特征预处理

## 算子简介

### 功能简介

特征预处理算子的目的是为spark的数据框提供一个特征提取、特征转换、特征选择等（以下统一称特征预处理）的接口，方便后面spark.ml或spark.mllib调用。

该算子主要封装spark.ml.feature中的特征预处理的API（目前截止到spark 2.3）为一个算子，并添加一些字段属性判断、缺失值过滤等功能。

### 设计思想

1. **由于特征预处理是一个高度灵活的过程，因此本算子的设计思想是，将特征提取的算法拆解成通用的几个环节，以达到分层复用的目的。**

**因此用户需要忍受对数据输入格式的硬性要求来获得处理过程的灵活性。**

示例：

比如tf-idf转换，一种直接的算子处理过程可能是：输入词汇（ArrayType[StringType]类型）得到tf-idf向量（Vector）。

而本算子是将其拆解为两个过程：

1. TF（词频统计），输入词汇记录（ArrayType[StringType]）类型进行词频统计，获得词频向量（Vector类型）；
2. IDF（逆文档频率），输入Vector形式的词频统计向量，进行逆文档频率的映射（Vector类型）。

在此过程中，我们拆分为TF和IDF两个阶段，导致需要分两个阶段完成，增加了参数输入和数据输入格式的要求，但是同时获得了灵活性，我们可以将两个阶段和其他特征预处理方式进行合理的组合，比如：

1. 一般的词频统计+idf
2. Hash词频统计+idf
3. 数据中已有Vector形式的词频统计列+idf
4. 数据中有词频统计的若干列+向量集成+idf
5. 词频统计+特征选择+idf
6. 其他合理的操作
7. 如何使用特征预处理算子组合进行特征预处理
8. 可以在分析流中进行组合，可参考demo。
9. 通过何种形式能够将多列转换为Vector类型和ArrayType？

由于DataFrame中最长见的类型是原生类型的String、Double等类型，而spark.ml进行机器学习的常用类型是Vector和ArrayType，

1）原生类型之间的转换，比如转为Double或String，可以通过“数据建模”算子进行转换。

2）因此我们有时候需要将多个原生类型的列转为Vector或ArrayType。可以通过以下方式进行转换：

A.向量集成

B.数组集成——**暂时还没有该需求，同时源码里也没有，如果需要的话可以写个算法加入到本算子的“特征选择中”**

C.向量数组互转——**暂时还没有该需求，“数据建模”算子目前也做不到这一点，如果需要的话可以写个算法加入到本算子的“特征选择中”**

### 算法归类

由于特征预处理的方法较多，算子根据处理数据的类型和功能对算法进行了分类，将其划分为五类：

1. 属性类特征变换

即属性类的特征的转换，这里包括：正则表达式分词、词频统计、hash词频统计、文本向量化、停用词移除和n-gram。主要是属性类的特征（最常见的如自然语言处理中的词汇）进行的转换：如分词、统计频次等特征转换。

1. 数值类特征变换

即数值类的特征的转换。这里数值类型可以是Double类型以及Vector类型。这里的变换包括：离散化、独热编码、逆文档频率转换、低变异性数值特征索引化、提取主成分、多项式分解、离散余弦变换

1. 数量尺度变换

也是数值类型的特征的转换，不过只是更加简单的尺度变换。变换包括：正则化、标准化和加权

1. 字符串和数值索引互转

包括属性类的特征转为数值索引，以及数值索引转为对应字符串

1. 特征选择

多个特征之间的简单拆分或组合，称为特征选择，区别于特征转换，特征选择往往不涉及数值的变化。本算子的特征组合变换包括：选取子向量、卡方特征选择、向量集成。

## 具体算法和参数简介

本算子的具体参数简介如下：

全局参数简介

输入预处理表名

请选择预处理种类

### 属性类特征变换

#### 正则表达式分词

预处理列：此时需要类型为String类型

输出列：输出一个ArrayType(StringType,true)类型的列

匹配模式：请输出合适的正则表达式模式。如果正则表达式不正确，限于spark的lazy模式，目前只能捕捉非执行过程的异常。

是以此为分隔还是以此为匹配类型：如果选择以此为分隔，则算子会根据正则表达式找到对应的字符串并以此为分隔符对整个字符串进行分割，否则则会遍历整个字符串找到对应的子串。

是否转为小写：会将预处理列中的字符串中的英文转为小写

#### 词频统计

根据运行的逻辑可以将算法拆分为两个阶段：训练和预测，训练是指训练一个分词模型（主要是根据设定参数训练一个带顺序的词汇表），预测是根据分词模型将输入数据进行频次统计后转为Vector类型。

**参数说明：**

预处理列：ArrayType(StringType,true)类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1. 从持久化引擎中获取模型：

输入模型路径

输入预测表输入列名

1. 训练新模型

训练表名：训练表的名称

训练列名：ArrayType(StringType,true)类型的列

最大词汇数，系统会根据所有词汇的词频取取最常出现的n个，这里n就是最大词汇数，默认值是2^18。可以根据数据设置合适的大小，建议不要太大，该算法中需要将所有的词汇表collect到driver端，过大的词汇数会导致总词汇的内存超过akka.frameSize的上界。

最小文档频率，当某词汇出现在文档中的频率低于该频率时会被从词汇表中清除掉。

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

最小词频：某条记录的词汇中低于此词频数的词汇会被过滤掉。

预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

Hash词频统计

该算法对每个词汇采用，hash并将hash值通过取模的方式映射到指定索引（从0到（长度-1））。不同于词频统计，此处没有将每个词汇collect获得词汇表再统计词频，而是建立一种由词汇到索引映射关系再获得词频向量，因而避免了获取词汇表时的结点通信。缺点是，不同词汇的hash值可能相等，取模时不同的hash可能相等，庆幸的是当我们将参数调整到合理的值时，这种概率是很小的。

**参数说明：**

输入列名：ArrayType(StringType,true)类型的列

词汇数：默认值是2^18，需要小于2^31

输出列名：Vector

#### 文档向量化

Word2vec是现在自然语言处理中常用语义分析的一种技术，它采用一系列代表文档的词语来训练word2vecmodel。该模型将每个词语映射到一个固定大小的向量。word2vecmodel使用文档中每个词语的平均数来将文档转换为向量，然后这个向量可以作为预测的特征，来计算文档相似度计算等等。常见的Word2Vec模型包括：Skip-gram、CBOW等，目前spark API实现的是Skip-gram。

Skip-gram本质上是一个自学习的神经网络（自编码器），与一般神经网络不同的是这里的隐含层没有激活函数，而只是在输出层使用了激活函数。因此在输出层之前的隐含层实现的就是一个由（词汇数\*1）到（隐藏层\*1）的一个线性映射，这里隐藏层等于向量长度，也就是乘以一个维度为（词汇数\*向量长度）的矩阵。

**参数说明：**

预处理列：ArrayType(StringType,true)类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1. 从持久化引擎中获取模型：

输入模型路径

输入预测表输入列名：ArrayType(StringType,true)类型的列

1. 训练新模型

训练表名：训练表的名称

训练列名：ArrayType(StringType,true)类型的列

最大词汇数：系统会根据所有词汇的词频取取最常出现的n个，这里n就是最大词汇数，默认值是2^18。可以根据数据设置合适的大小，建议不要太大，该算法中需要将所有的词汇表collect到driver端，过大的词汇数会导致总词汇的内存超过akka.frameSize的上界。

最小文档频率，当某词汇出现在文档中的频率低于该频率时会被从词汇表中清除掉。

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

最小词频：某条记录的词汇中低于此词频数的词汇会被过滤掉，这里说的是所有记录中的词频，通过最小词频可以去掉一些频数比较小的词频，降噪，同时还可以降低词汇表的数目，减少集群间数据传输的压力（可参见本文算法部分“参数使用注意事项3”）。

向量长度默认值为100

学习率默认值为0.025

最低词频默认值为5

NumPartition并行度默认值为1，即不并行。

这里并行度等于设定的分区数，对每个分区的词汇分别进行训练得到对应系数最后在进行整合。因此在集群还有空闲运算资源的情况下，并行度越大运算越快，但模型越不精确。

在集群能够运行的情况下可以设小一些以求得到最大精度。

循环次数：将模型训练多少次，每次训练都会使用上次迭代结果作为初值。默认值为1，即循环一次即终止。

预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

**参数使用注意事项：**

基于以上原理我们在模型是需要注意：

1. 训练数据的数目需要大于模型数——词汇数目\*向量长度
2. 词汇数目\*向量长度最大不能超过2^31 / 8 = 2^28
3. 为了akka.frameSize超界（spark的API会将所有词汇的频次统计collect到driver端）等问题同时也为了增加模型的自由度，最好确保总词汇数目不要过大。可以先将词汇进行标准化，比如现将词汇去掉停用词、词汇去掉数字等，以降低词汇数目，您还可以通过调大最低词频来降低总的词汇数目。

#### 停用词移除

在自然语言中，很多语句中的词汇在进行训练时对我们用处不大，比如在进行语义分析时一些停用词如“具体来说，以下分为三个方面：1）时间发生了变化；2）地方发生了变化”。在一定假设条件下“具体来说”可以作为停用词，因为我们可以认为它只是语气意义上的延续。

**参数说明：**

输入数据：要处理的表名。

输入预处理列名：ArrayType(StringType,true)类型的列

请选择停用词模式：可以选择“汉语”、“英语”、“手动输入”，“从hdfs集群中读取”四种模式。其中“汉语”停用词收录了互联网上整理的哈工大停用词词库1872个停用词，“英语”停用词收录了“Glasgow Information Retrieval Group”贡献的318个停用词。“手动输入”是以动态列表的形式输入，从“hdfs集群中读取”，需要提供路径和分隔符。

是否忽略大小写：如果选择是，会不计大小写过滤掉停用词。

输入输出列名：输出一个ArrayType(StringType,true)类型的列

#### n-gram

即自然语言处理中常用的ngram方法，本质就是通过一个定长的平滑窗口，对一个记录依次进行取值，从而获得一个考虑到前后文的词语特征。例如一个[“我”，“听说”，“小明”，“中了”，“彩票”]的文档，如果采用窗口为2的n-gram方法进行特征转换，会得到[“我听说”，“听说小明”，“小明中了”，“中了彩票”]。窗口数越大获得的上下文信息越多，不过不同的词汇数也因此增加，需要结合后面用到的机器学习算法设定恰当的gram窗口长度。

**参数说明：**

输入数据：要处理的表名。

输入预处理列名：ArrayType(StringType,true)类型的列

请输入gram窗宽：输入一个大于1的整数。

输入输出列名：输出一个ArrayType(StringType,true)类型的列

### 数值类型特征提取

#### 数值数据离散化

将数值类型的数据通过分箱形式离散化，返回所在箱子的索引。

**参数说明：**

输入数据：要处理的表名。

输入预处理列名：ArrayType(StringType,true)类型的列

请选择分箱模式：等宽分箱、等深分箱、自定义边界分箱

1. 等宽分箱

周期初始相位——某个箱子的起始数值：输入合理的初始值。可以是任意箱子的起始值。

箱子宽度：输入正数。

1. 等深分箱

以深度方式设定分箱、以箱子数方式设定分箱

以深度方式设定分箱：

箱子深度：大于0的整数，同时小于数据总数

以箱子数设定分箱

箱子数：大于0的整数

1. 自定义边界分箱

请输入分箱边界：是一个动态列表的形式，需要数据数值类型的数据。

是否以负无穷作为分箱边界的最小值

是否忽略大小写：如果选择是，会不计大小写过滤掉停用词。

输入输出列名：输出一个ArrayType(StringType,true)类型的列

**注意事项：**

1. 等深分箱中设定分箱数目，分箱边界的具体算法如下：
2. 选取用于数据用于获得分箱边界，样本小于10000时，选取全部数据，样本量大于10000时，如果分箱数（如果等深分箱，箱子数等于数据除以深度的向上取整）的平方大于10000，随机抽取分箱数的平方，否则10000条。
3. 其次选取分箱边界

**源码的分箱算法有个小bug。**

此处选取分箱边界的依据不是按箱子数，而是依据近似计算的每个箱子深度。

步长 = 选取数据条数除以分箱数的向上取整，箱子边界从最小值开始以此选取，选取恰当的值作为边界直到数据数目超过了当前箱子的数目，依次选取直至选取完所有边界。因此会造成，**当数据量和分箱数差不多时，实际分箱数不等于设置的分箱值**:

箱子有5个值，选择分成4个箱子，此时选择的步长为5.0/4.0的向上取整，为2.0，因此分箱试图为每个箱子装2.0个元素，会造成分箱后，箱子的最大id只可能是2.0，也就是箱子数为3个箱子。

1. 最后根据分箱边界进行分箱
2. 等深分箱箱子分箱的边界依据抽样进行的，当样本较大时，随机抽取一定数目（箱子数目的平方和10000的最大值）的数据。

#### 独热编码

独热编码是将

**参数说明：**

输入数据：要处理的表名。

输入预处理列名：Double类型的列，里面每个值需要为索引值，即该值取整和该值相等，如：1.0,2.0等

是否去掉最后一个索引：如果去掉，向量长度变为最大索引数（否则为最大索引数+1——因为从0开始）。改设定主要是保证在独热编码后进行机器学习算法加入常数项后不会共线性（否则每列向量只有一个1.0的值，向量的所有列值之和为一个值全部为1.0的向量，此时和常数项一致，因此导致了共线性）。

输出： Vector类型的列，每列为一个向量，但该向量只有一个值不为0，为1.0，这也是one-hot称谓的由来。

#### IDF转换

将词频统计后的数据进行逆文档频率变换，改变换和词频统计合称词频-逆文档频率(TF-IDF)。

词频－逆向文件频率（TF-IDF）是一种在文本挖掘中广泛使用的特征向量化方法，它采用了频率乘以特异性的模式，可以兼顾一个文档中词语在语料库中的稀有程度和词频两方面的特征。

特异性主要用逆文档频率来衡量，如果以D代表整个语料库，以t代表词，以d代表文档，则tf-idf变换如下：



总的tf-idf公式就是：



这里就是IDF转换部分。

**参数说明：**

预处理列：Vector类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1. 从持久化引擎中获取模型：
2. 训练新模型

训练表名：训练表的名称

训练列名：Vector类型的列等长向量

最小出现文档数：需要输入大于等于0的整数，如果某个词汇出现的文档数小于该数值就会被过滤掉。

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

1. 输出列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

#### 低变异性数值特征索引化

这里的名称是本算子开发人员自己取的。spark源码中的名称叫做VectorIndexer，目前没有一个较好的翻译，互联网上的叫法有“向量索引”、“部分特征到类别索引的转换”，都没有很好的体现算法的特性。

该算法用于将实际分布为离散分布的数值特征转为属性类型。一般来说数值类型的数据分布是连续的，而属性类型的数据是离散的。VectorIndexer是对数据集特征向量中的类别（离散值）特征（index categorical features categorical features ）添加索引。

这里区分连续和离散的方式是设定一个阈值，如果某个特征在数据中出现的所有不同的频数小于该阈值，则认为是离散特征，由此对其中每条记录添加索引。

基于上述分析，本算子给VectorIndexer翻译的名称叫做“离散型数值特征索引”。

**参数说明：**

预处理列：Vector类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1） 从持久化引擎中获取模型：

输入模型路径

2） 训练新模型

训练表名：训练表的名称

训练列名：Vector类型的列等长向量

离散型特征频次阈值：某特征的所有不同值低于此阈值时认为是离散型特征

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

3)预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

#### 主成分分析

主成分分析是一种降维方法。通过正交变换将一组可能存在相关性的变量经过线性边换为一组线性不相关的变量，转换后的这组变量叫主成分，通常主成分小于特征数，因此称作降维。而这一线性变换的选取原则是依据特征之间的相关关系获得最能代表目前特征信息的线性映射。

这里特征预处理过程中的主成分分析和平台上的主成分分析算子的唯一不同之处在于，它是基于Vector类型的。

**参数说明：**

预处理列：Vector类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1） 从持久化引擎中获取模型：

输入模型路径

2） 训练新模型

训练表名：训练表的名称

训练列名：Vector类型的列等长向量

主成分数：需要小于等于特征数，且是一个正整数。

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

3)预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

#### 多项式展开

将特征映射到高维的一种变换。即将特征以高次交叉项的形式映射到新的特征列。

**参数说明：**

预处理列：Vector类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

主成分数：多项式展开的幂大于0的正整数

预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

#### 离散余弦变换

是将时域的N维实数序列转换成频域的N维实数序列的过程（有点类似离散傅里叶变换）。

离散余弦变换是与傅里叶变换相关的一种变换，它类似于离散傅立叶变换但是只使用实数。离散余弦变换相当于一个长度大概是它两倍的离散傅里叶变换，这个离散傅里叶变换是对一个实偶函数进行的（因为一个实偶函数的傅里叶变换仍然是一个实偶函数）。离散余弦变换，经常被信号处理和图像处理使用，用于对信号和图像（包括静止图像和运动图像）进行有损数据压缩。

### 数量尺度变换

也是数值类型的特征的转换，不过只是更加简单的尺度变换。变换包括：正则化、标准化和加权。

#### 正则化

正则化是将每个向量除以同一个尺度，如果将每个向量看做一行，那正则化是基于“行”进行尺度变化的，除以的尺度为该向量的p阶范式，其中p为正则化的阶数，如果某个向量的p阶范式为0则不进行任何变化。

#### 标准化

标准化是认为数据中的向量等长，同时每个向量的同一个位置代表同一个特征，标准化是对向量同一特征进行的，也就是按“列”进行尺度变化的。这里的尺度变化包括：同时减去一个数、同时乘以或除以一个数。

分为z-score标准化和min-max标准化

1，z-score标准化

即传统的均值-方差归一化。可以选择是否归一化均值和是否归一化方差，两者至少选择一个。

2.min-max标准化

将数据通过min-max的方式映射到给定区间，给定区间是由最大值和最小值组成的，其中最大值需要大于最小值。

#### 向量加权

即认为向量的每个位置代表一个特征（数据需要是等长向量），依次输入权重，会得到加权后的向量值。

### 特征选择

多个特征之间的简单拆分或组合，称为特征选择，区别于特征转换，特征选择往往不涉及数值的变化。本算子的特征组合变换包括：子向量选取、卡方特征选择、向量集成。

#### 子向量选取

即认为向量的每个位置代表一个特征（数据需要是等长向量），依次输入要选取的子向量的id，从原特征中选取对应特征。

#### 卡方特征选择

即认为向量的每个位置代表一个特征（数据需要是等长向量），通过卡方值评价每种特征对标签的识别程度，从而选取最能识别的若干特征。该算法需要输入标签列，是监督学习的特征选择过程。

#### 向量集成

将Double、Int、Long、Float等Numeric类型以及Vector类型集合为一个大的Vector。

## 常见异常处理

由于特征预处理对数据的要求较强，同时spark的处理过程是lazy模式，单个算子的编译过程更多依据的是数据的传递过程而非数据本身，因此有些异常可能需要到有action类型操作的算子才会触发（比如视图算子）。文档的该部分主要介绍如果处理这方面的异常：

1. 独热编码

**java.lang.AssertionError: assertion failed: Values from column hour must be indices, but got 2.2**

出现该异常的原因是数据中有非整数的值。独热编码要求所有的数据必须是1.0,2.0类型的，不能为非整数值。