task four

MLlib是Spark的机器学习（Machine Learning）库，旨在简化机器学习的工程实践工作，并方便扩展到更大规模。MLlib由一些通用的学习算法和工具组成，包括分类、回归、聚类、协同过滤、降维等，同时还包括底层的优化原语和高层的管道API

具体包括

- 算法工具：常用的学习算法，如分类、回归、聚类和协同过滤
- 特征化工具：特征提取、转化、降维和工具选择
- 管道(pipeline)：用于构建、评估和调整机器学习管道的工具
- 持久性：保存和加载算法、模型和管道
- 实用工具：线性代数、统计、数据处理等工具

Spark 机器学习库从 1.2 版本以后被分为两个包：

- spark.mllib 包含基于RDD的原始算法API。Spark MLlib 历史比较长，在1.0 以前的版本即已经包含了，提供的算法实现都是基于原始的 RDD。
- spark.ml 则提供了基于DataFrames 高层次的API，可以用来构建机器学习工作流（PipeLine）。ML Pipeline 弥补了原始 MLlib 库的不足，向用户提供了一个基于 DataFrame 的机器学习工作流式 API 套件。

使用 ML Pipeline API可以很方便的把数据处理，特征转换，正则化，以及多个机器学习算法联合起来，构建一个单一完整的机器学习流水线。这种方式给我们提供了更灵活的方法，更符合机器学习过程的特点，也更容易从其他语言迁移。Spark官方推荐使用spark.ml。如果新的算法能够适用于机器学习管道的概念，就应该将其放到spark.ml包中，如：特征提取器和转换器。开发者需要注意的是，从Spark2.0开始，基于RDD的API进入维护模式（即不增加任何新的特性），并预期于3.0版本的时候被移除出MLLib。因此，我们将以ml包为主进行介绍

目前MLlib支持的主要的机器学习算法：

	离散数据	**连续数据**
监督学习	Classification、LogisticRegression(with Elastic-Net)、SVM、DecisionTree、RandomForest、GBT、NaiveBayes、MultilayerPerceptron、OneVsRest	Regression、LinearRegression(with Elastic-Net)、DecisionTree、RandomFores、GBT、AFTSurvivalRegression、IsotonicRegression
无监督学习	Clustering、KMeans、GaussianMixture、LDA、PowerIterationClustering、BisectingKMeans	Dimensionality Reduction, matrix factorization、PCA、SVD、ALS、WLS

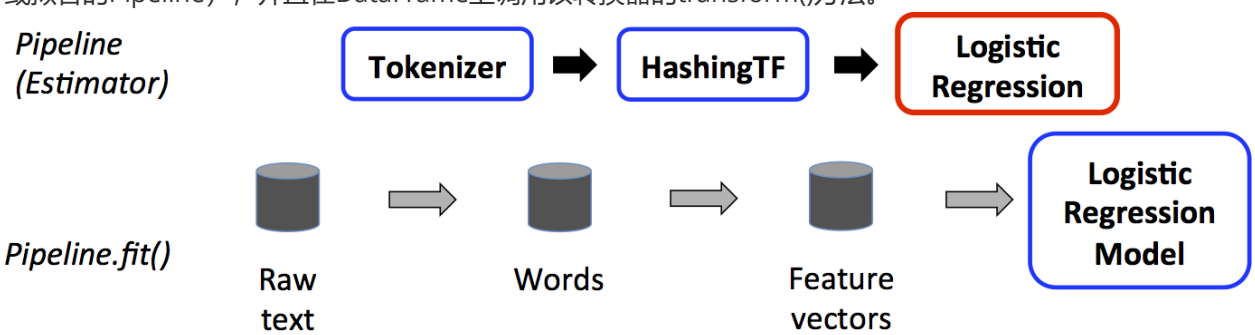
机器学习工作流(ML pipelines)

一个典型的机器学习过程从数据收集开始，要经历多个步骤，才能得到需要的输出。这非常类似于流水线式工作，即通常会包含源数据ETL（抽取、转化、加载），数据预处理，指标提取，模型训练与交叉验证，新数据预测等步骤。

- DataFrame：使用Spark SQL中的DataFrame作为数据集，它可以容纳各种数据类型。 较之 RDD，包含了 schema 信息，更类似传统数据库中的二维表格。它被 ML Pipeline 用来存储源数据。例如，DataFrame中的列可以是存储的文本，特征向量，真实标签和预测的标签等。
- Transformer：翻译成转换器，是一种可以将一个DataFrame转换为另一个DataFrame的算法。比如一个模型就是一个 Transformer。它可以把 一个不包含预测标签的测试数据集 DataFrame 打上标签，转化成另一个包含预测标签的 DataFrame。技术上，Transformer实现了一个方法transform（），它通过附加一个或多个列将一个DataFrame转换为另一个DataFrame。
- Estimator：翻译成估计器或评估器，它是学习算法或在训练数据上的训练方法的概念抽象。在 Pipeline 里通常是被用来操作 DataFrame 数据并生产一个 Transformer。从技术上讲，Estimator实现了一个方法 fit（），它接受一个DataFrame并产生一个转换器。如一个随机森林算法就是一个 Estimator，它可以调用 fit（），通过训练特征数据而得到一个随机森林模型。
- Parameter：Parameter 被用来设置 Transformer 或者 Estimator 的参数。现在，所有转换器和估计器可共享用于指定参数的公共API。ParamMap是一组（参数，值）对。
- PipeLine：翻译为工作流或者管道。工作流将多个工作流阶段（转换器和估计器）连接在一起，形成机器学习的工作流，并获得结果输出。

工作流如何工作:

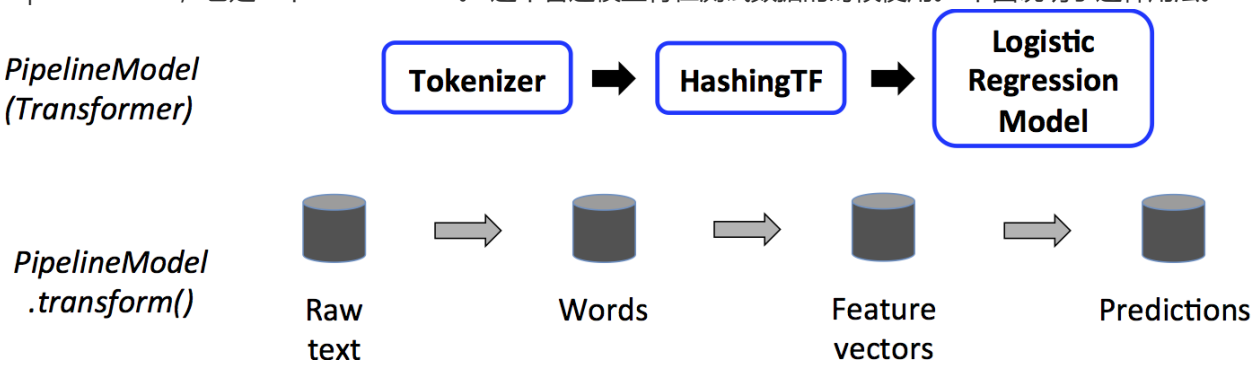
1. 需要定义 Pipeline 中的各个工作流阶段PipelineStage，（包括转换器和评估器），比如指标提取和转换模型训练等。有了这些处理特定问题的转换器和 评估器，就可以按照具体的处理逻辑有序的组织PipelineStages并创建一个Pipeline。比如 `pipeline =Pipeline(stages=[stage1,stage2,stage3])`
2. 把训练数据集作为输入参数，调用 Pipeline 实例的 fit 方法来开始以流的方式来处理源训练数据。这个调用会返回一个 PipelineModel 类实例，进而被用来预测测试数据的标签。更具体的说，工作流的各个阶段按顺序运行，输入的DataFrame在它通过每个阶段时被转换。 对于Transformer阶段，在DataFrame上调用transform（）方法。 对于估计器阶段，调用fit（）方法来生成一个转换器（它成为PipelineModel的一部分或拟合的Pipeline），并且在DataFrame上调用该转换器的transform()方法。



顶行表示具有三个阶段的流水线。 前两个（Tokenizer和HashingTF）是Transformers（蓝色），第三个（LogisticRegression）是Estimator（红色）。 底行表示流经管线的数据，其中圆柱表示DataFrames。

1. 在原始DataFrame上调用Pipeline.fit（）方法，它具有原始文本文档和标签。
2. Tokenizer.transform（）方法将原始文本文档拆分为单词，向DataFrame添加一个带有单词的新列。
3. HashingTF.transform（）方法将字列转换为特征向量，向这些向量添加一个新列到DataFrame。
4. 现在，由于LogisticRegression是一个Estimator，Pipeline首先调用LogisticRegression.fit（）产生一个LogisticRegressionModel。
5. 如果流水线有更多的阶段，则在将DataFrame传递到下一个阶段之前，将在DataFrame上调用LogisticRegressionModel的transform（）方法。

值得注意的是，工作流本身也可以看做是一个估计器。在工作流的fit（）方法运行之后，它产生一个PipelineModel，它是一个Transformer。 这个管道模型将在测试数据的时候使用。 下图说明了这种用法。



在上图中，PipelineModel具有与原始流水线相同的级数，但是原始流水线中的所有估计器都变为变换器。

当在测试数据集上调用PipelineModel的transform（）方法时，数据按顺序通过拟合的工作流。 每个阶段的transform（）方法更新数据集并将其传递到下一个阶段。工作流和工作流模型有助于确保培训和测试数据通过相同的特征处理步骤。

特征相关处理以及对应算法举例

特征抽取：从原始数据中抽取特征

TF-IDF(HashingTF and 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，它可以体现一个文档中词语在语料库中的重要程度。**大致含义为一个词在更少的文档中出现更多的次数则词更重要。**
TF-IDF

词语由t表示，文档由d表示，语料库由D表示。词频TF(t,d)是词语t在文档d中出现的次数。文件频率DF(t,D)是包含词语的文档的个数。如果我们只使用词频来衡量重要性，很容易过度强调在文档中经常出现，却没有太多实际信息的词语，比如“a”，“the”以及“of”。如果一个词语经常出现在语料库中，意味着它并不能很好的对文档进行区分。IDF就是在数值化文档信息，衡量词语能提供多少信息以区分文档。其定义如下：

$$IDF(t,D) = log\frac{|D|+1}{DF(t,D)+1}$$

此处D是语料库中总的文档数。公式中使用log函数，当词出现在所有文档中时，它的IDF值变为0，加1是为了避免分母为0的情况，TF-IDF度量值表示如下：

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

在Spark ML库中，TF-IDF被分成两部分：TF (+hashing) 和 IDF。

TF: HashingTF 是一个Transformer，在文本处理中，接收词条的集合然后把这些集合转化成固定长度的特征向量。这个算法在哈希的同时会统计各个词条的词频。

IDF: IDF是一个Estimator，在一个数据集上应用它的fit（）方法，产生一个IDFModel。 该IDFModel 接收特征向量（由HashingTF产生），然后计算每一个词在文档中出现的频次。IDF会减少那些在语料库中出现频率较高的词的权重。

Spark.mllib 中实现词频率统计使用特征hash的方式，原始特征通过hash函数，映射到一个索引值。后面只需要统计这些索引值的频率，就可以知道对应词的频率。这种方式避免设计一个全局1对1的词到索引的映射，这个映射在映射大量语料库时需要花费更长的时间。但需要注意，通过hash的方式可能会映射到同一个值的情况，即不同的原始特征通过Hash映射后是同一个值。为了降低这种情况出现的概率，我们只能对特征向量升维。i.e., 提高hash表的桶数，默认特征维度是 2^20 = 1,048,576.

特征转换：特征的维度、特征的转化、特征的修改

在机器学习处理过程中，为了方便相关算法的实现，经常需要把标签数据（一般是字符串）转化成整数索引，或是在计算结束后将整数索引还原为相应的标签。

Spark ML包中提供了几个相关的转换器，例如：StringIndexer、IndexToString、OneHotEncoder、VectorIndexer，它们提供了十分方便的特征转换功能，这些转换器类都位于org.apache.spark.ml.feature包下。

用于特征转换的转换器和其他的机器学习算法一样，也属于ML Pipeline模型的一部分，可以用来构成机器学习流水线，以StringIndexer为例，其存储着进行标签数值化过程的相关 超参数，是一个Estimator，对其调用fit(..)方法即可生成相应的模型StringIndexerModel类，很显然，它存储了用于DataFrame进行相关处理的 参数，是一个Transformer（其他转换器也是同一原理） 下面对几个常用的转换器依次进行介绍。

StringIndexer

StringIndexer转换器可以把一列类别型的特征（或标签）进行编码，使其数值化，索引的范围从0开始，该过程可以使得相应的特征索引化，使得某些无法接受类别型特征的算法可以使用，并提高诸如决策树等机器学习算法的效率。

索引构建的顺序为标签的频率，优先编码频率较大的标签，所以出现频率最高的标签为0号。 如果输入的是数值型的，我们会把它转化成字符型，然后再对其进行编码。

StringIndexer

IndexToString

与StringIndexer相对应，IndexToString的作用是把标签索引的一列重新映射回原有的字符型标签。

其主要使用场景一般都是和StringIndexer配合，先用StringIndexer将标签转化成标签索引，进行模型训练，然后在预测标签的时候再把标签索引转化成原有的字符标签。当然，你也可以另外定义其他的标签。

OneHotEncoder

把一列类别性特征（或称名词性特征，nominal/categorical features）映射成一系列的二元连续特征的过程，原有的类别性特征有几种可能取值，这一特征就会被映射成几个二元连续特征，每一个特征代表一种取值，若该样本表现出该特征，则取1，否则取0。

One-Hot编码适合一些期望类别特征为连续特征的算法，比如说逻辑斯蒂回归等。

VectorIndexer

tringIndexer是针对单个类别型特征进行转换，倘若所有特征都已经被组织在一个向量中，又想对其中某些单个分量进行处理时，Spark ML提供了VectorIndexer类来解决向量数据集中的类别性特征转换。

通过为其提供maxCategories超参数，它可以自动识别哪些特征是类别型的，并且将原始值转换为类别索引。它基于不同特征值的数量来识别哪些特征需要被类别化，那些取值可能性最多不超过maxCategories的特征需要会被认为是类别型的。

特征选取：从大规模特征集中选取一个子集

特征选择（Feature Selection）指的是在特征向量中选择出那些“优秀”的特征，组成新的、更“精简”的特征向量的过程。它在高维数据分析中十分常用，可以剔除掉“冗余”和“无关”的特征，提升学习器的性能。

特征选择方法和分类方法一样，也主要分为有监督（Supervised）和无监督（Unsupervised）两种，卡方选择则是统计学上常用的一种有监督特征选择方法，它通过对特征和真实标签之间进行卡方检验，来判断该特征和真实标签的关联程度，进而确定是否对其进行选择。