9.4 Spark ML 基本统计工具 - 课堂讲授脚本

슬라이드 1 - 课程导入

大家好！今天我们要学习的是 Spark ML (Machine Learning) 中的基本统计工具。

这是一个非常重要的章节，因为统计分析是机器学习的基础。

现在请大家看一下今天的学习目标。

我们会学习三个主要内容：

首先是相关性分析，这个在数据科学中非常常用，可以帮助我们了解不同变量之间的关系。

然后是假设检验，这是验证我们分析结果是否具有统计意义的重要方法。

最后是汇总统计，这个功能可以快速给我们提供数据的基本统计信息。

好，我们开始第一部分。

슬라이드 2 - 9.4.1 相关性介绍

现在我们进入第一个主题 - 相关性分析。

同学们，相关性分析在我们日常的数据分析工作中是非常常见的。

比如说，你想知道学习时间和考试成绩之间是否有关系，或者广告投入和销售额**xiāo shòu é**之间的关联程度，这些都需要用到相关性分析。

Spark ML (Machine Learning) 包为我们提供了非常强大的功能，可以在多组数据之间计算两两相关性。目前主要支持两种方法：

第一种是皮尔逊相关 (Pearson correlation)，这是最常用的线性相关系数。

第二种是斯皮尔曼相关 (Spearman correlation)，这个更适合处理非线性关系。

现在我们来看看皮尔逊相关系数的数学公式。这个公式可能看起来有点复杂，但是理解起来并不难。

皮尔逊相关系数的取值范围是从 -1 到 +1。当结果是 0 的时候，表示两个变量之间没有线性关系；正值表示正相关，也就是一个变量增加，另一个也倾向于增加；负值表示负相关，一个增加另一个倾向于减少。

系数的绝对值越大，相关性就越强。

슬라이드 3 - 斯皮尔曼相关系数

接下来我们看斯皮尔曼相关系数。

同学们要注意，皮尔逊相关系数主要适用于服从正态分布的变量。但是在实际工作中，我们经常遇到不服从正态分布的数据，这时候就需要用斯皮尔曼相关系数了。

斯皮尔曼相关系数有一个很大的优势，就是它可以更好地处理变量之间的排序关系，不仅仅局限于线性关系。

这里的计算公式稍微复杂一些，但是不用担心，Spark ML (Machine Learning) 会帮我们自动计算。

计算步骤详解（通过一个例子）(spearman 구하는 예제)

假设我们研究5个学生的 学**习时间（X）** 和 **考试成绩（Y）** 之间的关系。

原始数据：

学生	学习时间（小时）	考试成绩（分）
甲jiǎ	4	85
乙yǐ	1	60
丙bǐng	7	90
丁dīng	3	70
戊wù	5	80

第一步：分别给两个变量排名

- **变量X（学习时间）排名：** 数值越大，排名数字越大（1为最小）。
    - 乙(1小时) -> 第1名
    - 丁(3小时) -> 第2名
    - 甲(4小时) -> 第3名
    - 戊(5小时) -> 第4名
    - 丙(7小时) -> 第5名
- **变量Y（考试成绩）排名：** 同样，分数越高，排名越大。
    - 乙(60分) -> 第1名
    - 丁(70分) -> 第2名
    - 戊(80分) -> 第3名
    - 甲(85分) -> 第4名
    - 丙(90分) -> 第5名

第二步：将排名整理到表格，并计算每一对的等级差 d_i 和差的平方 d_i^2

学生	X排名（Rx）	Y排名（Ry）	等级差 $d_i = Rx - Ry$	d_i^2
甲	3	4	3 - 4 = **-1**	1
乙	1	1	1 - 1 = **0**	0
丙	5	5	5 - 5 = **0**	0
丁	2	2	2 - 2 = **0**	0
戊	4	3	4 - 3 = **+1**	1

第三步：对 d_i^2 求和

$$\sum d_i^2 = 1 + 0 + 0 + 0 + 1 = 2$$

第四步：代入斯皮尔曼公式（n=5）

$$\rho = 1 - \frac{6 \times \sum d_i^2}{n(n^2 - 1)} = 1 - \frac{6 \times 2}{5 \times (25 - 1)} = 1 - \frac{12}{120} = 1 - 0.1 = 0.9$$

如何理解“等级差”d_i 的含义？

d_i 的大小和符号直接反映了两个变量排名的一致性：

- $d_i = 0$: **完美一致**。 该个体在两个变量上的排名完全相同。
    - 例如学生乙，学习时间最少（第1名），成绩也最低（第1名）。
- d_i **为很小的正/负数**: **基本一致**。 排名略有差异。
- d_i **绝对值很大**: **不一致**。 两个排名相差甚远。
    - 假设一个学生学习时间排第1（最短），但成绩排第5（最高），则 $d = 1 - 5 = -4$，这是一个很大的负差值，表明“学得少但考得好”的异常情况。
- **所有 d_i 的和 $\sum d_i$ 恒等于 0**。这是排名的数学性质，所以在计算中我们使用 d_i^2 来消除正负影响，只关心差异的“程度”。

核心逻辑： 如果两个变量的**单调关系越强**，那么每个个体的 d_i 就会越接近 0（排名越同步），从而 $\sum d_i^2$ 就越小，最终计算出的斯皮尔曼相关系数 ρ 的绝对值就越接近 1。

总结：“每一对观测值的等级差”d_i 是连接原始数据排名与最终相关系数的桥梁，它量化了每个数据点在两个维度上排名不一致的程度。

슬라이드 4 - 代码实践开始

好，理论讲完了，现在我们开始实际操作。这是编程课程最有趣的部分！

第一步，我们需要导入相关性分析所需要的包。请大家注意这几个 import 语句：

首先要导入 SparkSession，这是我们使用 Spark 的入口点。

然后是 Vectors，用来处理向量数据。

Row 用来创建数据行。

最重要的是 Correlation，这个就是我们今天要用的相关性分析工具。

这些包的导入顺序不是强制的，但是我建议大家按照这个顺序，这样比较有逻辑性。

슬라이드 5 - 创建实验数据

第二步，我们来创建一些实验数据。

首先创建一个 SparkSession。如果你们之前已经创建过，这一步可以跳过，但是为了确保代码的完整性，我们还是写上。

然后我们创建一些示例数据。注意这里我特意使用了两种类型的向量：

第一个是稀疏向量（sparse vector），只指定非零元素的位置和值。这在处理大型稀疏数据时非常有效。

后面两个是稠密向量（dense vector），包含所有元素的值。

最后，我们把这些向量数据转换成 DataFrame。这是 Spark 中处理结构化数据的标准格式。

슬라이드 6 - 查看数据

让我们看看创建的数据是什么样子的。

使用 df.show() 可以显示 DataFrame 的内容。

大家可以看到，第一行显示的是稀疏向量的表示形式，后面两行是稠密向量。

这种数据格式在机器学习中非常常见，特别是在处理特征向量的时候。

슬라이드 7 - 计算皮尔逊相关系数

现在进入关键步骤 - 计算相关性矩阵。

我们调用 Correlation.corr() 函数，传入我们的 DataFrame 和特征列的名称。

默认情况下，这个函数计算的是皮尔逊相关系数。

然后我们提取相关性矩阵的值并打印出来。

大家看这个结果，这是一个 4x4 的矩阵，因为我们有 4 个特征。

对角线上都是 1，这很正常，因为每个变量和自己的相关性当然是完全相关的。

其他位置的数值表示不同特征之间的相关性。比如第一个特征和第二个特征的相关性是 0.97，这是很强的正相关。

슬라이드 8 - 计算斯皮尔曼相关系数

现在我们来计算斯皮尔曼相关系数，看看结果有什么不同。

注意这里我们在 corr() 函数中添加了第三个参数 "spearman"，这样就指定使用斯皮尔曼方法。

大家比较一下两个结果，可以发现数值确实有所不同。

斯皮尔曼相关系数通常会给出不同的结果，特别是当数据不服从正态分布时，差异会更明显。

在实际应用中，我们通常会同时计算两种相关系数来全面了解变量之间的关系。

```
from pyspark.sql import SparkSession
from pyspark.ml.linalg import Vectors
from pyspark.sql import Row
from pyspark.ml.stat import Correlation


def main():
    # 1. 创建 SparkSession
    spark = (
```

```
        SparkSession.builder
        .appName("CorrelationExample")
        .master("local[*]")
        .getOrCreate()
)

# 2. 构造实验数据（稀疏向量 + 稠密向量）
data = [
        Vectors.sparse(4, [(0, 2.0), (2, -1.0)]),
        Vectors.dense([3.0, 0.0, 4.0, 5.0]),
        Vectors.dense([6.0, 8.0, 0.0, 7.0])
]
# X =
# [
#       [2.0, 0.0, -1.0, 0.0],
#       [3.0, 0.0, 4.0, 5.0],
#       [6.0, 8.0, 0.0, 7.0]
# ]

# 3. 转换为 DataFrame
rows = [Row(features=vec) for vec in data]
df = spark.createDataFrame(rows)

# 4. 查看 DataFrame
print("=== Input DataFrame ===")
df.show(truncate=False)

# 5. 计算 Pearson 相关系数矩阵
corr_matrix_pearson = Correlation.corr(df, "features", "pearson")
pearson_matrix = corr_matrix_pearson.collect()[0]["pearson(features)"]

print("₩n=== Pearson Correlation Matrix ===")
print(pearson_matrix)

# 6. 计算 Spearman 相关系数矩阵
corr_matrix_spearman = Correlation.corr(df, "features", "spearman")
spearman_matrix = corr_matrix_spearman.collect()[0]["spearman(features)"]
```

```
    print("₩n=== Spearman Correlation Matrix ===")
    print(spearman_matrix)

    # 7. 关闭 SparkSession
    spark.stop()


if __name__ == "__main__":
    main()
```

슬라이드 9 - 假设检验介绍

好，现在我们进入第二个重要主题 - 假设检验 (**jiǎ shè jiǎn yàn** Hypothesis Testing)。

假设检验在统计学中是一个非常重要的概念。

简单来说，它帮助我们判断我们观察到的现象是真实存在的，还是只是巧合(**qiǎo hé)**。

卡方检验 (Chi-square test) 是假设检验中应用最广泛的方法之一。

它主要有两种用途：

第一种是适合度检验，用来验证观察到的频率分布是否符合理论分布。

第二种是独立性检验，用来验证两个变量之间是否相互独立。

目前 Spark ML 支持的是皮尔逊卡方检验的独立性检验。

独立性检验通常使用列联表 (contingency table) 来记录数据。

列联表就是一个交叉分类的频数分布表，在社会科学研究中经常用到。

간단한 예시: 성별과 선호하는 커피 종류(contingency table)

	아메리카노	카페라떼	에스프레소	**행 합계**
남성	20	15	10	**45**
여성	25	30	5	**60**
열 합계	**45**	**45**	**15**	**105**

contingency"의 핵심은 '다른 것에 의존하거나, 예기치 않게 발생할 수 있는 가능성'을 의미합니다

슬라이드 10 - 假设检验的步骤

独立性检验有一个标准的流程，让我给大家详细解释一下。

首先，我们需要建立假设：

- 原假设 H0：两个变量相互独立 :귀무 가설
- 备择假设 H1：两个变量相互不独立 **bèi zé jiǎ shè**。대립가설

然后计算自由度。

这个公式很重要：如果列联表有 r 行 c 列，那么自由度就是 (r-1)(c-1)。

接着计算卡方统计值，使用这个求和公式。这里 O 表示观察频数，E 表示期望频数。

最后，根据设定的置信水平，查找卡方分布的临界值，与我们计算的统计值进行比较，就可以决定是否拒绝原假设。

这个过程听起来复杂，但是 Spark ML 会帮我们自动完成所有计算。

1. 계산 공식

i행 j열에 있는 셀의 기대빈도 E_{ij}는 다음과 같이 계산합니다.

$$E_{ij} = \frac{(\text{행 } i\text{의 합계}) \times (\text{열 } j\text{의 합계})}{\text{전체 표본 크기}}$$

$$E_{ij} = \frac{(R_i) \times (C_j)}{N}$$

- R_i: i번째 행의 **행 합계**
- C_j: j번째 열의 **열 합계**
- N: 전체 관측치 수 (표의 오른쪽 하단 총계)

2. 계산 예시

다음은 성별(남/여)과 흡연 여부(예/아니오)에 대한 교차표입니다.

	흡연 (예)	흡연 (아니오)	**행 합계**
남성	30 (관측)	70	**100**
여성	20	80	**100**
열 합계	**50**	**150**	**200 (N)**

셀 (남성, 흡연 예)의 기대빈도 계산:

- **행 합계 (남성)**: 100
- **열 합계 (흡연 예)**: 50
- **전체 (N)**: 200

$$E_{\text{남,예}} = \frac{100 \times 50}{200} = \frac{5000}{200} = 25$$

셀 (여성, 흡연 아니오)의 기대빈도 계산:

- **행 합계 (여성)**: 100
- **열 합계 (흡연 아니오)**: 150

$$E_{\text{여,아니오}} = \frac{100 \times 150}{200} = \frac{15000}{200} = 75$$

3. 완성된 기대빈도표

모든 셀에 대해 계산하면 다음과 같습니다.

	흡연 (예)	흡연 (아니오)
남성	**25**	**75**
여성	**25**	**75**

확인: 각 행/열의 기대빈도 합계는 원본 표의 행/열 합계와 동일합니다.

- 남성 행: 25 + 75 = 100 ✓
- 흡연 예 열: 25 + 25 = 50 ✓

4. 계산의 논리 (왜 이 공식을 쓰는가?)

귀무가설(성별과 흡연은 독립)이 참이라면:

- 전체 200 명 중 흡연자는 50 명이므로, **전체 흡연율** = 50/200 = 0.25 (25%).
- 성별과 무관하게, **어떤 집단이든 동일한 흡연율(25%)**이 적용되어야 합니다.
- 따라서 남성 100 명 중 기대되는 흡연자 수는: $100 \times 0.25 = 25$.
- 이것이 공식 $(100 \times 50)/200$과 정확히 일치합니다.

5. 카이제곱 검정에서의 활용

카이제곱 통계량(χ^2)은 관측빈도(O)와 기대빈도(E)의 차이를 기반으로 계산됩니다.

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

- **차이가 클수록** → 두 변수가 독립일 가능성 낮음 → 귀무가설 기각 증거.
- 위 예시에서 **남성, 흡연 예** 셀의 기여도:

$$\frac{(30-25)^2}{25} = \frac{25}{25} = 1$$

6. 데이터 정리

셀 위치	관측빈도 (O)	기대빈도 (E)
남성, 흡연 예	30	25
남성, 흡연 아니오	70	75
여성, 흡연 예	20	25
여성, 흡연 아니오	80	75

전체 표본 수 N = 200

---

7. 카이제곱 통계량 계산 공식

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

각 셀에 대해 $(O-E)^2/E$를 계산한 후 모두 합산합니다.

셀별 계산

1. **남성, 흡연 예**:

$$\frac{(30-25)^2}{25} = \frac{25}{25} = 1.0$$

2. **남성, 흡연 아니오**:

$$\frac{(70-75)^2}{75} = \frac{25}{75} \approx 0.3333$$

3. **여성, 흡연 예**:

$$\frac{(20-25)^2}{25} = \frac{25}{25} = 1.0$$

4. **여성, 흡연 아니오**:

$$\frac{(80-75)^2}{75} = \frac{25}{75} \approx 0.3333$$

합산

$$\chi^2 = 1.0 + 0.3333 + 1.0 + 0.3333 = 2.6666$$

반올림하면:

$$\boxed{\chi^2 \approx 2.667}$$

---

8. 검정 수행 (추가 설명)

- **자유도(df)**:

$$df = (\text{행수} - 1) \times (\text{열수} - 1) = (2-1) \times (2-1) = 1$$

- **유의수준 α = 0.05** 일 때, **카이제곱 임계값($\chi^2_{(1)}$)** ≈ 3.841 입니다.
- **결론**:
  계산된 $\chi^2 \approx 2.667 < 3.841$ 이므로,
  p-값 > 0.05 로 귀무가설(성별과 흡연은 독립)을 기각할 충분한 증거가 없습니다.
  → "이 데이터에서는 성별과 흡연 여부 사이에 통계적으로 유의한 연관성이 있다고 보기 어렵다."

✅ **최종 답변**

제시된 예제의 **카이제곱 통계량은 약 2.667** 입니다

슬라이드 11 - 卡方检验代码实践

现在我们开始卡方检验的代码实践。

首先导入必要的包。注意这里我们导入的是 ChiSquareTest，这是专门用于卡方检验的工具。

Vectors 还是用来处理向量数据的。

```
# 第1步：导入卡方检验所需要的包。
from pyspark.sql import SparkSession, Row  # Row：把 (label, features) 这种 Python 元组转换为 DataFrame 行对象。
from pyspark.ml.linalg import Vectors  # Vectors：用来创建特征向量（features vector）。
from pyspark.ml.stat import ChiSquareTest  # hiSquareTest：用于做卡方独立性检验（Chi-square test of independence）。
```

슬라이드 12 - 创建检验数据

第二步，我们创建一些用于检验的数据。

这里的数据格式稍微不同，每一行包含一个标签（label）和一个特征向量（features)。

标签是 0.0 或 1.0，表示两个不同的类别。

特征向量包含两个数值，可能代表某些测量值。

这种数据格式在分类问题中非常常见。

```
def main():
    # 第2步：创建 SparkSession
    spark = (
        SparkSession.builder
        .appName("ChiSquareTestExample")
        .getOrCreate()
    )

    # 第2步：创建数据集（5个样本、2个特征维度，label ∈ {0.0, 1.0}）
    data = [
        (0.0, Vectors.dense([3.5, 40.0])),
        (0.0, Vectors.dense([3.5, 30.0])),
```

```
        (1.0, Vectors.dense([1.5, 30.0])),
        (0.0, Vectors.dense([1.5, 20.0])),
        (0.0, Vectors.dense([0.5, 10.0])),
]

# DataFrame 변환
df = spark.createDataFrame([Row(label=label, features=vec) for label, vec in data])

print("=== Input DataFrame ===")
df.show(truncate=False)
```

슬라이드 13 - 执行卡方检验

现在执行卡方独立性检验。

我们调用 ChiSquareTest.test() 函数，传入 DataFrame、特征列名和标签列名。

这个函数会自动将（特征，标签）对转换成列联矩阵，然后计算卡方统计量。

输出结果包含三个重要信息：

- pValues：p 值，用来判断是否显著
- degreesOfFreedom：自由度
- statistics：卡方统计值

```
# 第3步：执行卡方独立性检验
# chiSquareTest.test(df, "features", "label") 会对 features 向量中的每一维特征分别与
# label做独立性检验：
#
# - 内部逻辑是把(feature_j, label) 统计成 contingency table（列联表 / 列联矩阵），
# - 然后计算该特征与标签是否独立的卡方统计量
chi_square_result = ChiSquareTest.test(df, "features", "label")

print("=== ChiSquareTest.test(...) Result (full row) ===")
chi_square_result.show(truncate=False)
```

슬라이드 14-16 - 检验结果分析

让我们分别查看这三个结果。

首先是 p 值。

P 值告诉我们观察到的结果是由随机因素造成的概率。

一般来说，如果 p 值小于 0.05，我们就认为结果是显著的。

从这个结果可以看出，p 值都大于 0.05，所以我们不能拒绝原假设，也就是说两个变量可能是独立的。

自由度是 [2, 3]，这对应我们数据的维度。

统计值是 [1.875, 1.875]，这些是计算出的卡方统计量。

在实际应用中，我们通常最关注 p 值，它直接告诉我们结果是否具有统计意义。

```
# 第4步：分别获取 pValues / degreesOfFreedom / statistics
# pValues：每个特征维度对应一个 p - value（右尾概率，用于显著性判断）。
# degreesOfFreedom：每个特征维度对应一个自由度（df）。
# statistics：每个特征维度对应一个卡方统计量（χ²）。
# 你示例里的输出对应：
# -pValues = [0.3916..., 0.5987...]
# -degreesOfFreedom = [2, 3]
# -statistics = [1.875..., 1.875]

print("p-values:")
chi_square_result.select("pValues").show(truncate=False)

print("Degrees of Freedom:")
chi_square_result.select("degreesOfFreedom").show(truncate=False)

print("Statistics:")
chi_square_result.select("statistics").show(truncate=False)

spark.stop()
    # 전체 결론 (Overall conclusion)
    # 두 개의 feature 모두에 대해: p-value > 0.05
    # 따라서 이 데이터셋에서는: 任何一个特征与标签之间都没有显著的统计学相关性
```

```
        # 즉, 현재 데이터에서는
        # 어떤 feature 도 label 을 구분하는 데 통계적으로 유의하지 않음
        # feature selection 관점에서는  → 이 두 feature 는 label 예측에 유용하지 않다고 해석 가능


if __name__ == "__main__":
    main()
```

슬라이드 17 - 汇总统计介绍

最后我们来学习汇总统计 (Summary Statistics)。

在 Spark 3.0 版本中，我们可以使用 Summarizer 包来快速获取数据的各种统计信息。

这些统计信息包括：最大值、最小值、平均值、总和、方差、标准差、非零元素个数，以及总计数等等。

这在数据探索和数据质量检查中非常有用。

现在我们来看一个实际例子，演示如何计算数据的平均值和方差。

슬라이드 18 - 汇总统计代码准备

首先还是导入必要的包。

除了前面用过的包，这里还需要导入 Summarizer 和 col 函数。

col 函数用来引用 DataFrame 中的列。

```
# 第1步：导入包
# Vectors：创建 DenseVector（稠密向量）。
# Summarizer：对向量列做汇总统计（summary statistics），支持 mean / variance 等指标，并支持权重（weight）。
# col：把列名包装成 Column 对象，传入 Summarizer.summary()。


from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.ml.linalg import Vectors
from pyspark.ml.stat import Summarizer
```

슬라이드 19 - 创建带权重的数据

这次我们创建的数据有点特殊，包含权重信息。

第一组数据的权重是 1.0，第二组数据的权重是 3.0。

在很多实际应用中，不同的数据点可能有不同的重要性，这时候就需要使用权重。

总权重是 4.0，这个在后面的计算中会用到。

```
def main():
    # 第2步：创建 SparkSession
    spark = (
        SparkSession.builder
        .appName("SummarizerExample")
        .getOrCreate()
    )

    # 第2步：创建数据集（两组数据，权重分别为 1.0 和 3.0）
    # 第2步：构造数据与DataFrame
    # df 有两列：
    # features：向量列（vector column）
    # weight：权重列（double）
    # 权重分别为 1.0 和 3.0，权重和为 4.0。
    data = [
        (Vectors.dense([1.0, 2.0, 4.0]), 1.0),
        (Vectors.dense([4.0, 3.0, 6.0]), 3.0),
    ]

    columns = ["features", "weight"]
    df = spark.createDataFrame(data, columns)

    print("=== Input DataFrame ===")
    df.show(truncate=False)
    df.printSchema()
```

슬라이드 20 - 计算加权统计量

现在我们计算加权平均值和加权方差。

使用 Summarizer.metrics() 函数指定我们要计算的统计量类型。

summary() 函数接受特征列和权重列作为参数。

结果显示，加权平均值是 [3.25, 2.75, 5.5]，加权方差是 [4.5, 0.5, 2.0]。

这些结果考虑了每个数据点的权重，所以更能反映数据的真实分布特征。

```
# 第3步：加权 mean / variance 加权统计（weighted mean / weighted variance）
# 调用： summary(col("features"), col("weight"))
# 结果：
# 加权均值（weighted mean）会受权重影响（第二行权重更大，所以均值更靠近第二行向量）。
# 加权方差（weighted variance）在该例中输出为[4.5, 0.5, 2.0]。

weighted_summary = df.select(
    Summarizer.metrics("mean", "variance")
    .summary(col("features"), col("weight"))
    .alias("summary")
)

weighted_mean = weighted_summary.select("summary.mean").first()[0]
weighted_variance = weighted_summary.select("summary.variance").first()[0]

print("=== Weighted (with weight column) ===")
print("Mean Values:", weighted_mean)
print("Variance Values:", weighted_variance)
```

슬라이드 21 - 计算无权重统计量

为了对比，我们也计算一下无权重的统计量。

这次 summary() 函数只传入特征列，不传入权重列。

可以看到，无权重的平均值是 [2.5, 2.5, 5.0]，与加权结果有明显差异。

这说明权重确实影响了统计结果。在实际应用中，我们需要根据具体情况选择是否使用权重。

```
    # 第4步：无权重统计（unweighted mean / variance）
    # 调用：summary(col("features"))
    # 结果：
    # 无权重均值就是普通平均：([1, 2, 4] + [4, 3, 6]) / 2 = [2.5, 2.5, 5.0]
    # 方差同样输出为[4.5, 0.5, 2.0]（Spark Summarizer 的 variance 计算方式与其内部定义一致）。
    unweighted_summary = df.select(
        Summarizer.metrics("mean", "variance")
        .summary(col("features"))
        .alias("summary")
    )

    unweighted_mean = unweighted_summary.select("summary.mean").first()[0]
    unweighted_variance = unweighted_summary.select("summary.variance").first()[0]

    print("=== Unweighted (no weight) ===")
    print("平均值：", unweighted_mean)
    print("方差：", unweighted_variance)

    spark.stop()


if __name__ == "__main__":
    main()
```

课程总结

好，今天的课程就到这里。让我们回顾一下学到的内容：

第一，我们学习了相关性分析，包括皮尔逊相关和斯皮尔曼相关。这些工具可以帮助我们理解变量之间的关系。

第二，我们学习了假设检验，特别是卡方独立性检验。这是验证分析结果统计意义的重要方法。

第三，我们学习了汇总统计，这是数据探索的基础工具。

这些统计工具在 Spark ML 中都有很好的实现，可以处理大规模数据，非常适合在生产环境中使用。

课后大家可以用自己的数据试试这些方法，多练习才能真正掌握。