**数据预处理的工具有许多，机器学习中主要有两种：pandas数据预处理和scikit-learn中的sklearn.preprocessing数据预处理，本课程主要学习第二种。**

**首先，sklearn.preprocessing包提供了几个常用的实用函数和转换器类，以将原始特征向量转换为更适合下游估计器的表示。**

**同学们需要自己定义数据（矩阵即可），矩阵大小，形状以及其中的内容均由自己定义，然后，学习sklearn的文档，掌握以下几种常用的数据处理发明方法：**

1. **标准化**

# 使用sklearn.preprocessing.scale()函数，将给定数据进行标准化（样例）

# 使用sklearn.preprocessing.StandardScaler类

# 比如又来了一组新的样本new\_x ，也想得到相同的转换

scaler.transform(new\_x)

# MinMaxScaler

# MinMaxScaler中是给定了一个明确的最大值与最小值

# X\_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

# X\_scaled = X\_std / (max - min) + min

# 同样的，如果有新的测试数据进来

min\_max\_scaler.transform()

函数中的其他功能，大家都可以试一试

还有一些其他的类：MaxAbsScaler、RobustScaler等，大家有时间run一下

**2. 正则化（ Normalizer）**

sklearn.preprocessing.Normalizer(norm=’l2’, copy=True)

# preprocessing还提供了一个实用类Normalizer

# 实用transform方法同样也可以对新的数据进行同样的转换

# 根据训练数据创建一个正则器

对训练数据进行正则, 对新的测试数据进行正则

normalizer.transform(x)

1. **特征二值化**

# 特征的二值化是指将数值型的特征数据转换成布尔类型的值。可以使用实用类Binarizer

# 默认是根据0来二值化，大于0的都标记为1，小于等于0的都标记为0

# 当然也可以自己设置这个阀值，只需传出参数threshold即可

1. **编码分类特征**

LabelEncoder()

OneHotEncoder()

OrdinalEncoder()

1. **缺失值处理**



因此，如果要进行处理的数据集中包含缺失值一般步骤如下：

# 1、使用字符串'nan'来代替数据集中的缺失值；

# 2、将该数据集转换为浮点型便可以得到包含np.nan的数据集；

# 3、使用sklearn.preprocessing.Imputer类来处理使用np.nan对缺失值进行编码过的数据集

使用数组去“训练”一个Imputer类

然后用该类的对象去处理数组x中的缺失值

缺失值的处理方式是使用训练数据中的均值（axis=0表示按列进行）代替x中的缺失值

1. **创建多项式特征**

PolynomialFeatures()

1. **自定义特征的转换函数**

# 通俗的讲，就是把原始的特征放进一个函数中做转换，这个函数出来的值作为新的特征

# 比如说将特征数据做log转换，做倒数转换等等

# FunctionTransformer 可以实现这个功能

**以下选做：**

1. **非线性变换：**

sklearn.preprocessing.QuantileTransformer ()

该方法将特征变换为均匀分布或正态分布

sklearn.preprocessing.PowerTransformer()

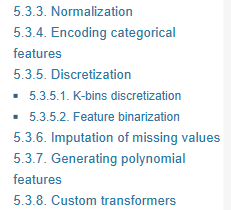
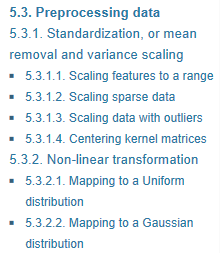
将数据从任意分布映射到尽可能接近高斯分布(稳定方差和最小化偏度)

1. **离散化**

sklearn.preprocessing.KBinsDiscretizer()

**参考：sklearn的文档**

https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing



完成下面这道题：（编程完成并简要叙述得到结果的理由）

**LabelEncoder与OneHotEncoder的混合使用方式**

a = OneHotEncoder()

b = LabelEncoder()

temp是LabelEncoder对[123,456,789]处理的结果

将LabelEncoder的结果作为OneHotEncoder特征输入

输出特征[123,789]的OneHotEncoder的编码结果

要求：最后进行OneHotEncoder编码的时候，temp和OneHotEncoder特征[123,789]均做reshape(-1,1)处理

**from sklearn.preprocessing import LabelEncoder, OneHotEncoder**

**a = OneHotEncoder()**

**b = LabelEncoder()**

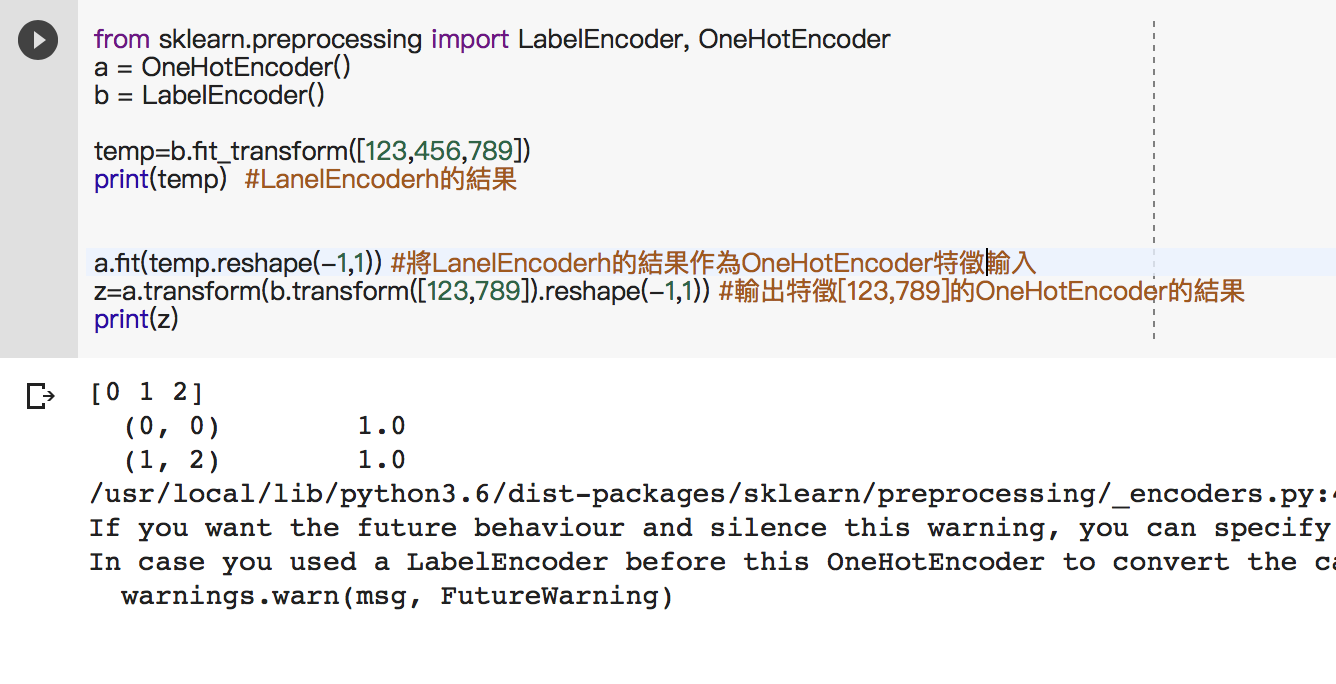
**temp=b.fit\_transform([123,456,789])**

**print(temp) #LanelEncoderh的結果**

**a.fit(temp.reshape(-1,1)) #將LanelEncoderh的結果作為OneHotEncoder特徵輸入**

**z=a.transform(b.transform([123,789]).reshape(-1,1)) #輸出特徵[123,789]的OneHotEncoder的結果**

**print(z)**

****