各种方法资料：

前言：

注意：org.apache.spark.ml.feature包里的操作，基本都是面向Vector的

即DataFrame里面的column怎么转成Vector， 又怎么将Vector转成其他Vector，怎么将Vector解析，以及Vector的其他操作。

该包的3个部分，均是对特征进行操作：提取、转换、选择

提取：主要是面向“文本”的特征抽取——————将文本转成Double形式的数据

转换：很多是对“向量”的操作、转换

选择：

DCT(Discrete Cosine Transform，离散余弦变换)：

应用领域：图像处理，压缩算法

<https://blog.csdn.net/newchenxf/article/details/51719597/>

Features:

# Feature Transformers

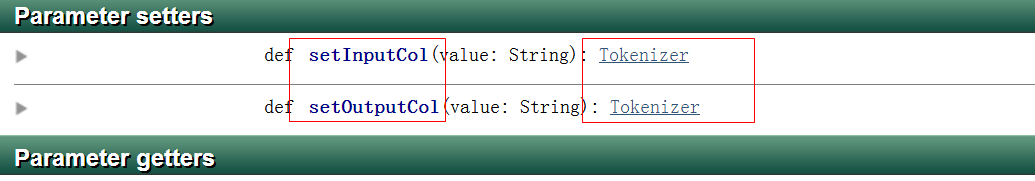
## Tokenizer

分词器：——将单词或字符串分割

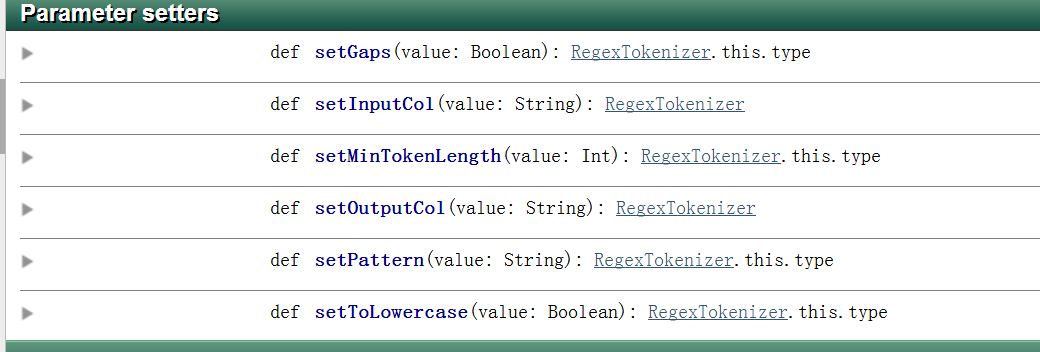
有两种：

Tokenizer——————常规分词器（应该是以空格分割）没有核心参数，只有setInputCol和setOutputCol

RegexTokenizer————正则匹配分词器（按正则表达式切割），核心参数:setPattern(RegexString)



正则分割器，除了分割外，还有可选项，如是否转成小写

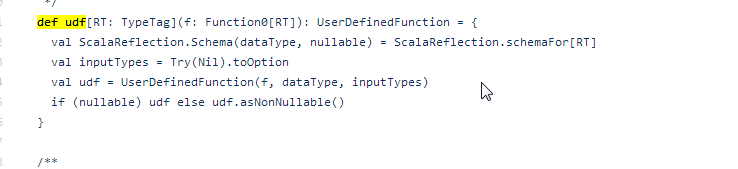


另外，这里还学到了在DataFrame下的udf，很简单，就是先import org.apache.spark.sql.functions.\_模块， 然后直接调用udf。 此处udf是一个函数：

源码如下， 这是其中一种udf实现方式，返回UserDefinedFunction,

而UserDefinedFunction类是一个case class，由functions里的udf函数产生。截图如下：

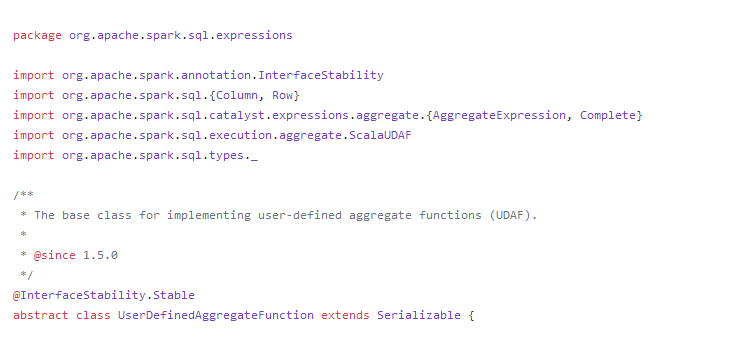




发现：其实udf函数做的事情， 就是将一个f:Function和DataType及InputTypes这3个参数传给了UserDefinedFunction这个case class。

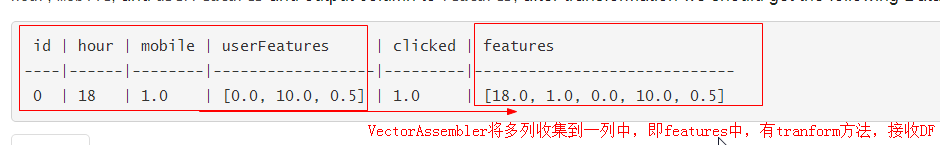
但目前不知道如何使用。

但是UDAF则不同，是一个抽象类：UserDefinedAggregateFunction

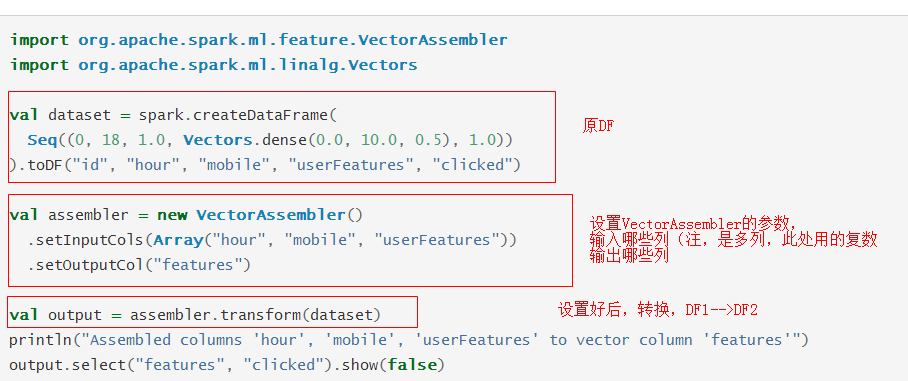


VectorAssembler

作用:将DataFrame中的多列特征变量，放到一列里，用于机器学习



语法：



注意:DataFrame里面什么类型都可以装的，也即Row里面任意类型都可以装

但是如何解析出来呢？可能要转换？

可以转换，就是将Any通过asInstanceOf转换，强制转换成对应的类

如下代码：

val row = Row(Vectors.dense(Array(1.0,2.0,3.0)))

val vec =row.get(0)

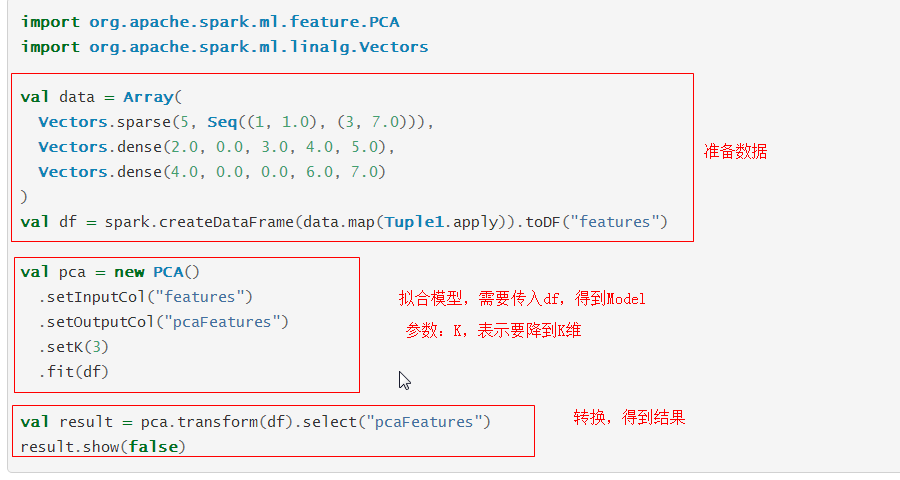
vec.asInstanceOf[DenseVector]

其他类型都可以如此从Row中解析出来。

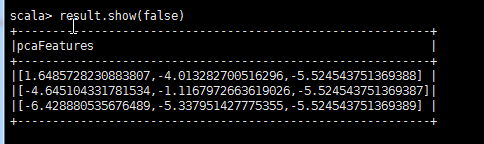
DF.show(false) 可以显示所有信息，而不是部分显示，不会截断或只显示前20行

PCA：

主成分分析，降维的一种方式——————常用于特征提取，语法简单：



结果形如：



注意：PCA算子拟合的DF的特征是5列，也即DF的字段名是5列，通过PCA的setK(3)降维到3列

setK中的K表示主成分的个数（也即降维后的维数）

PCA的原理见：

<https://baike.baidu.com/item/%E4%B8%BB%E6%88%90%E5%88%86%E5%88%86%E6%9E%90/829840?fr=aladdin>

## 多项式扩展——PolynomialExpansion

作用和原理：?

见：

## Discrete Cosine Transform (DCT)

离散余弦转换：

## VectorIndexer

## 向量索引器:

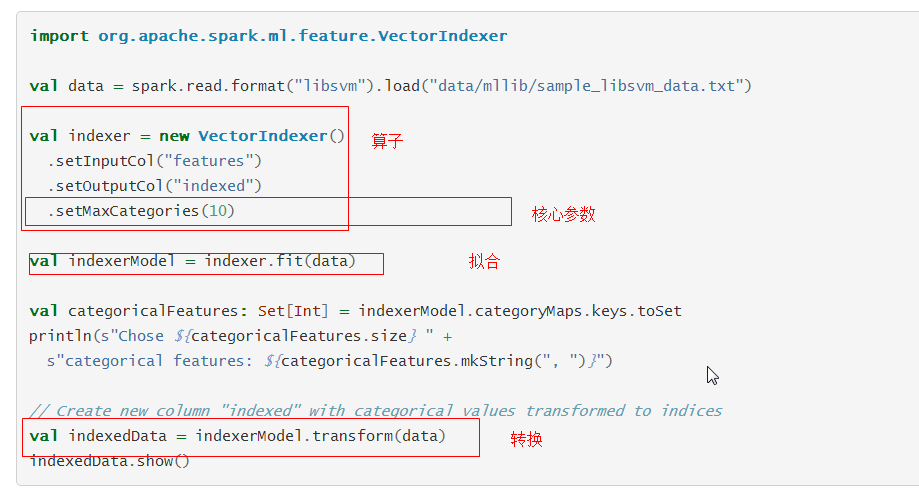
用途：用于推断哪些变量是分类特征、哪些是连续特征

接收核心参数：setMaxCategories(10)——此处表示，离散特征最多有10种不同值，某特征变量的不同值，超过10个则判定为“连续型特征”，<=10个则为“离散特征”

经过fit和transform算子后，会输出IndexedFeatures，将判定为“离散特征”的变量，标记并转换为“分类特征”————也即有特殊处理，不同于连续型的。

我们知道，one-hot编码就是将类似于1,2,3,4…这种表示分类的离散特征转码，因为用1,2,3,4虽然能够表示分类， 但做回归或机器学习时，其值的大小，会影响计算结果，因此使用分类一次热编码处理。

代码：



## Interaction

向量的交叉相乘， 如两个向量[a1,a2,a3]和[b1,b2,b3]，则转换为 二者作笛卡尔交叉相乘：

得到————[ a1\*b1,a1\*b2,a1\*b3, a2\*b1,a2\*b2,a2\*b3, a3\*b1,a3\*b2,a3\*b3]

举例：

有个DF

v1 | v2

[a1,a2,a3] | [b1,b2,b3]

这两个向量均作为输入，输出则如下：

v1 | v2 | transformedVector

[a1,a2,a3] | [b1,b2,b3] | [ a1\*b1,a1\*b2,a1\*b3, a2\*b1,a2\*b2,a2\*b3, a3\*b1,a3\*b2,a3\*b3]

代码：

这里assembled2与assembled1确实不同，但是做interaction的不是这2者，看参数，是对vec1和vec2做交叉乘积

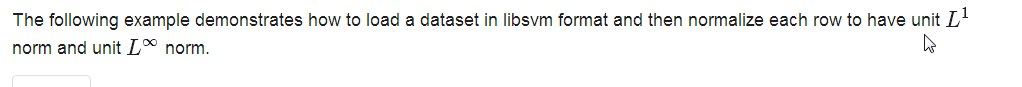


## Normalizerz

正则化----------将不同单位的数据，处理后同等看待，即去量纲

参数：setP-----------------P-Norm,即P范数

有两种正则方式：L1和L∞



对于向量 [a1,a2,a3,…,ai,…]

则L1的正则化方式为：--------------转换后元素= ai/(Σ |ai|)

即生成如下向量：

假设向量范数为： norm = |a1|+|a2|+|a3|+…+|ai|+… ,此处|ai| 表示ai的绝对值，取norm为分母

转换后的向量：

[a1/norm,a2/norm,a3/norm,…,ai/norm,…]

而L∞的正则化方式为：取元素ai中 绝对值最大的 Max(|ai|)，取其为分母，

默认归一化的范数为2，即将数据映射到【-1，1】区间

归一化为无量纲处理

## StandardScaler

标准化：——————将数据全部映射到1个小的空间，和归一化有差别

即Z-score处理， z变换 ： z = (ξ-mean)/stddev

两个参数（值为true or false）：



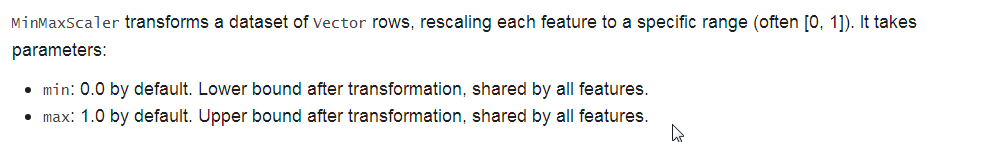
## MinMaxScaler

离差标准化：

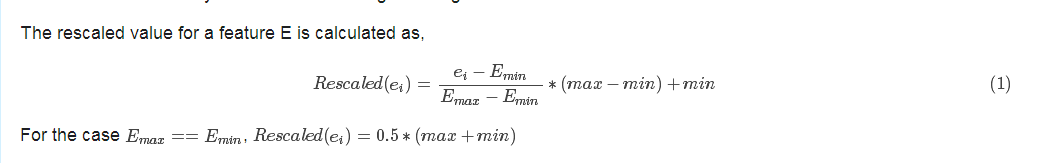
2个核心参数：

setMin:左区间 ，默认为0.0

setMax：右区间， 默认为1.0



公式，很简单：



也是fit、transform

## MaxAbsScaler

最大绝对值标准化：

没有核心参数：

对于向量vec = [a1,a2,a3,…,ai,…]

最大绝对值 MaxAbs = Max(|ai|)

转换后的vec1

= [a1,a2,a3,…,ai,…]/MaxAbs = [a1/MaxAbs,a2/MaxAbs,a3/MaxAbs,…,ai/MaxAbs,…]

## Bucketizer

桶分区：  
接收参数：splits ————其中splits是Array[Double]类型，若其长度为n，则会形成n-1个桶，这些桶其实就是区间：

例如：将（-∞，+∞）分成多个桶（即区间）

若splits为 Array( Double. NegativeInfinity, a1, a2, a3, PositiveInfinity)

则会形成如下4个桶：

【Double. NegativeInfinity, a1】，【a1, a2】，【a2, a3】，【a3,PositiveInfinity】

且桶分别标记为：

0 ，1, 2 ，3

当某个column的某元素，落入【Double. NegativeInfinity, a1】时，会被标记为0；落入【a1, a2】时，被标记为1， 依次类推。 落入哪个桶，则标记为对应的索引。

标记后，其转换算子，该元素所在行对应的outPutCol处就是 该标记值。

就是“桶分区”算子

注：2.3里面inputCols可以接受Array， outPutCol也可以接受Array，即输入N列，则也输出N列的转换值（即桶算子的结果）

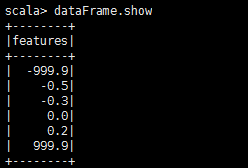
截图：

计算前：

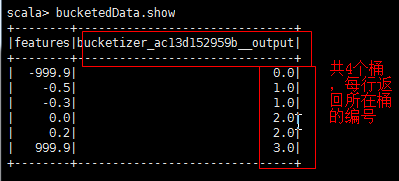
splits:

Array(Double.NegativeInfinity,-0.5,0.0,0.5,Double.PositiveInfinity)

DF:



计算后:



## ElementwiseProduct——回归（或神经网络）中的赋权操作：

元素级别乘法算子：

即向量与向量的元素级别的相乘， 并得到新的向量：

如果向量1 作为向量2、向量3、…、向量I 做 该算子， 那向量1，被称为“权”

即 VectorWeight Ops VectorWeighted

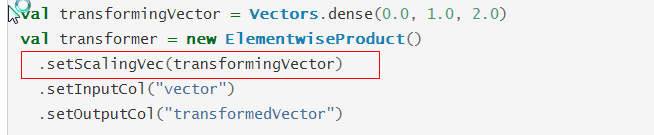
此处： 向量1对应VectorWeight， 向量2、向量3、…、向量I均属于VectorWeighted（被赋权） ，Ops对应算子操作（元素级别乘法算子）

试看回归： y = XT\*β + α

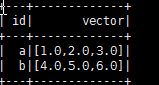
其实XT\*β 就是做的 该算子 操作。 其中β就是“权” ，而XT 矩阵中“行向量” 则对应VectorWeighted。

参数：setScalingVec() ,里面传参为 向量

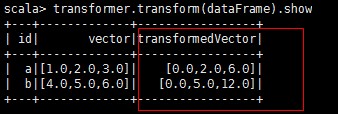




转换前：



转换后：

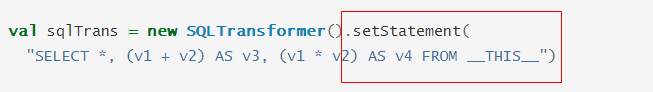


## SQLTransformer

sql转换算子：

参数：

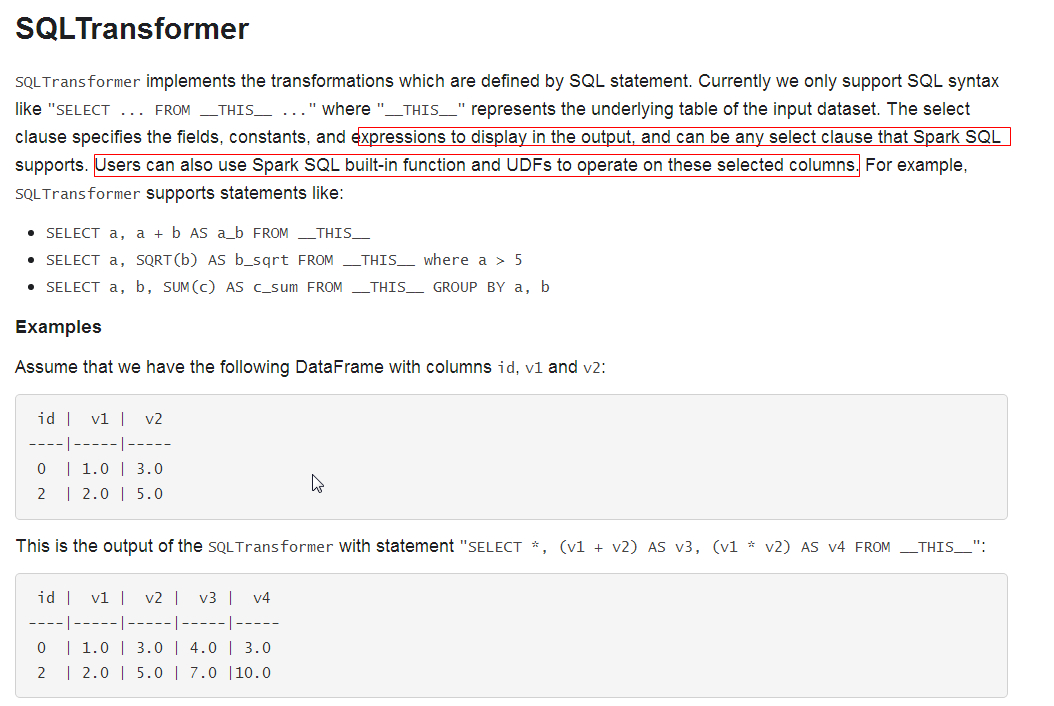
setStatement(sql) ,其中 sql对应的是sql语句

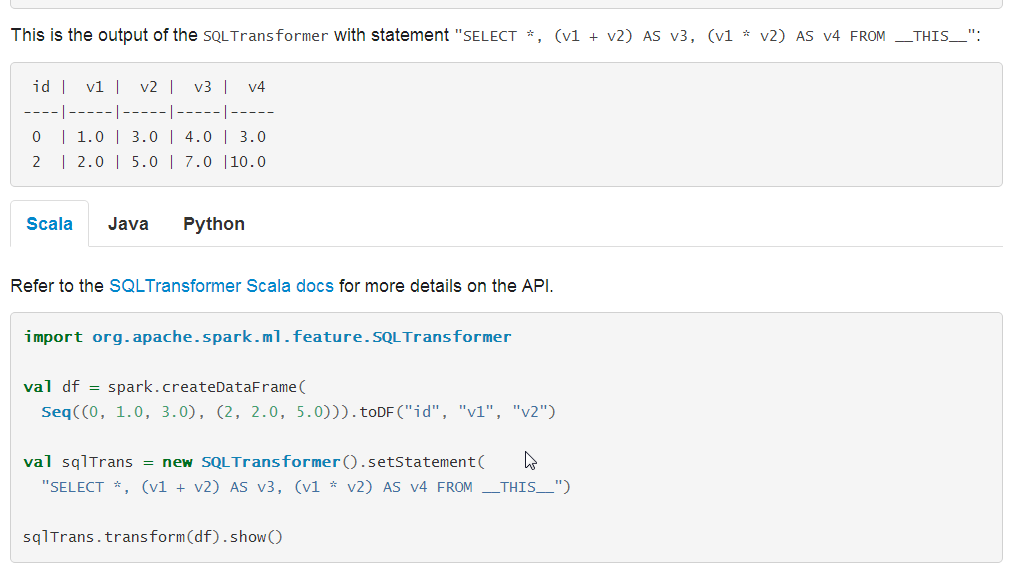


注意:\_\_This\_\_代表后面transform(df)里的表。 表示对该表作的sql操作

目前只支持：标准的SQL操作。（这个SQL转换算子，应该不支持hive）

全文如下：





## VectorSizeHint

向量长度绑定算子：

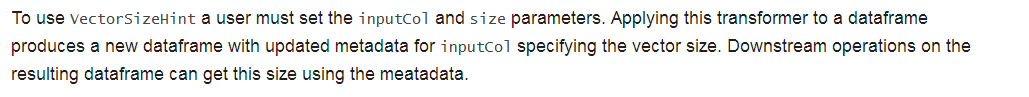
用途：某个column中，其元素若为Vector，该算子可用于“预先定义”向量的长度

VectorAssembler本身自带向量长度的信息，但是对于streaming data传过来的DF，有时向量长度是无法预知的。

因此需要该算子.

参数：

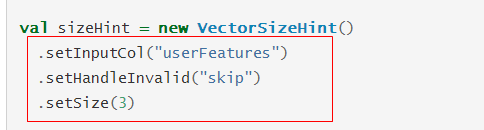
1.inputCol 和size



2. handleInvalid:--------------当column中有“无效参数”时，采取的策略（忽视、保留、报错）

该参数有3个枚举值：“error”,”skip”,”optimistic”,默认是“error”

参数示例如下：



注：貌似2.2.0没有该算子，2.3后才支持

## QuantileDiscretizer（按分位数，划分成N个桶，标记元素落入哪个桶）

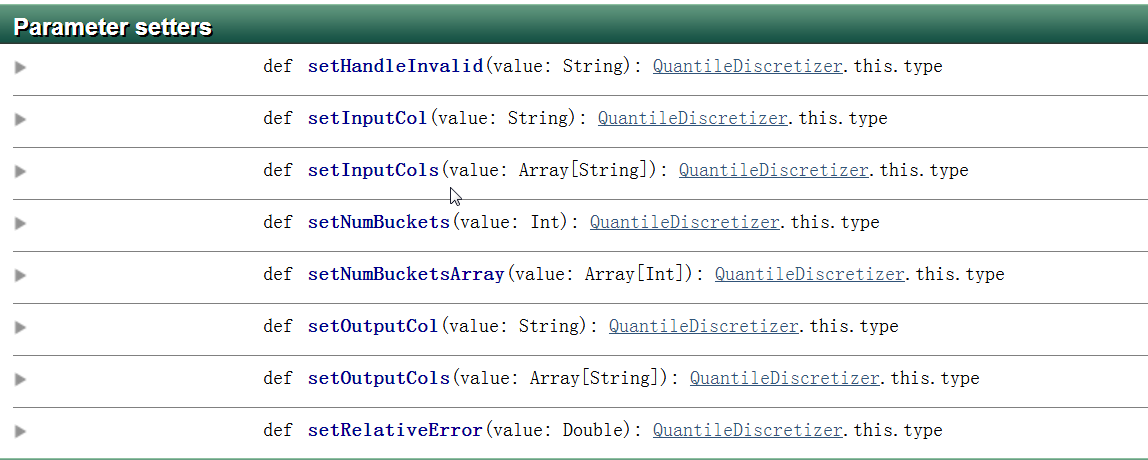
分位离散算子：

用途：针对”连续型数据” ，本质上和进行Bucketizer很类似， 区别在于: Bucketizer是自己指定splits，即桶的上下区间； 而QuantileDiscretizer则是根据用户设置有多少个桶，自行计算“分位”区间。

注意：N个桶（分位桶），就是计算出 分位数（ 1/N分位数，表示[-∞,a（1/N）， a(1/N)表示排序后，前第1/N个数对应的坐标值）————分位数产生的桶，一般不是等间距的， 每个桶里的元素个数却一定相同。

参数：numBuckets 整数

对于NaN值， 会将其单独放入一个桶，即NaN桶



## Imputer——修复数据

估算算子

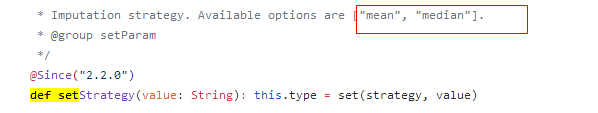
用途： 填充“缺失值”——用（求平均、中位数替代）估算并填充数据

注意：目前不支持“分类”特征变量

参数：

1.setMissingValue(custom\_value)——————参数为，自定义的“缺失值”标记，任意类型

2. setStrategy————枚举值，2种： “mean” 或“median”



# Feature Selectors——特征选择

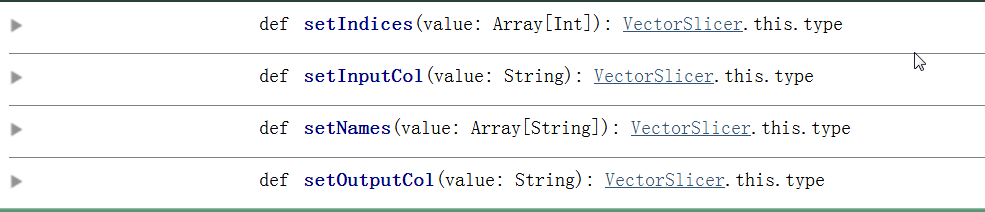
## VectorSlicer——向量分片，抽取向量的（一部分）

即DF中组合的features列是Vector形式时，转换成新的列selectedFeatures,其包含的也是Vector，不过其是原Vector的子集了。

参数：

setIndices: 按整数索引，获取部分向量

setNames: 按向量的“字段名s”获取向量中的某些元素



注意：setIndices和setNames可以一起用，但是不能出现重复的特征列，否则会报错

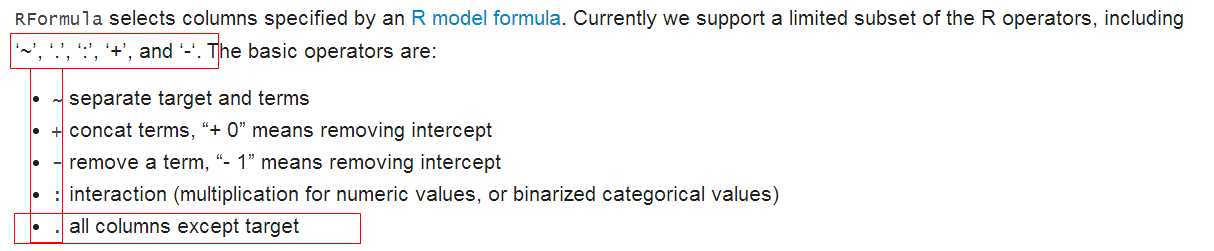
## RFormula

R形式算子：

注意：仍然是用于特征选取

作用：以更简洁的R的模型表达方式， 选取哪些列作为特征列，哪个列作为“标签列”

基本的类R运算符号：



这些符号，要在.setFormula("clicked ~ country + hour")参数里使用：

其中~ 就是y和X之间的分隔符

+ 就是表示变量们相“加” ，+0表示去除截距项

* 就是表示去除某个变量 ，-1 表示去除截距项

： 就是变量“相乘” ——对数值变量、分类变量均可

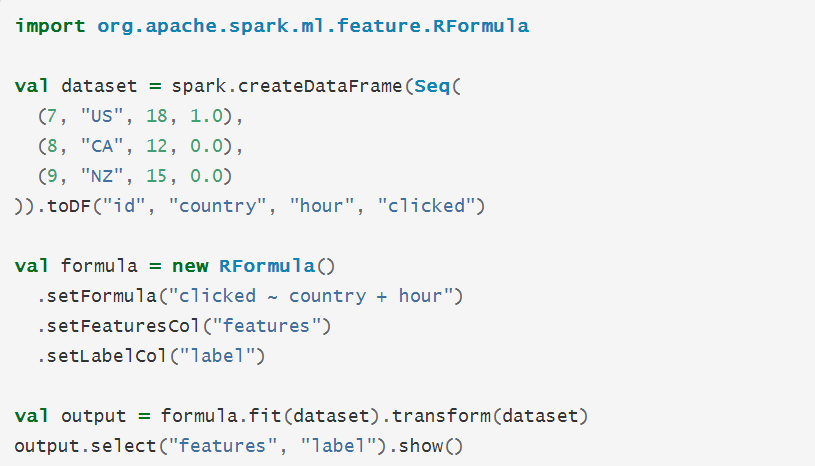
. 就是除了y的所有列， 即只选取X

注意：

对于“String类型”的“特征变量”， 会进行one-hot编码转换

对于“String类型”的“标签变量”，会进行StringIndexer转换

代码如下：



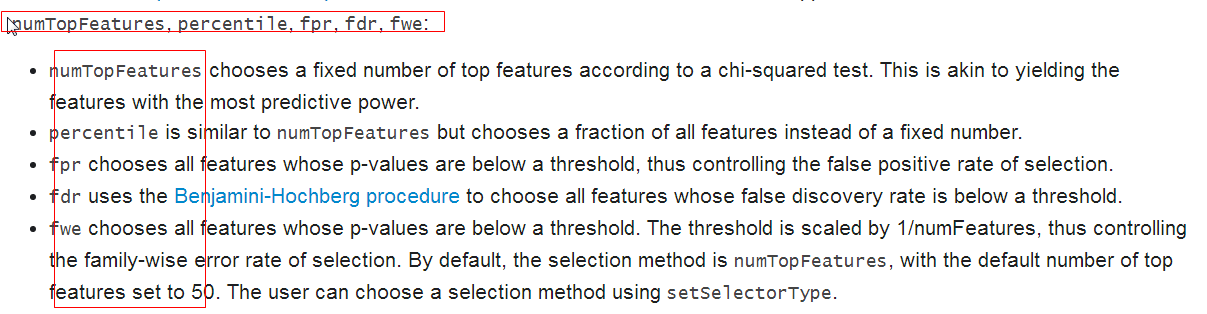
## ChiSqSelector

卡方选择算子：

用途： 一种“特征筛选”的算子

原理：利用“卡方独立性检验”，获取最具解释性的变量

方法有5种：



numTopFeatures: 选取前N个变量

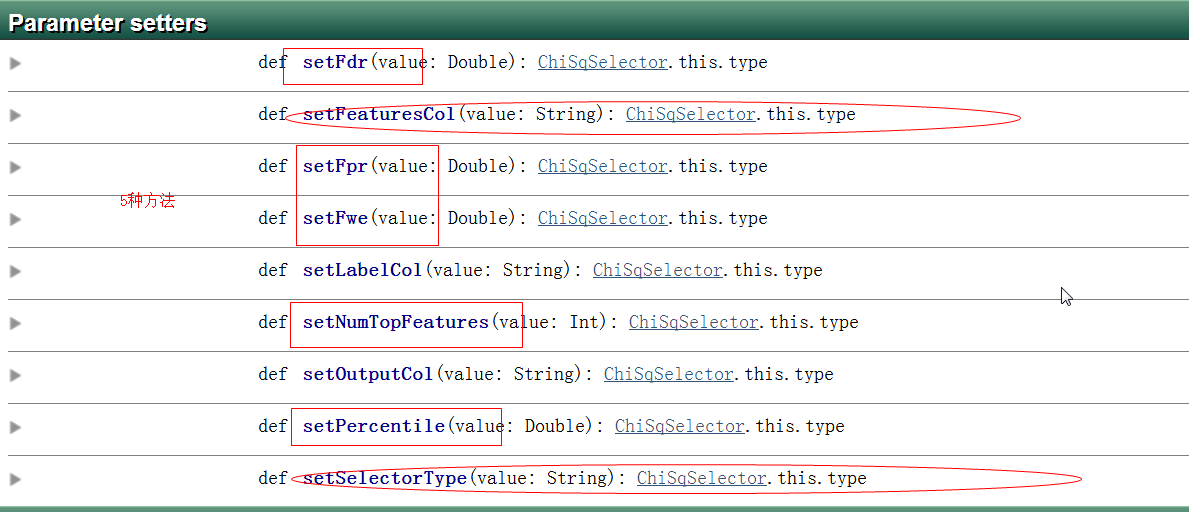
percentile: 选取前N%分位的变量

fpr： 设置p-values的阀值， 控制“假正率”

fdr: 控制“错误发现率”

fwe： 设置p-values的阀值， 该阀值会被1/NumFeatures 标准化，控制famliy-wise error rate

参数：

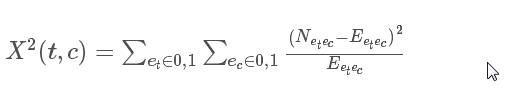


关于“卡方检验”用于“特征筛选”的原理：

<https://blog.csdn.net/shuzfan/article/details/52993427>

思想：若X和Y变量是相互独立的，则互不相关，X影响Y的概率很小，那么X就不是一个好的特征变量；反之，则不独立，说明X和Y相关性很强，需要选择之。

计算卡方值的公式：



也即右侧随机变量， 符合卡方分布，卡方分布对应一个p值

评估：卡方值越大，p值越小，说明互相独立的概率越小（与我平常的理解不一致，但右侧公式比较复杂，应该是这个逻辑），则越相关。

简言之： 卡方值越大，p值越小，越应该选取该特征变量。

LSH

<https://blog.csdn.net/guoziqing506/article/details/53019049>