**scikit\_learn——数据预处理**

**任务目标**

1.了解预处理中各种方法的变换公式

2.熟练掌握sklearn.preprcessing包下各种函数的使用

**相关知识**

在sklearn.preprcessing包下包含很多数据预处理的方法。

规范化：

MinMaxScaler :最大最小值规范化

Normalizer :使每条数据各特征值的和为1

StandardScaler :为使各特征的均值为0，方差为1

编码：

LabelEncoder ：把字符串类型的数据转化为整型

OneHotEncoder ：特征用一个二进制数字来表示

Binarizer :为将数值型特征的二值化

MultiLabelBinarizer：多标签二值化



**任务内容**

练习scikit\_learn中数据预处理的使用方法。

**系统环境**

Python

Jupyter

**任务步骤**

1.首先打开终端模拟器，更新scikit-learn库。

2.然后在终端模拟器的命令行输入ipython notebook

3.新建一个ipynb文件，用于编写并执行代码。

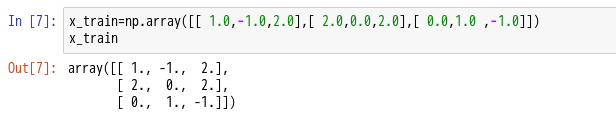
标准化

4.导入sklearn库中的preprocessing模块，导入包numpy，用别名np表示。

1. from sklearn **import** preprocessing
2. **import** numpy as np

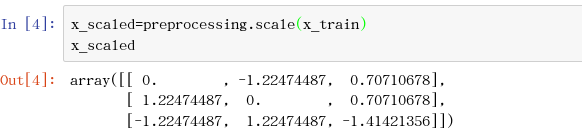
5.使用numpy中的array函数，创建一个3行3列的二维数组x\_train。

1. x\_train=np.array([[ 1.0,-1.0,2.0],[ 2.0,0.0,2.0],[ 0.0,1.0 ,-1.0]])
2. x\_train



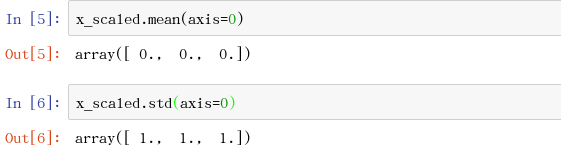
6.使用preprocessing模块中的scale函数，对x\_train数组进行标准化，返回值赋值给x\_scaled。

1. x\_scaled=preprocessing.scale(x\_train)
2. x\_scaled



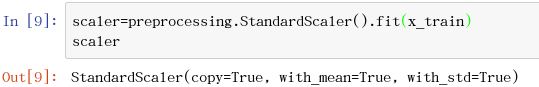
7.x\_scaled具有零均值和单位方差。

1. x\_scaled.mean(axis=0)
2. x\_scaled.std(axis=0)



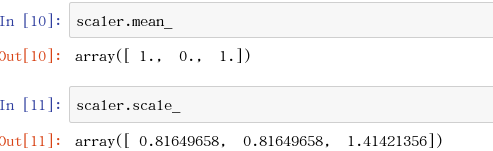
8.第二种方法，使用preprocessing模块中StandardScaler()类的fit()函数，通过数组x\_train构建标准化模型scaler。

1. scaler=preprocessing.StandardScaler().fit(x\_train)
2. scaler



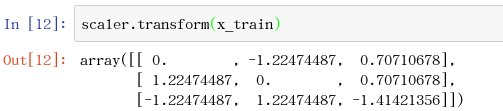
9.查看训练集中的每列特征的平均值和数据的相对缩放比例。

1. scaler.mean\_
2. scaler.scale\_



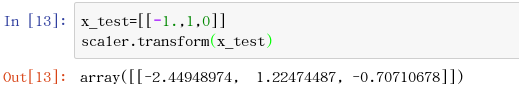
10.调用模型scaler的transform函数对x\_train数组进行标准化。

1. scaler.transform(x\_train)

[](https://www.ipieuvre.com/doc/exper/365f3071-91ad-11e9-beeb-00215ec892f4/img/6a227379-6139-4ee1-ab73-13f698303e3c.png)

11.创建一个测试数据x\_test，使用scaler模型的transform函数对x\_test数据进行标准化。

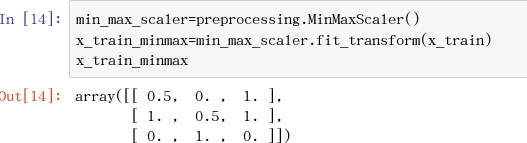
1. x\_test=[[-1.,1,0]]
2. scaler.transform(x\_test)



区间缩放

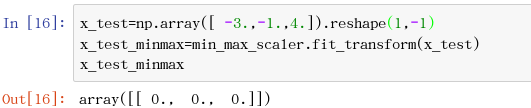
12.使用preprocessing模块中的MinMaxScaler函数，创建一个实例对象min\_max\_scaler，使用实例对象的fit\_transform方法将x\_train中的元素缩放到0-1之间，将缩放的结果返回到x\_train\_minmax中。

1. min\_max\_scaler=preprocessing.MinMaxScaler()
2. x\_train\_minmax=min\_max\_scaler.fit\_transform(x\_train)
3. x\_train\_minmax



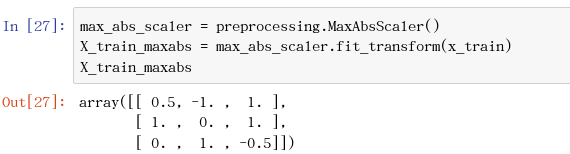
13.使用np.array函数创建元素值为-3，-1,4的一维数组x\_test，然后使用实例对象min\_max\_scaler的fit\_transform方法将x\_test数组中元素进行缩放到[0,1]之间。

1. x\_test=np.array([ -3.,-1.,4.]).reshape(1,-1)
2. x\_test\_minmax=min\_max\_scaler.fit\_transform(x\_test)
3. x\_test\_minmax



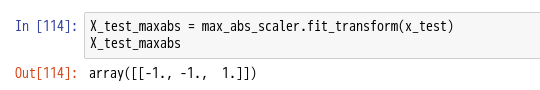
14.使用preprocessing模块中的MaxAbsScaler函数，创建一个实例对象max\_abs\_scaler，使用实例对象的fit\_transform方法将x\_train中的元素缩放到0-1之间，将缩放的结果返回到X\_train\_maxabs中。

1. max\_abs\_scaler = preprocessing.MaxAbsScaler()
2. X\_train\_maxabs = max\_abs\_scaler.fit\_transform(x\_train)
3. X\_train\_maxabs



15.使用实例对象max\_abs\_scaler的fit\_transform方法将x\_test数组中元素进行缩放到[-1,1]之间。

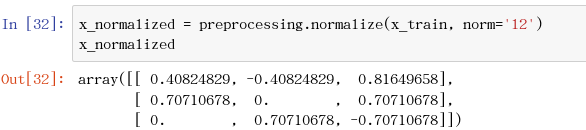
1. X\_test\_maxabs = max\_abs\_scaler.fit\_transform(x\_test)
2. X\_test\_maxabs



正则化

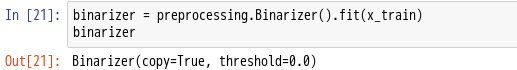
16.使用preprocessing模块中的normalize函数，对x\_train数组进行正则化，返回值赋值给x\_normalized 。

1. x\_normalized = preprocessing.normalize(x\_train, norm='l2')
2. x\_normalized



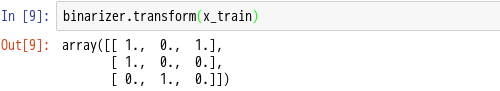
17.使用preprocessing模块中Binarizer中的fit()函数，通过x\_train数组创建二值化模型binarizer，

1. binarizer = preprocessing.Binarizer().fit(x\_train)
2. binarizer



然后使用transform()方法对x\_train数组进行二值化 。

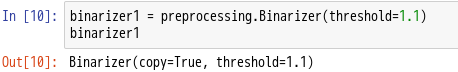
1. binarizer.transform(x\_train)



定量特征二值化sklearn.preprocessing.Binarizer(threshold=0.0, copy=True)根据阈值将数据设置为0或1的特征值。

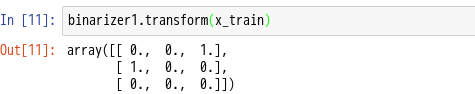
18.调整Binarizer()函数的阀值threshold=1.1，创建二值化模型binarizer1，

1. binarizer1 = preprocessing.Binarizer(threshold=1.1)
2. binarizer1



使用binarizer1模型的transform()方法对x\_train数组进行二值化 。

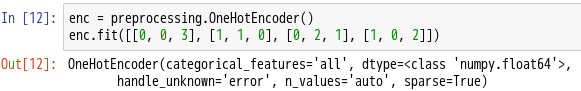
1. binarizer1.transform(x\_train)



定性特征编码

19.使用preprocessing模块中OneHotEncoder()函数，创建一个实例对象enc，使用enc对象的fit方法，通过[[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]]数组创建一个特征编码模型，

1. enc = preprocessing.OneHotEncoder()
2. enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])



使用transform()函数对[[0,1,3]]二维数组进行特性特征编码，并将结果通过toarray函数转换为数组。

1. enc.transform([[0, 1, 3]]).toarray()



20.使用preprocessing模块中OneHotEncoder()函数，创建一个实例对象enc，通过[[1, 0, 3], [0, 2, 0]]数组创建一个特征编码模型码

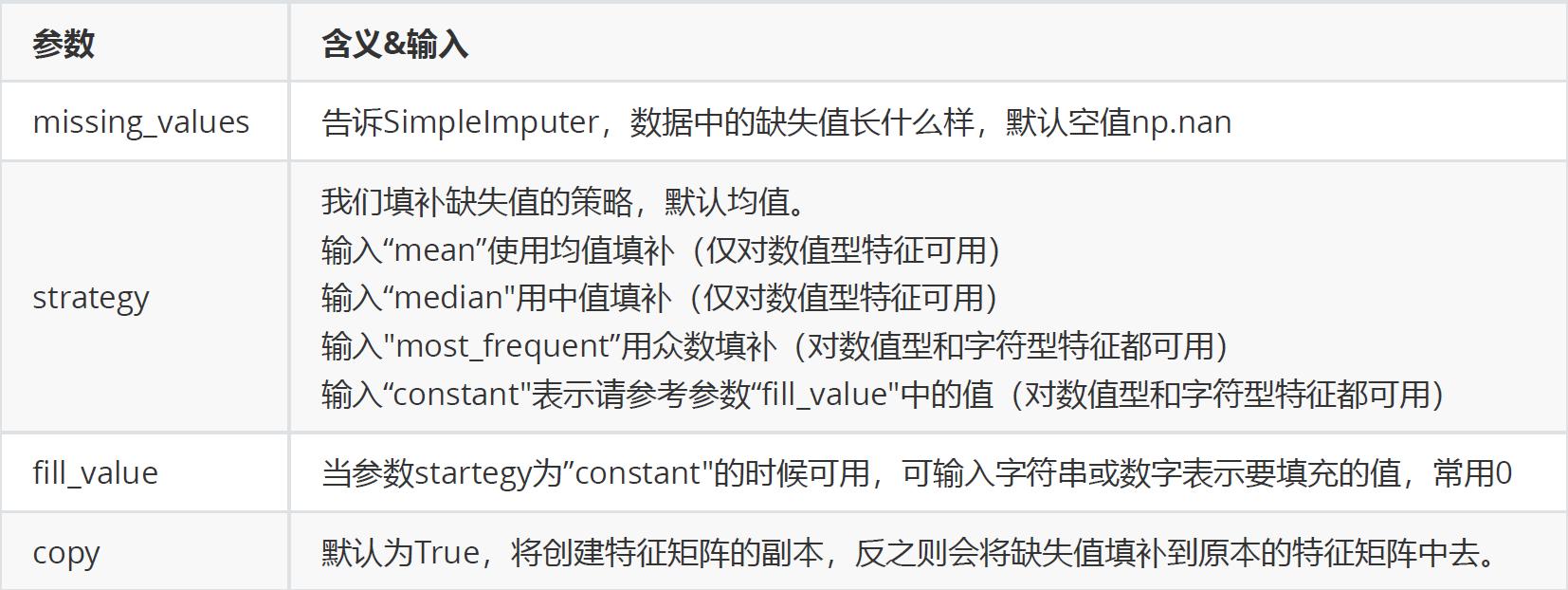
1. enc = preprocessing.OneHotEncoder()
2. enc.fit([[1, 0, 3], [0, 2, 0]])

使用transform()函数对[[1,0,0]]二维数组进行特性特征编码，并将结果通过toarray函数转换为数组。

1. enc.transform([[1, 0, 0]]).toarray()

缺失值计算

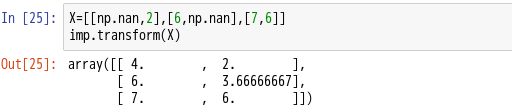
21.使用preprocessing模块中SimpleImputer函数，设置缺失值参数missing\_values=np.nan, 填充缺失值的类型参数strategy='mean'。创建一个实例对象imp，使用实例对象imp的fit方法，通过[[1, 2], [np.nan, 3], [7, 6]]数组创建一个缺失值模型，



1. imp=preprocessing.SimpleImputer(missing\_values=np.nan,strategy='mean')
2. imp.fit([[1,2],[np.nan,3],[7,6]])

创建一个二维列表X，使用imp中的transform()函数，对X的缺失值用上面数组每列的平均值进行填充。

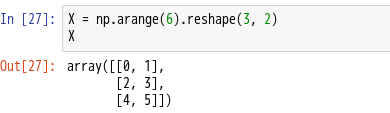
1. X=[[np.nan,2],[6,np.nan],[7,6]]
2. imp.transform(X)



生成多项式特征

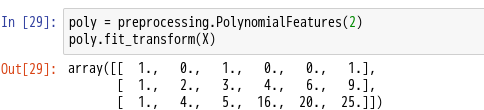
22.使用numpy中的arange函数创建一个值由0到5的整数组成的3行2列数组X，

1. X = np.arange(6).reshape(3, 2)
2. X



使用preprocessing中的PolynomialFeatures函数创建一个多项特征程度为2的多项式对象poly，然后使用poly的fit\_transform方法生成X的多项式特征，X的特征已经从（X1，X2）变成（1，X1，X2，X1\*\*2，X1\*X2，X2\*\*2）。

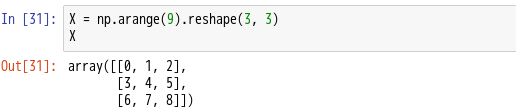
1. poly = preprocessing.PolynomialFeatures(2)
2. poly.fit\_transform(X)



在某些情况下，只需要特征之间的交互项，可以通过设定参数interaction\_only=True。

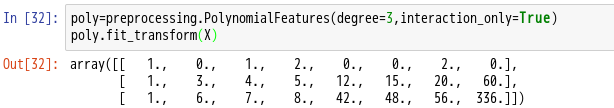
23.使用numpy中的arange函数创建一个值由0到8连续的9个整数组成的3行3列数组X，

1. X = np.arange(9).reshape(3, 3)
2. X



使用preprocessing中的PolynomialFeatures函数创建一个多项特征程度为3的多项式对象poly，设置参数interaction\_only=True，然后使用poly的fit\_transform方法生成X的多项式特征，X的特征已经从（X1，X2，X3）变成（1，X1，X2，X3，X1\*X2，X1\*X3，X2\*X3，X1\*X2\*X3）。

1. poly=preprocessing.PolynomialFeatures(degree=3,interaction\_only=True)
2. poly.fit\_transform(X)

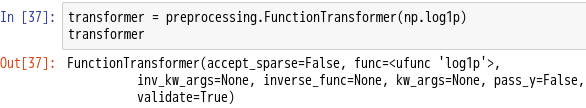


自定义函数变换

24.可以使用FunctionTransformer从任意函数中实现变换，例如，要构建一个在管道中应用log转换的变压器，

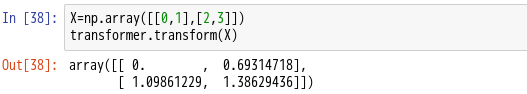
使用preprocessing模块中FunctionTransformer函数，创建一个自定义函数变换实例对象transformer，设置变换函数参数为func=np.loglp，

1. transformer = preprocessing.FunctionTransformer(np.log1p)
2. transformer



使用numpy中的array函数创建一个0到3连续整数组成的2行2列的二维数组X，使用transformer对象的transform方法，将变换后的函数作用于数组X，其中log1p(x) = log(1+x)。

1. X = np.array([[0, 1], [2, 3]])
2. transformer.transform(X)

[](https://www.ipieuvre.com/doc/exper/365f3071-91ad-11e9-beeb-00215ec892f4/img/a1e40344-a858-4c8e-9208-a87b0bb35448.png)