슬라이드 1: 课程标题介绍

同学们好!

今天我们来学习机器学习流水线（Machine Learning Pipeline)。

这是一个非常重要的概念，特别是当我们需要处理复杂的机器学习任务时。

流水线可以让我们把多个数据处理步骤串联起来，形成一个完整的、可重复使用的机器学习工作流程。

今天的内容主要分为两个部分：

- 首先我们会学习流水线的基本概念
- 然后我们会看看流水线是如何工作的

大家准备好了吗？我们开始吧!

슬라이드 2: 流水线概念介绍

好，在正式介绍流水线之前，我们需要先了解几个核心概念。

这些概念是理解整个流水线架构的基础。

首先是 **DataFrame**。

大家应该还记得我们之前学过的RDD，对吧？

DataFrame可以说是RDD的升级版。

它使用Spark SQL（Spark SQL）中的DataFrame作为数据集，最大的优势是什么呢？

就是它可以容纳(**róng nà)**各种数据类型。

与RDD相比，DataFrame包含了schema信息，也就是说它知道每一列的数据类型是什么。

这让它更像传统数据库中的二维表格，使用起来更加方便。

在ML Pipeline（ML Pipeline）中，DataFrame被用来存储源数据。

比如说，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等。

这样大家理解了吗？

有什么问题吗？

슬라이드 3: Transformer概念详解

接下来我们看第二个重要概念——**Transformer（转换器）**。

这个名字听起来有点复杂，但是理解起来其实很简单。

Transformer就是一种可以将一个DataFrame转换为另一个DataFrame的算法。

我给大家举个例子：

假设我们训练了一个机器学习模型，这个模型就是一个Transformer。

为什么呢？

因为它可以把一个不包含预测标签的测试数据集DataFrame，通过预测过程，转化成另一个包含预测标签的DataFrame。

从技术角度来说，Transformer实现了一个叫做 transform() 的方法。

这个方法通过附加一个或多个列，

将一个DataFrame转换为另一个DataFrame。

大家可以把Transformer想象成一个"转换工厂"，原材料进去，加工后的产品出来。

明白了吗？

슬라이드 4: Estimator概念详解

现在我们来看第三个概念——**Estimator（评估器）**。

Estimator翻译成中文叫估计器或者评估器。

它是学习算法或在训练数据上的训练(**xùn liàn)**方法的概念抽象。

听起来有点抽象？我们用具体例子来解释。

在Pipeline里，Estimator通常是被用来操作DataFrame数据并生成一个Transformer。

注意这个关系：Estimator通过训练产生Transformer。

从技术上讲，

Estimator实现了一个叫 fit() 的方法，它接受一个DataFrame并产生一个转换器。

比如说，一个随机森林算法（Random Forest）就是一个Estimator。

当我们调用它的 fit() 方法，通过训练特征数据，就能得到一个随机森林模型。

而这个训练好的模型，就是一个Transformer了。

简单来说：Estimator + 训练数据 = Transformer。

大家理解了吗？

슬라이드 5: Pipeline和Parameter概念

好，现在我们来看最后两个重要概念。

Pipeline（流水线）：这就是我们今天的主角(**zhǔ jué)**!

Pipeline将多个工作流阶段，也就是转换器和估计器，连接在一起，形成机器学习的工作流，并获得结果输出。

你们可以把Pipeline想象成一条生产线。

原材料从一端进入，经过多个加工站点，

最后从另一端输出成品。

每个加工站点就是一个stage（阶段)。

Parameter（参数)：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（API)。

ParamMap是一组(参数，值)对。

这就像是给每个加工站点设置操作参数一样。

比如温度设置多少度，速度调到多快，这些都是参数。

大家觉得这些概念清楚了吗?

📊 슬라이드 6: 流水线工作过程介绍

现在我们开始学习流水线是如何工作的。

要构建一个Pipeline流水线，我们需要做什么呢？

首先需要定义Pipeline中的各个**流水线阶段PipelineStage**。这些阶段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，

我们就可以按照具体的处理逻辑有序地组织PipelineStages并创建一个Pipeline。

看屏幕上的代码：

pipeline = Pipeline(stages=[stage1,stage2,stage3])

这就是创建一个包含三个阶段的流水线。

然后我们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。

这个过程大家理解了吗？

📊 슬라이드 7: 流水线执行机制

现在我们详细看看流水线各个阶段是如何按顺序运行的。

这是一个很关键的理解点：输入的DataFrame在它通过每个阶段时被转换。

对于**转换器阶段**，在DataFrame上会调用 transform() 方法。

对于**评估器阶段**，先调用 fit() 方法来生成一个转换器，然后在DataFrame上调用该转换器的 transform() 方法。

所以你们看，无论是哪种阶段，最终都是通过 transform() 方法来处理数据的。

区别在于：

- 转换器直接调用 transform()
- 评估器先 fit() 再 transform()

这样的设计让整个流水线非常统一和优雅。大家觉得呢？

슬라이드 8: 流水线示例图解

现在我们看一个具体的例子来理解流水线的工作过程。

这里有一个流水线具有3个阶段:

- 前两个阶段: **Tokenizer (分词器)** 和 **HashingTF (哈希词频)**, 它们是转换器
- 第三个阶段: **Logistic Regression (逻辑回归)**, 它是评估器

大家看这个图, 下面一行表示流经这个流水线的数据, 圆柱表示DataFrame。

数据处理流程是这样的:

1. 原始文本 (Raw text) 通过Tokenizer变成单词 (Words)
2. 单词通过HashingTF变成特征向量 (Feature vectors)
3. 特征向量通过Logistic Regression训练得到模型

当我们在原始DataFrame上调用 Pipeline.fit() 方法时, 就是在执行这个流水线。

这个过程清楚吗? 有问题的话可以随时问我。

슬라이드 9: PipelineModel概念

这里有一个非常重要的概念需要大家理解:

流水线本身就是一个评估器 (Estimator)!

因此, 在流水线的 fit() 方法运行之后, 会产生一个**流水线模型 (PipelineModel)**。

这个PipelineModel是什么呢?

它是一个转换器 (Transformer), 可在测试数据的时候使用。

看这个图, 你们可以看到:

- 训练阶段: Pipeline (Estimator) 通过 fit() 产生 PipelineModel (Transformer)
- 预测阶段: PipelineModel 通过 transform() 处理新数据得到预测结果

这就是为什么我们说"训练一次, 使用多次"。

训练得到模型后, 就可以反复用这个模型来处理新的数据了。

大家明白这个概念了吗?

슬라이드 10: 参数设置详解

现在我们来学习如何设置参数。这在实际使用中非常重要。

MLlib评估器和转换器使用统一的API（API）指定参数。这里，Param是一个自描述包含文档的命名参数，而ParamMap是一组(参数,值)对。

将参数传递给算法主要有以下两种方法：

方法1：设置实例的参数 例如，lr是的一个LogisticRegression实例，用 lr.setMaxIter(10) 进行参数设置以后，可以使 lr.fit() 至多迭代10次。

方法2：传递ParamMap给fit()或transform()函数 ParamMap中的任何参数，将覆盖先前通过set方法指定的参数。

特别需要注意的是：参数同时属于评估器和转换器的特定实例。

如果同一个流水线中的两个算法实例

比如LogisticRegression实例lr1和lr2），都需要设置maxIter参数，

则可以建立一个ParamMap： ParamMap(lr1.maxIter -> 10, lr2.maxIter -> 20)

然后传递给这个流水线。这样就可以为不同的组件设置不同的参数了。

大家理解了吗？