据属性类型来选择属性。

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.compose import ColumnTransformer  from sklearn.preprocessing import OneHotEncoder, StandardScaler  from sklearn.pipeline import make\_pipeline  from sklearn.preprocessing import OneHotEncoder  from sklearn.compose import make\_column\_selector  df = pd.read\_csv('data/faculty.csv')  df['education'] = df['education'].astype(str)  preprocessor = ColumnTransformer(  [  ("num",  StandardScaler(),  **make\_column\_selector**(dtype\_include=np.number)),  ("cat",  OneHotEncoder(handle\_unknown="ignore"),  **make\_column\_selector**(dtype\_include=object),  ),  ]  )  X=preprocessor.fit\_transform(df) |

在练习5.1的数据集中，属性education应该是类别属性，但使用pandas的read\_csv读入时，自动转换成了数值属性。所以上面的代码首先给它转换会str类型。

df['education'] = df['education'].astype(str)

在创建ColumnTransformer的转换器时，其参数是用多个元组表示的多个处理步骤。每个处理步骤是一个元组，例如

("num", StandardScaler(), make\_column\_selector(dtype\_include=np.number)),

表示该处理步骤是用make\_column\_selector(dtype\_include=np.number)选择数值型的属性，进行零-均值规范化处理，该处理步骤的名词是num。

("cat", OneHotEncoder(handle\_unknown="ignore"), make\_column\_selector(dtype\_include=object) )

这一处理步骤将类型是object，即不是数值类型的都看作是类别属性。然后进行one-hot编码。其中handle\_unknown="ignore"表示，one-hot编码时，遇到不知道的属性值，则忽略。

## 5.2 在混合类型数据集上建立SVM分类模型

在《数据挖掘》教材的6.6.2节使用SVM在Universal Bank数据集上建立分类模型时，将类别属性用数值表示，将其看作了数值属性。本节将采用one-hot编码处理类别属性。并采用accuracy, balanced accuracy和F1指标评价模型性能。

**练习5.4： Universal Bank数据集的SVM模型**

Universal Bank数据集上有4个类别属性：'Securities Account', 'CD Account', 'Online', 'CreditCard'。它们将被one-hot编码。数值类型属性将被零-均值规范化。'Education'是序数属性，我们这里当作是数值属性。

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  from sklearn.pipeline import make\_pipeline  from sklearn.preprocessing import StandardScaler  from sklearn.svm import SVC, NuSVC  from sklearn.compose import ColumnTransformer  from sklearn.compose import make\_column\_selector  from sklearn.preprocessing import OneHotEncoder  df=pd.read\_csv('data/UniversalBank.csv')  y = df['Personal Loan']  df = df.drop(['ID', 'ZIP Code','Personal Loan'], axis=1)  cat\_attr = ['Securities Account', 'CD Account', 'Online', 'CreditCard']  for attr in cat\_attr:  df[attr] = df[attr].astype(str)  preprocessor = ColumnTransformer(  [  ("num",  StandardScaler(),  make\_column\_selector(dtype\_include=np.number)),  ("cat",  OneHotEncoder(handle\_unknown="ignore"),  make\_column\_selector(dtype\_include=object),  ),  ]  )  X=preprocessor.fit\_transform(df)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  model = SVC(gamma='auto', C=3, class\_weight={0:1,1:9})  model.fit(X\_train, y\_train)  y\_pred = model.predict(X\_test)  from sklearn.metrics import accuracy\_score  res = accuracy\_score(y\_test, y\_pred)  print('SVC模型的ACC:\n', res)  from sklearn.metrics import balanced\_accuracy\_score  res = balanced\_accuracy\_score(y\_test, y\_pred)  print('SVC模型的BAC:\n', res)  from sklearn.metrics import f1\_score  res = f1\_score(y\_test, y\_pred)  print('SVC模型的F1:\n', res) |

思考：

1. 将上面的实验结果和《数据挖掘》教材的6.6.2节的实验结果进行比较。能得出什么结论？
2. model = SVC(gamma='auto', C=3, class\_weight={0:1,1:9})语句中

class\_weight={0:1,1:9}是什么含义？

## 5.3 在混合类型数据集上的回归模型

《数据挖掘》教材中没有给出MAE和MSE的评分计算函数。这里给出sklearn提供的评分计算函数。

1. **MAE**

sklearn.metrics.mean\_absolute\_error(y\_true, y\_pred)

主要参数介绍如下：

y\_true：真实目标值

y\_pred：模型预测结果

1. MSE

sklearn.metrics.mean\_squared\_error(y\_true, y\_pred, squared=True)

主要参数同MAE。其中

Squared是布尔值, 默认是True返回MSE，否则返回RMSE。默认是True

两个函数的使用例子如下：

|  |
| --- |
| from sklearn.metrics import mean\_squared\_error  y\_true = [3, -0.5, 2, 7]  y\_pred = [2.5, 0.0, 2, 8]  mean\_squared\_error(y\_true, y\_pred) |

**练习5.5： King County的房屋销售数据集上的回归模型**

美国的King County的房屋销售数据集包括了2014年五月到2015年五月期间的King County的房屋销售数据。它包括三个文件kc\_house\_train.csv是训练集；kc\_house\_test.csv是不包含价格的测试集；kc\_house\_label.csv是对应测试集中每条数据的房屋价格。数据集各个属性的描述见表5-1。

表5-1：King County的房屋销售数据集的属性描述

|  |  |  |
| --- | --- | --- |
| 序号 | 属性名 | 解释 |
| 1 | date | 房屋销售日期 |
| 2 | bedrooms | 卧室数量 |
| 3 | bathrooms | 浴室数量，0.5表示是无浴室厕所 |
| 4 | sqft\_living | 生活空间面积（平方英尺） |
| 5 | sqft\_lot | 室外空间面积（平方英尺） |
| 6 | floors | 房屋层数 |
| 7 | waterfront | 是否能看见海滨（0或1） |
| 8 | view | 房屋的视野（0-4星评分） |
| 9 | condition | 房屋的当前状况（1-5星评分） |
| 10 | grade | 针和condition不同的是，grade对公寓结构、构造、家具等给出的1-13星级评分。例如，1901年建的豪华房屋有12星的grade评分，但condition是1星。 |
| 11 | sqft\_above | 地面上部分的面积 |
| 12 | sqft\_basement | 地下室部分的面积 |
| 13 | yr\_built | 建造的年份 |
| 14 | yr\_renovated | 翻新的年份 |
| 15 | zipcode | 邮政编码 |
| 16 | lat | 维度 |
| 17 | long | 经度 |
| 18 | sqft\_living15 | 最近的15个邻居的平均生活空间面积（平方英尺） |
| 19 | sqft\_lot15 | 最近的15个邻居的平均室外空间面积（平方英尺） |
| 20 | Price | 房屋销售价格 |

数据集的预处理工作包括：

1. date：针对日期数据，抽取其中的年份。一共有2014和2015两个年份，如此将该属性转换成一个0、1值的属性。
2. yr\_built：将该属性转换，建造年份的最大值减去当前建造年份
3. yr\_renovated：将该属性转换，翻新年份的最大值减去当前翻新年份
4. zipcode：邮政编码能反映出房屋的区域位置。邮政编码是6位。当前数据集一共有70个独特值。取前4位作为区域划分，有19个独特值。将他们转换成类别属性。
5. 删除了三个属性：id、lat、long