山东大学计算机科学与技术学院

大数据分析实践课程实验报告

组员：陈德康 危弘毅 徐昌华 朱宣玮 郑坤武	
实验题目：Spark 实践	
实验学时：2	实验日期：2025.11.28

实验目标：

本实验旨在介绍学习者如何配置和运行 Apache Spark，以及如何使用 Spark 进行简单的数据处理和分析。实验将涵盖以下内容：

1.安装和配置 Apache Spark。

2.运行一个简单的 Spark 应用程序，以理解 Spark 的基本概念。

3.使用 Spark 进行数据处理和分析。

实验环境：

硬件环境

个人笔记本电脑

软件环境

操作系统：Windows 11

Java：java 17.0.8

Spark：PySpark

Python：3.11

Jupyter Notebook：VS Code Jupyter 插件

数据集

销售数据 http://storage.amesholland.xyz/sales_data.csv。

实验步骤与结果：

一、安装和配置 Apache Spark

本实验中，我们通过 pip 安装 PySpark。

Java 是 Spark 运行的必需环境。安装 PySpark 前，检查本地 Java 版本和 JAVA_HOME 环境变量是否已设置。

```
C:\Users\ASUS>Java --version
java 17.0.8 2023-07-18 LTS
Java(TM) SE Runtime Environment (build 17.0.8+9-LTS-211)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.8+9-LTS-211, mixed mode, sharing)
```

```
C:\Users\ASUS>echo %JAVA_HOME%
D:\Java\jdk-17
```

确认 Java 环境后，执行 pip install --upgrade PySpark 以安装最新版的 Apache Spark。

```
D:\>pip install --upgrade PySpark
Requirement already satisfied: PySpark in d:\python\python311\lib\site-packages (4.0.0)
Collecting PySpark
  Downloading pyspark-4.0.1.tar.gz (434.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434.2/434.2 MB 2.0 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: py4j==0.10.9.9 in d:\python\python311\lib\site-packages (from PySpark) (0.10.9.9)
Building wheels for collected packages: PySpark
  Building wheel for PySpark (pyproject.toml) ... done
  Created wheel for PySpark: filename=pyspark-4.0.1-py2.py3-none-any.whl size=434813901 sha256=185feaef4350140474d8cd096
55e402446b322a9bb193de848b97757e53d294e
  Stored in directory: d:\python\pipcache\wheels\76\f8\dc\9195b82b8586561710077d42370ae400e8a023af43052d1fec
Successfully built PySpark
```

```
Successfully installed PySpark-4.0.1
```

PySpark 已安装，版本为 4.0.1。

二、运行一个简单的 Spark 应用程序

对示例数据执行简单的筛选和聚合操作

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("SimpleSparkApp") \
    .getOrCreate()

data = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

result = data.filter(data["Product_Category"] == "Accessories") \
    .groupBy("Date") \
    .sum("Revenue")

result.show()
```

Pytho

```
+----------+------------+
|      Date|sum(Revenue)|
+----------+------------+
| 2016/5/12|       21505|
| 2015/7/29|        5994|
|  2013/8/2|       17700|
|2013/11/10|       20464|
|2015/11/24|       28005|
|2015/12/23|       19262|
| 2014/2/28|       28865|
| 2016/3/22|       16684|
|2015/12/19|       30493|
| 2013/11/3|       26902|
| 2016/4/17|       22595|
| 2016/4/25|       19838|
|  2014/1/4|       14332|
```

三、使用 Spark，对示例数据进行数据处理和分析

该部分的完整代码和输出在 analysis.ipynb 中。

1. 导入需要的库，初始化 SparkSession

```python
# 1. 导入需要的库，初始化SparkSession

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum, avg, count, max, min, when, month, year, expr, lit
from pyspark.sql.window import Window
from pyspark.ml.feature import VectorAssembler, StringIndexer, StandardScaler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.types import DoubleType

spark = SparkSession.builder \
    .appName("ComplexSalesAnalysis") \
    .master("local[*]") \
    .config("spark.driver.memory", "2g") \
    .config("spark.executor.memory", "2g") \
    .getOrCreate()
```

✓ 0.0s Python

2. 加载数据集，打印行数和数据结构

```python
# 2. 加载数据集

sales_df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)
print(f"数据集行数: {sales_df.count()}")

# 显示数据结构
sales_df.printSchema()
```

✓ 0.4s Python

```
数据集行数: 113036
root
 |-- Date: string (nullable = true)
 |-- Day: integer (nullable = true)
 |-- Month: string (nullable = true)
 |-- Year: integer (nullable = true)
 |-- Customer_Age: integer (nullable = true)
 |-- Age_Group: string (nullable = true)
 |-- Customer_Gender: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- State: string (nullable = true)
 |-- Product_Category: string (nullable = true)
 |-- Sub_Category: string (nullable = true)
 |-- Product: string (nullable = true)
 |-- Order_Quantity: integer (nullable = true)
 |-- Unit_Cost: integer (nullable = true)
 |-- Unit_Price: integer (nullable = true)
 |-- Profit: integer (nullable = true)
 |-- Cost: integer (nullable = true)
 |-- Revenue: integer (nullable = true)
```

3. 创建多个表并连接

```python
# 3. 创建多个表并连接

# 创建产品信息表
print("创建产品信息表...")
product_info = sales_df.select(
    "Product",
    "Product_Category",
    "Sub_Category",
    "Unit_Cost",
    "Unit_Price"
).distinct()
print(f"产品数量: {product_info.count()}")

# 创建客户信息表
print("创建客户信息表...")
customer_info = sales_df.select(
    "Customer_Age",
    "Age_Group",
    "Customer_Gender",
    "Country",
    "State"
).distinct()
print(f"客户数量: {customer_info.count()}")

# 创建订单信息表
print("创建订单信息表...")
order_info = sales_df.select(
    "Date",
    "Product",
```

```
创建产品信息表...
产品数量: 138
创建客户信息表...
客户数量: 2162
创建订单信息表...
订单数量: 113036

连接产品信息和订单信息...
连接后数据行数: 139696
+--------------------+---------+--------------+------+-------+----------------+------------+--------
|             Product|     Date|Order_Quantity|Profit|Revenue|Product_Category|Sub_Category|Unit_Cos
+--------------------+---------+--------------+------+-------+----------------+------------+--------
|Fender Set - Moun...|2016/2/20|             1|     9|     17|        Clothing|      Gloves|
|Fender Set - Moun...|2014/2/20|             2|    18|     34|        Clothing|      Gloves|
|Fender Set - Moun...|2015/11/1|            16|   147|    275|        Clothing|      Gloves|
|Fender Set - Moun...|2013/11/1|            18|   165|    309|        Clothing|      Gloves|
|Fender Set - Moun...|2015/8/31|            16|   157|    285|        Clothing|      Gloves|
+--------------------+---------+--------------+------+-------+----------------+------------+--------
only showing top 5 rows

连接三个表...
完全连接后数据行数: 113036
+----------+------------------+------------+---------------+-------+
|      Date|           Product|Customer_Age|Customer_Gender|Revenue|
+----------+------------------+------------+---------------+-------+
|2013/11/26|Hitch Rack - 4-Bike|          19|              M|    950|
|2015/11/26|Hitch Rack - 4-Bike|          19|              M|    950|
| 2014/3/23|Hitch Rack - 4-Bike|          49|              M|   2401|
```

4. 使用 Spark SQL，执行多个复杂查询

（1）查询每个产品类别的总收入和平均利润

（2）查询每个国家的销售情况

（3）查询每月销售趋势

（4）使用窗口函数，查询每个类别的前 3 名产品

（5）复杂条件查询：识别高价值客户

```python
# 4. 使用Spark SQL执行复杂查询

# 注册临时表
sales_df.createOrReplaceTempView("sales")
product_info.createOrReplaceTempView("products")
customer_info.createOrReplaceTempView("customers")

# 查询每个产品类别的总收入和平均利润
print("每个产品类别的总收入和平均利润:")
category_stats = spark.sql("""
    SELECT
        Product_Category,
        COUNT(*) as Order_Count,
        SUM(Revenue) as Total_Revenue,
        AVG(Profit) as Avg_Profit,
        SUM(Profit) as Total_Profit,
        ROUND((SUM(Profit) / SUM(Revenue)) * 100, 2) as Profit_Margin_Percent
    FROM sales
    GROUP BY Product_Category
    ORDER BY Total_Revenue DESC
""")
category_stats.show()
```

输出如下。

每个产品类别的总收入和平均利润：

Product_Category	Order_Count	Total_Revenue	Avg_Profit	Total_Profit	Profit_Margin_Percent
Bikes	25982	61782134	789.7496728504349	20519276	33.21
Accessories	70120	15117992	126.38871933827724	8862377	58.62
Clothing	16934	8370882	167.67727648517774	2839447	33.92

每个国家的销售情况：

Country	Order_Count	Total_Revenue	Total_Profit	Avg_Customer_Age
United States	39206	27975547	11073644	37.39111360506045
Australia	23936	21302059	6776030	34.38394050802139
United Kingdom	13620	10646196	4413853	35.55183553597651
Germany	11098	8978596	3359995	34.868084339520635
France	10998	8432872	2880282	35.11693035097291
Canada	14178	7935738	3717296	36.238961771759065

每月销售趋势：

Year	Month	Order_Count	Monthly_Revenue	Monthly_Profit	Avg_Order_Quantity
2011	January	188	675193	212849	1.925531914893617
2011	February	171	637598	207144	2.0292397660818713
2011	March	199	708517	226404	1.9949748743718594

每个产品类别的前3名畅销产品：

Product_Category	Product	Total_Revenue	Revenue_Rank
Accessories	Sport-100 Helmet,...	2019021	1
Accessories	Sport-100 Helmet,...	1948695	2
Accessories	Sport-100 Helmet,...	1775081	3
Bikes	Road-150 Red, 62	3829416	1
Bikes	Mountain-200 Blac...	3366248	2
Bikes	Road-150 Red, 52	3180840	3
Clothing	Women's Mountain ...	663155	1
Clothing	Long-Sleeve Logo ...	643544	2
Clothing	Women's Mountain ...	596287	3

高价值客户识别：

Customer_Age	Age_Group	Customer_Gender	Country	Order_Count	Total_Spent	Avg_Order_Value	Max_Order_Value
34	Young Adults (25-34)	F	United States	650	711179	1094.1215384615384	9577
35	Adults (35-64)	M	United States	694	691849	996.900576368876	11163
31	Young Adults (25-34)	M	United States	742	688585	928.0121293800539	14026
43	Adults (35-64)	M	United States	606	666237	1099.40099009901	14026
32	Young Adults (25-34)	F	United States	706	648301	918.2733711048159	9094
30	Young Adults (25-34)	M	United States	590	623449	1056.693220338983	14026
39	Adults (35-64)	F	United States	582	598359	1028.1082474226805	14026
28	Young Adults (25-34)	M	United States	564	577322	1023.6205673758865	14026
31	Young Adults (25-34)	F	United States	664	570412	859.0542168674699	9231
34	Young Adults (25-34)	M	United States	780	568659	729.05	14026

5. 使用 DataFrame API 进行复杂转换

（1）添加 Profit_Margin，Unit_Profit，Customer_Segment，Revenue_Category 四个派生列

（2）计算每个产品的移动平均收入

（3）数据透视：按产品类别和客户性别统计收入

```
print("添加派生列:")
enriched_df = sales_df.withColumn(
    "Profit_Margin",
    expr("ROUND((Profit / Revenue) * 100, 2)")
).withColumn(
    "Unit_Profit",
    expr("Unit_Price - Unit_Cost")
).withColumn(
    "Customer_Segment",
    when(col("Customer_Age") < 25, "Youth")
    .when((col("Customer_Age") >= 25) & (col("Customer_Age") < 40), "Young_Adult")
    .when((col("Customer_Age") >= 40) & (col("Customer_Age") < 60), "Middle_Aged")
    .otherwise("Senior")
).withColumn(
    "Revenue_Category",
    when(col("Revenue") < 500, "Low")
    .when((col("Revenue") >= 500) & (col("Revenue") < 1000), "Medium")
    .otherwise("High")
)
enriched_df.select("Product", "Customer_Age", "Customer_Segment", "Revenue", "Revenue_Category", "Profit_Margin").show(5)

# 使用窗口函数计算移动平均
print("计算每个产品的移动平均收入:")
window_spec = Window.partitionBy("Product").orderBy("Date").rowsBetween(-2, 0)
moving_avg_df = enriched_df.withColumn(
```

输出如下。

```
添加派生列:
+------------------+------------+----------------+-------+----------------+-------------+
|           Product|Customer_Age|Customer_Segment|Revenue|Revenue_Category|Profit_Margin|
+------------------+------------+----------------+-------+----------------+-------------+
|Hitch Rack - 4-Bike|          19|           Youth|    950|          Medium|        62.11|
|Hitch Rack - 4-Bike|          19|           Youth|    950|          Medium|        62.11|
|Hitch Rack - 4-Bike|          49|     Middle_Aged|   2401|            High|        56.89|
|Hitch Rack - 4-Bike|          49|     Middle_Aged|   2088|            High|         56.9|
|Hitch Rack - 4-Bike|          47|     Middle_Aged|    418|             Low|        56.94|
+------------------+------------+----------------+-------+----------------+-------------+
only showing top 5 rows
计算每个产品的移动平均收入:
+---------+------------+-------+------------------+
|     Date|     Product|Revenue|Moving_Avg_Revenue|
+---------+------------+-------+------------------+
|2013/10/1|AWC Logo Cap|     88|              88.0|
|2013/10/1|AWC Logo Cap|     84|              86.0|
|2013/10/1|AWC Logo Cap|    102| 91.33333333333333|
|2013/10/1|AWC Logo Cap|     59| 81.66666666666667|
|2013/10/1|AWC Logo Cap|    121|              94.0|
+---------+------------+-------+------------------+
only showing top 5 rows
数据透视：按产品类别和客户性别统计收入:
+----------------+---------------+-------------+---------------+-------------+
|Product_Category|F_Total_Revenue|F_Order_Count|M_Total_Revenue|M_Order_Count|
...
|        Clothing|        3860434|         8028|        4510448|         8906|
|     Accessories|        7092647|        33844|        8025345|        36276|
+----------------+---------------+-------------+---------------+-------------+
```

6. 使用 MLlib 进行机器学习任务

我们选择的任务目标为预测收入类别（Revenue_Category）。

这是一个三分类任务，基于产品、客户、订单等特征预测订单收入水平（Low/Medium/High）。

因为 Revenue_Category 基于 Revenue 分箱，Revenue = Order_Quantity × Unit_Price，所以选择特征时需要去掉选择 Order_Quantity、Unit_Price、Unit_Cost，避免数据泄露。

使用随机森林分类器。

```
# 特征标准化
print("特征标准化...")
scaler = StandardScaler(
    inputCol="features_raw",
    outputCol="features",
    withStd=True,
    withMean=True
)

scaler_model = scaler.fit(ml_data)
ml_data = scaler_model.transform(ml_data)

# 拆分训练集和测试集
(train_data, test_data) = ml_data.randomSplit([0.7, 0.3], seed=42)

# 使用随机森林分类器
print("训练随机森林分类模型...")
rf = RandomForestClassifier(
    featuresCol="features",
    labelCol="label_index",
    numTrees=100,
    maxDepth=10,
    seed=42
)

rf_model = rf.fit(train_data)
```

模型指标如下，准确率为 0.7952，F1 分数为 0.7337，对于本任务可以接受。
Medium 类别的分类准确率较差，因为数据集中 Medium 类型的数量较少，随机森林模型未充分学习特征。

```
模型性能评估:
  准确率（Accuracy）: 0.7952
  F1分数: 0.7337

各类别预测统计:
+------+------------------+-----+
| label|predicted_category|count|
+------+------------------+-----+
|  High|              High| 5866|
|  High|               Low| 1057|
|  High|            Medium|   57|
|   Low|              High|  347|
|   Low|               Low|20755|
|   Low|            Medium|  135|
|Medium|              High| 1532|
|Medium|               Low| 3774|
|Medium|            Medium|  182|
+------+------------------+-----+
```

最重要的特征为 Product_Category 和 Sub_Catagory，其他特征（客户、地理、时间）贡献很小。

```
特征重要性（前10名）:
   1. Product_Category_index        : 0.4502
   2. Sub_Category_index            : 0.4348
   3. Year                          : 0.0632
   4. Customer_Age                  : 0.0169
   5. State_index                   : 0.0130
   6. Month_index                   : 0.0083
   7. Country_index                 : 0.0070
   8. Age_Group_index               : 0.0034
   9. Customer_Gender_index         : 0.0032
```

结论分析与体会：

结论分析

本实验完成了 Spark 的安装配置、数据处理分析和机器学习任务。

通过 Spark SQL 和 DataFrame API 实现了复杂查询和数据转换，验证了 Spark 的数据处理的能力。在机器学习任务中，我们使用 MLlib 预测收入类别，准确率为 79.52%，验证 Spark 在机器学习场景下的应用。我们在实验过程中识别并避免了数据泄露，确保了模型结果的可靠性。

体会

通过本次实验，我们深入理解了 Spark 的核心概念和实际应用。Spark 的分布式计算架构能高效处理大规模数据，SQL 接口便于复杂查询，DataFrame API 支持灵活的数据转换操作。

在环境配置和代码调试过程中，我们掌握了 Spark 的基本使用方法，提升了大数据处理能力。团队分工明确，共同解决了技术问题。

参考资料：

Apache Spark 官网：https://spark.apache.org/