9.4 基本统计工具(Statistical Methods in MLlib)



9.4.1 相关性(correlation)
9.4.2 假设检验(hyperthesis testing)
9.4.3 汇总统计(summary statistics)

9.4.1 相关性



计算两组数据之间的相关性是统计学中的常见运算。spark.ml包提供了在多组数据之间计算两两相关性的方法。目前，支持的相关性方法有皮尔逊（pearson)相关和斯皮尔曼(spearman)相关，其相关系数可以反映两个变量之间变化趋势的方向及程度。

皮尔逊相关系数是一种线性相关系数，如下，

$$\rho_{X,Y} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

皮尔逊相关系数的输出范围介于-1到+1之间，0表示无相关性，正值表示正相关，负值表示负相关，相关系数的绝对值越大，相关度越强，相关系数越接近于0，相关度越弱。

9.4.1 相关性



斯皮尔曼相关系数：皮尔森相关系数主要用于服从正态分布的变量，对不服从正态分布的变量，可以使用斯皮尔曼相关系数进行相关性分析，斯皮尔曼相关系数可以更好地用于测量变量的排序关系，其计算公式为：

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(x_i - y_i)^2}{n(n^2-1)}$$

- ρ：斯皮尔曼相关系数
- d_i：每一对观测值的等级差 ($di=xi-yi$)
- n：观测值的对数

9.4.1 相关性



下面我们给出使用spark.ml库提供的方法进行相关性分析的实例

第1步：导入相关性方法所需要的包。

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

9.4.1 相关性



第2步：创建实验数据，并转化成DataFrame。

```
# 创建一个 SparkSession（如果尚未创建）
>>> spark = SparkSession.builder.appName("example").getOrCreate()

# 示例数据，包含稀疏向量和稠密向量
>>> 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])
]

# 将数据集转换为 DataFrame
>>> rows = [Row(features=vec) for vec in data]
>>> df = spark.createDataFrame(rows)
```

9.4.1 相关性



```
# 打印 DataFrame
>>> df.show()
+--------------------+
|            features|
+--------------------+
|(4,[0,2],[2.0,-1.0])|
|    [3.0,0.0,4.0,5.0]|
|    [6.0,8.0,0.0,7.0]|
+--------------------+
```

9.4.1 相关性



第3步：调用Correlation包中的corr()函数来获取皮尔森相关性系数。

```
# 计算特征列的相关性矩阵
>>> corr_matrix = Correlation.corr(df, "features")

# 提取相关性矩阵的值
>>> coeff1 = corr_matrix.collect()[0]["pearson(features)"]

>>> print(coeff1)
```

9.4.1 相关性



第4步：使用指定的斯皮尔曼相关性方法计算输入数据集的相关性。

```
# 计算特征列的Spearman相关性矩阵
>>> corr_matrix_spearman = Correlation.corr(df, "features", "spearman")

# 提取Spearman相关性矩阵的值
>>> coeff2 = corr_matrix_spearman.collect()[0]["spearman(features)"]

>>> print(coeff2)
DenseMatrix([[1.       , 0.8660254, 0.5      , 1.       ],
             [0.8660254, 1.       , 0.       , 0.8660254],
             [0.5      , 0.       , 1.       , 0.5      ],
             [1.       , 0.8660254, 0.5      , 1.       ]])
             0.5
1.0                          0.8660254037844387    0.5
                    1.0
```

9.4.2 假设检验



假设检验用来确定结果是否具有统计意义，以及该结果是否偶然发生。而卡方检验是用途非常广的一种假设检验方法，包括适合度检验和独立性检验。

- 适配度检验用于验证一组观察值的次数分配是否异于理论上的分配。独立性检验用于验证从两个变量抽出的配对观察值组是否互相独立。Spark.ml库目前支持皮尔森卡方检验的独立性检验。

- 独立性检验一般采用列联表的形式记录观察数据，列联表是由两个以上的变量进行交叉分类的频数分布表。

9.4.2 假设检验



独立性检验的步骤为，首先需要建立原假设，即H0：两变量相互独立，H1：两变量相互不独立。接着计算自由度和卡方检验的统计值。

- 自由度的计算公式为（假设列联表共有 r 行 c 列）：

$$df = (r-1)(c-1)$$

- 卡方检验的统计值的计算公式为：卡方检验的统计值的计算公式为：

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}$$

根据设定的置信水准，查出自由度为df的卡方分配临界值，比较它与卡方检验的统计值可以推测是否能拒绝原假设。

9.4.2 假设检验



下面我们给出使用spark.ml库提供的方法进行相关性分析的实例

第1步：导入卡方检验所需要的包。

```
>>> from pyspark.ml.linalg import Vectors
>>> from pyspark.ml.stat import ChiSquareTest
```

9.4.2 假设检验



第2步：创建实验数据

```
# 创建一个 SparkSession（如果尚未创建）
>>> spark = SparkSession.builder.appName("example").getOrCreate()
# 创建数据集
>>> 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]))
]

# 将数据集转换为DataFrame
>>> df = spark.createDataFrame([Row(label=label, features=vec) for label, vec in data])
```

```
+-----+----------+
|label|features  |
+-----+----------+
|0.0  |[3.5,40.0]|
|0.0  |[3.5,30.0]|
|1.0  |[1.5,30.0]|
|0.0  |[1.5,20.0]|
|0.0  |[0.5,10.0]|
+-----+----------+
```

9.4.2 假设检验



第3步：调用ChiSquareTest包中的test ()函数，将(特征，标签)对转换成一个列联矩阵，计算卡方统计量。

```
# 执行卡方独立性检验
>>> chi_square_result = ChiSquareTest.test(df, "features", "label")

>>> chi_square_result.show()
```

```
=== ChiSquareTest.test(...) Result (full row) ===
+----------------------------------------+---------------+--------------------------+
|pValues                                 |degreesOfFreedom|statistics                |
+----------------------------------------+---------------+--------------------------+
|[0.3916056266767989,0.5987516330675617]|[2, 3]          |[1.8750000000000002,1.875]|
+----------------------------------------+---------------+--------------------------+
```

9.4.2 假设检验



第4步：分别获取卡方分配右尾机率、自由度、统计值。

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

```
p-values:
+----------------------------------------+
|pValues                                 |
+----------------------------------------+
|[0.3916056266767989,0.5987516330675617]|
+----------------------------------------+
```

9.4.2 假设检验



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

```
Degrees of Freedom:
+----------------+
|degreesOfFreedom|
+----------------+
|[2, 3]          |
+----------------+
```

9.4.2 假设检验



```
# 打印statistics
>>> print("Statistics:")
>>> chi_square_result.select("statistics").show(truncate=False)
```

```
Statistics:
+-------------------------+
|statistics               |
+-------------------------+
|[1.8750000000000002,1.875]|
+-------------------------+
```

9.4.3 汇总统计



Spark3.0版本中可以使用Summarizer中的包来查看列的最大值、最小值、平均值、总和、方差、标准差、非零数，以及总计数等信息。下面给出使用Summarizer包的工具查看数据平均值和方差的实例：

第1步：导入汇总统计所需要的包

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

9.4.3 汇总统计



第2步：创建实验数据，两组数据的权重分别为1.0和3.0，权重之和为4.0。

```
# 创建一个 SparkSession
>>> spark = SparkSession.builder.appName("example").getOrCreate()
# 创建数据集
>>> data = [
    (Vectors.dense([1.0, 2.0, 4.0]), 1.0),
    (Vectors.dense([4.0, 3.0, 6.0]), 3.0)
]
# 将数据集转换为 DataFrame
>>> columns = ["features", "weight"]
>>> df = spark.createDataFrame(data, columns)
```

9.4.3 汇总统计



```
# 显示数据集
>>> df.show()
+-------------+------+
|     features|weight|
+-------------+------+
|[1.0,2.0,4.0]|   1.0|
|[4.0,3.0,6.0]|   3.0|
+-------------+------+
```

9.4.3 汇总统计



第3步：计算得到数据的加权平均值和加权方差

```
>>> summary = df.select(Summarizer.metrics("mean", "variance")
                .summary(col("features"), col("weight")).alias("summary"))
>>> meanVal = summary.select("summary.mean").first()[0]
>>> varianceVal = summary.select("summary.variance").first()[0]

>>> print("Mean Values:", meanVal)
Mean Values: [3.25,2.75,5.5]
>>> print("Variance Values:", varianceVal)
Variance Values: [4.5,0.5,2.0]
```

9.4.3 汇总统计



第4步：计算得到数据无权重下的平均值和方差

```
>>> summary = df.select(Summarizer.metrics("mean", "variance")
                        .summary(col("features")).alias("summary"))
>>> meanVal = summary.select("summary.mean").first()[0]
>>> varianceVal = summary.select("summary.variance").first()[0]

>>> print("平均值：", meanVal)
平均值： [2.5,2.5,5.0]
>>> print("方差：", varianceVal)
方差： [4.5,0.5,2.0]
```