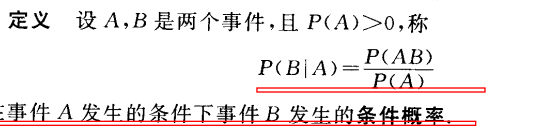
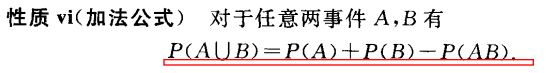


互斥事件：P（A∩B）= 0，P(B│A)= P(A│B)=0， A∩B=Φ，P（A∩B）= 0

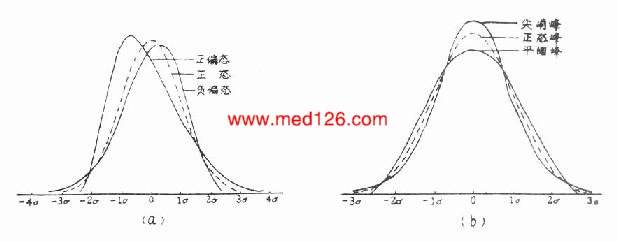
如果A、B**互相独立**，则 P（A∩B）= P（A）P（B）， P(B│A)= P(B)， P(A│B)=P（A）

条件概率：

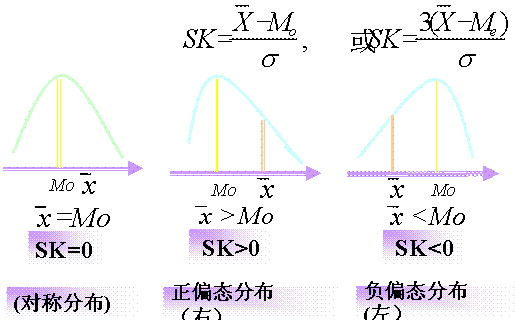




峰度（peakedness;kurtosis）又称峰态系数。表征概率密度分布曲线在平均值处峰值高低的特征数。直观看来，峰度反映了峰部的尖度。正态分布的峰度（系数）为常数3，均匀分布的峰度（系数）为常数1.8。以一般而言，正态分布为参照，峰度可以描述分布形态的陡缓程度，若bk<3，则称分布具有不足的峰度，若bk>3，则称分布具有过度的峰度



偏度系数是描述分布偏离对称性程度的一个特征数。当分布左右对称时，偏度系数为0。当偏度系数大于0时，即重尾在右侧时，该分布为右偏。当偏度系数小于0时，即重尾在左侧时，该分布左偏。



方差是在概率论和统计方差衡量随机变量或一组数据时离散程度的度量。概率论中方差用来度量随机变量和其数学期望（即均值）之间的偏离程度。

https://gss2.bdstatic.com/9fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D113/sign=c388d5738013632711edc632a28ea056/023b5bb5c9ea15cee484a9a6bc003af33a87b233.jpg

标准差能反映一个数据集的离散程度。

æ åå·®ä¸å¹³åå¼å®ä¹å¬å¼

协方差（Covariance）在概率论和统计学中用于衡量两个变量的总体误差。而方差是协方差的一种特殊情况，即当两个变量是相同的情况。

期望值分别为E[X]与E[Y]的两个实随机变量X与Y之间的协方差Cov(X,Y)定义为：

https://gss2.bdstatic.com/9fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D340/sign=6005955212dfa9ecf92e501352d1f754/359b033b5bb5c9ea8718f4add639b6003bf3b391.jpg

从直观上来看，协方差表示的是两个变量总体误差的期望。



公式简单翻译一下是：如果有X,Y两个变量，每个时刻的“X值与其均值之差”乘以“Y值与其均值之差”得到一个乘积，再对这每时刻的乘积求和并求出均值（其实是求“期望”，但就不引申太多新概念了，简单认为就是求均值了）

相关系数

协方差作为描述X和Y相关程度的量，在同一物理量纲之下有一定的作用，但同样的两个量采用不同的量纲使它们的协方差在数值上表现出很大的差异。范围：【-1,1】

公式：

http://h.hiphotos.baidu.com/baike/s%3D153/sign=1337cb6aa1cc7cd9fe2d30dc0a002104/574e9258d109b3de231a34efcebf6c81810a4c94.jpg 就是用X、Y的协方差除以X的标准差和Y的标准差。

数学期望：http://d.hiphotos.baidu.com/baike/s%3D373/sign=cef79ad44f10b912bbc1f0f9f0fcfcb5/5bafa40f4bfbfbeda882eb1070f0f736aec31f81.jpg

|ρXY | = 1时，称X,Y完全相关，此时，X,Y之间具有线性函数关系；

| ρXY | > 0.8时称为高度相关，当 | ρXY | < 0.3时称为低度相关，其它时候为中度相关

0，表示X与Y不相关

-1，完全负相关

机器学习L1和L2正则化（详细解析）

<https://blog.csdn.net/qq_36427732/article/details/81260110>

## 数据与处理processing模块

本篇文章主要简单介绍sklearn中的数据预处理preprocessing模块，它可以对数据进行标准化。preprocessing 模块提供了数据预处理函数和预处理类，预处理类主要是为了方便添加到pipeline 过程中。

以下内容包含了一些个人观点和理解，如有疏漏或错误，欢迎补充和指出。

**数据标准化**

数据标准化：当单个特征的样本取值相差甚大或明显不遵从高斯正态分布时，标准化表现的效果较差。实际操作中，经常忽略特征数据的分布形状，移除每个特征均值，划分离散特征的标准差，从而等级化，进而实现数据中心化。

公式为：(X-X\_mean)/X\_std 计算时对每个属性/每列分别进行.

将数据按其属性(按列进行)减去其均值，然后除以其方差。最后得到的结果是，对每个属性/每列来说所有数据都聚集在0附近，方差值为1。

标准化预处理函数：

* **preprocessing.scale**(X,axis=0, with\_mean=True, with\_std=True, copy=True)：  
  将数据转化为标准正态分布（均值为0，方差为1）
* **preprocessing.minmax\_scale**(X,feature\_range=(0, 1), axis=0, copy=True)：  
  将数据在缩放在固定区间，默认缩放到区间 [0, 1]
* **preprocessing.maxabs\_scale**(X,axis=0, copy=True)：  
  数据的缩放比例为绝对值最大值，并保留正负号，即在区间 [-1.0, 1.0] 内。唯一可用于稀疏数据 scipy.sparse的标准化
* **preprocessing.robust\_scale**(X,axis=0, with\_centering=True, with\_scaling=True,copy=True)：  
  通过 Interquartile Range (IQR) 标准化数据，即四分之一和四分之三分位点之间

它们对应的标准化预处理类：（四种特征缩放的方法）

数据的标准化主要有以下四种方法：

1、实现中心化和正态分布的Z-score

2、实现归一化的Max-Min

3、用于稀疏矩阵的MaxAbs

4、针对离群点的RobustScaler

* **classpreprocessing.StandardScaler**(copy=True, with\_mean=True,with\_std=True)：  
  标准正态分布化的类  
  属性：
  + **scale\_**：ndarray，缩放比例
  + **mean\_**：ndarray，均值
  + **var\_**：ndarray，方差
  + **n\_samples\_seen\_**：int，已处理的样本个数，调用partial\_fit()时会累加，调用fit()会重设
* **classpreprocessing.MinMaxScaler**(feature\_range=(0, 1),copy=True)：  
  将数据在缩放在固定区间的类，默认缩放到区间 [0, 1]，对于方差非常小的属性可以增强其稳定性，维持稀疏矩阵中为0的条目

属性：

* + **min\_**：ndarray，缩放后的最小值偏移量
  + **scale\_**：ndarray，缩放比例
  + **data\_min\_**：ndarray，数据最小值
  + **data\_max\_**：ndarray，数据最大值
  + **data\_range\_**：ndarray，数据最大最小范围的长度
* **classpreprocessing.MaxAbsScaler**(copy=True)：  
  数据的缩放比例为绝对值最大值，并保留正负号，即在区间 [-1.0, 1.0] 内。（特征数据/最大值）可以用于稀疏数据（保留为0的特征）  
  属性：
  + **scale\_**：ndarray，缩放比例
  + **max\_abs\_**：ndarray，绝对值最大值
  + **n\_samples\_seen\_**：int，已处理的样本个数
* **classpreprocessing.RobustScaler**(with\_centering=True,with\_scaling=True, copy=True)：
* 当数据集中含有离群点，即异常值时，可以用z-score进行标准化，但是标准化后的数据并不理想，因为异常点的特征往往在标准化之后容易失去离群特征。此时可以用该方法针对离群点做标准化处理。  
  通过 Interquartile Range (IQR) 标准化数据，即四分之一和四分之三分位点之间  
  属性：
  + **center\_**：ndarray，中心点
  + **scale\_**：ndarray，缩放比例
* **classpreprocessing.KernelCenterer**：  
  生成 kernel 矩阵，用于将 svm kernel 的数据标准化（参考资料不全）

以上几个标准化类的方法：

* + **fit**(X[,y])：根据数据 X 的值，设置标准化缩放的比例
  + **transform**(X[,y, copy])：用之前设置的比例标准化 X
  + **fit\_transform**(X[, y])：根据 X设置标准化缩放比例并标准化
  + **partial\_fit**(X[,y])：累加性的计算缩放比例
  + **inverse\_transform**(X[,copy])：将标准化后的数据转换成原数据比例
  + **get\_params**([deep])：获取参数
  + **set\_params**(\*\*params)：设置参数

**数据归一化**

* **preprocessing.normalize**(X,norm='l2', axis=1, copy=True)：  
  将数据归一化到区间 [0, 1]，norm 可取值 'l1'、'l2'、'max'。可用于稀疏数据 scipy.sparse
* **classpreprocessing.Normalizer**(norm='l2', copy=True)：  
  数据归一化的类。可用于稀疏数据 scipy.sparse  
  方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

**数值数据二值化**

* **preprocessing.binarize**(X,threshold=0.0, copy=True)：  
  将数据转化为 0 和 1，其中小于等于 threshold 为 0，可用于稀疏数据 scipy.sparse
* **classpreprocessing.Binarizer**(threshold=0.0,copy=True)：  
  二值化处理的类，可用于稀疏数据 scipy.sparse  
  方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)，其中fit 函数不会做任何操作

**类别数据编码**

数据的某些特征是文本，特征是无序的，比如国籍，但数字是有序的，所以不能直接用数字编码

* **classpreprocessing.OneHotEncoder**(n\_values='auto',categorical\_features='all', dtype='float', sparse=True,handle\_unknown='error')：  
  将具有多个类别的特征转换为多维二元特征，所有二元特征互斥，当某个二元特征为 1 时，表示取某个类别  
  参数：
  + **n\_values**：处理的类别个数，可以为‘auto’，int或者 int数组
  + **categorical\_features**：被当作类别来处理的特征，可以为“all”或者下标数组指定或者mask数组指定

属性：

* + **active\_features\_**：ndarray，实际处理的类别数
  + **feature\_indices\_**：ndarray，第 i个原特征在转换后的特征中的下标在 feature\_indices\_[i] 和 feature\_indices\_[i+1]之间
  + **n\_values\_**：ndarray，每维的类别数

方法：**fit**(X[, y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

* **classpreprocessing.LabelBinarizer**(neg\_label=0, pos\_label=1,sparse\_output=False)：  
  和 OneHotEncoder 类似，将类别特征转换为多维二元特征，并将每个特征扩展成用一维表示  
  属性：
  + **classes**：ndarry，所有类别的值
  + **y\_type\_**：str
  + **multilabel\_**：bool
  + **sparse\_input\_**：bool
  + **indicator\_matrix\_**：str

方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**inverse\_transform**(y)、**get\_params**([deep])、**set\_params**(\*\*params)

* **preprocessing.label\_binarize**(y,classes, neg\_label=0, pos\_label=1, sparse\_output=False)：  
  LabelBinarizer 类对应的处理函数
* **classpreprocessing.LabelEncoder**：  
  将类别特征标记为 0 到 n\_classes - 1 的数  
  方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**inverse\_transform**(y)、**get\_params**([deep])、**set\_params**(\*\*params)
* **classpreprocessing.MultiLabelBinarizer**(classes=None,sparse\_output=False)：  
  和 LabelBinarizer 类似

feature\_extraction.DictVectorizer类  
patsy包

**数据缺失**

* **classpreprocessing.Imputer**(missing\_values='NaN',strategy='mean', axis=0, verbose=0, copy=True)：  
  参数：
  + **missing\_values**：int 或者“NaN”，对np.nan的值用 "NaN"
  + **strategy**："mean"、"median"、"most\_frequent"

属性：

* + **statistics\_**：ndarray，当axis==0时，取每列填补时用的值

方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

**生成多项式数据**

可以将数据多项式结合生成多维特征，比如 [a,b] 的二次多项式特征为 [1, a, b, a^2, ab, b^2]

* **classpreprocessing.PolynomialFeatures**(degree=2,interaction\_only=False, include\_bias=True)：  
  参数：
  + **degree**：int，多项式次数
  + **interaction\_only**：boolean，是否只产生交叉相乘的特征
  + **include\_bias**：boolean，是否包含偏移列，即全为1 的列

属性：

* + **powers\_**：ndarray，二维数组。powers\_[i,j] 表示第 i 维输出中包含的第 j 维输入的次数
  + **n\_input\_features\_**：int，输入维数
  + **n\_output\_features\_**：int，输出维数

方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

**增加伪特征**

* **preprocessing.add\_dummy\_feature**(X,value=1.0)：  
  在 X 的第一列插入值为 value 的列

**自定义数据转换**

可以使用自定义的 python函数来转换数据

* + **classpreprocessing.FunctionTransformer**(func=None,validate=True, accept\_sparse=False, pass\_y=False)：  
    方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)