电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2019091608028

2019091604015

2019091604016

成员姓名 卞加勉、石佳一、夏野

（实验） 课程名称 大数据分析与智能计算

实验时间 2021-11-21

实验教师 罗瑜

**电 子 科 技 大 学**

**实 验 报 告**

1. **实验名称：**

基于 Hadoop 和 Spark 的数据预测实践项目

1. **实验学时：**

4学时

1. **实验目的：**
2. 掌握使用 PySpark 来分析数据；
3. 掌握使用 Spark 进行机器学习。
4. **实验原理：**
5. Spark 是一种通用并行分布式大数据计算框架。它基于MapReduce算法的分布式计算，拥有MapReduce的所有的优点。它将操作过程的中间结果放于内存中，所以不需要读写HDFS，能更好的适用于数据挖掘和机器学习的迭代算法。
6. **实验内容：**

针对 DataExpo2009 数据集，通过 Python、PySpark 等工具，进行航空公司延误和取消分析。

1. **实验器材（设备、元器件）：**
2. 软件环境：Ubuntu 20.04、jdk-8u181-linux-x64.tar、Hadoop 2.7.3、spark-3.1.2-bin-hadoop2.7.tgz、Jupyter、Parallels Desktop
3. 系统环境：MacOS
4. **实验步骤：**

针对 DataExpo2009 数据集，通过 Python、PySpark 等工具，进行数据读取、数据处理及可视化展示，并进行以下分析：

1. 查看飞机延误时间最长的前 10 名航班；
2. 计算延误的和没有延误的航空公司的比例；
3. 分析一天中、一周中延误最严重的飞行时间；
4. 比较取消短途航班和长途航班的严重性；
5. 建立机器学习算法模型，预测未来航班取消情况。
6. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**
7. PySpark 的安装及测试
8. 确认 Spark 环境

首先确认 Spark 的环境是否配置完成，使用如下命令确认 Hadoop 的环境变量，结果如图 1.1 所示。

|  |
| --- |
| $ echo $SPARK \_HOME |

|  |
| --- |
| 图 1.1 确认 Hadoop 环境变量 |

1. 测试 PySpark

运行 PySpark，测试 PySpark 是否可以成功启动，结果如图 1.2 所示。

|  |
| --- |
| $ pyspark --master local[\*] |

|  |
| --- |
| 图 1.2 测试 PySpark |

1. Jupyter Notebook 安装及测试
2. 下载 Anaconda

使用如下命令安装 Anaconda，结果如图 2.1 所示。

|  |
| --- |
| $ bash Anaconda3-2021.11-Linux-x86\_64.sh -b |

打开配置文件 .bash\_profile，导入配置，然后使用命令使配置生效，如图 2.2 所示。

|  |
| --- |
| 图电脑萤幕的截图  描述已自动生成 2.1 安装 Anaconda |
| 图文本  描述已自动生成 2.2 配置文件 |

1. Jupyter 安装配置

向 .bash\_profile 配置文件中添加如下内容，并使之生效。

|  |
| --- |
| export PYSPARK\_DRIVER\_PYTHON=jupyter export PYSPARK\_DRIVER\_PYTHON\_OPTS='notebook' |

1. 在 Notebook 中使用 Spark

使用以下命令创建并切换 ipynotebook 工作目录，之后启动 PySpark 服务，结果如图 2.3 所示。

|  |
| --- |
| $ mkdir -p ~/pythonwork/ipynotebook  $ cd ~/pythonwork/ipynotebook |

|  |
| --- |
| 图文本  描述已自动生成 2.3 启动 PySpark 服务 |

尝试从浏览器启动 Notebook 界面，默认是以 Spark 的本地模式运行，如图 2.4 所示。

|  |
| --- |
| 图 2.4 Notebook启动 |

1. 实践
2. 导入数据

使用如下命令，首先将数据集 DelayedFlights.csv 保存在 hadoop 用户的根目录下，然后在 HDFS 中创建 /pyspark\_data 目录，将数据上传到 HDFS 中。查看 HDFS 中数据集是否上传成功，结果如图 3.1 所示。

|  |
| --- |
| $ cp '/home/parallels/Desktop/Parallels Shared Folders/Home/Desktop/DelayedFlights.csv' ~  $ hadoop fs -mkdir /pyspark\_data  $ hadoop fs -put ~/DelayedFlights.csv /pyspark\_data/  $ Hadoop fs -ls /pyspark\_data |

|  |
| --- |
| 文本  描述已自动生成图 3.1 上传数据 |

新建立文件夹 /df\_pyspark 用于保存代码，如图 3.2 所示。

|  |
| --- |
| 图电脑萤幕的截图  描述已自动生成 3.2 建立目录 |

1. 查看飞机延误时间最长的前 10 名航班

如代码 3.1 所示，首先使用 Spark 提供的 read.load() 方法加载数据集，结果如图 3.3 所示。

|  |
| --- |
| 图图形用户界面, 文本, 应用程序  描述已自动生成 3.3 加载数据集 |

根据题意，为了查看平均延误时间最长的航班，我们需要得到不同航班（Origin-Dest）的总延误时间（TotalDelay）。总延误时间是实际经过时间（ActualElapsedTime）与计划经过时间（CRSElapsedTime）的差值，计算公式如下：

之后，我们求得总延误时间的平均值，按照从大到小的顺序，即可求得延误时间最长的前 10 个航空公司，运行结果如图 3.4 所示。

|  |
| --- |
| 图图形用户界面, 文本, 应用程序  描述已自动生成 3.4延误时间最长的前 10 名航班 |

1. 计算延误的和没有延误的航空公司的比例

航班延误，即航班的实际到达时间（ArrTime）晚于预计到达时间（CRSArrTime)。到达延误时间（ArrDelay）计算公式为：

也就是说，如果到达延误时间，那么我们可以判定这架航班属于延误航班。

据此，我们首先根据航空公司（UniqueCarrier）值对数据进行分组，通过判断到达延误时间是否大于0，统计延误航班的数量。如果延误航班的数量大于0，那么这家航空公司属于延误的航空公司；反之，则不属于。通过计算，我们可以看到所有的航空公司都有延误记录，因此延误的航空公司比例为 100%。

为了得到更有用的信息，我们接下来计算每一家航空公司延误的比例。我们对分组的结果进行聚合，统计每家航空公司的航班数量以及延误的航班数量，计算得到延误比例，如图 3.5 所示。

|  |
| --- |
| 图图形用户界面, 文本, 应用程序  描述已自动生成 3.5 延误航班 |

1. 分析一天中、一周中延误最严重的飞行时间

首先，为了分析不同时段，我们将一天（24小时）划分为6个时间段，分别是：0-6、6-12、12-18、18-24；将一周划分为7天。我们按照每一航班的预计出发时间作为航班时间段划分的依据，认为如果到达延误时间，就认为该航班属于延误航班。

计算一天六个时间段的延误率，结果如图 3.6 所示。我们可以看到，一天中，0-6小时之间延误率最低，为 84% 左右；之后延误率逐渐上升，在 18-24小时达到最高，为 89% 左右。

|  |
| --- |
| （1） |
| 图形用户界面, 文本, 应用程序  描述已自动生成（2）  图 3.6 一天中延误率计算 |

对一周每一天的延误率进行计算，结果如图 3.7 所示。分析可以得到，一周之中，星期六的延误率最低，为 86% 左右；星期五的延误率最高，为 90% 左右。

|  |
| --- |
| 图形用户界面, 文本, 应用程序  描述已自动生成（1） |
| 图形用户界面, 应用程序  描述已自动生成（2）  图 3.7 一周延误率计算 |

1. 比较取消短途航班和长途航班的严重性

在民用航空界领域，使用航班的飞行时间长度作为距离长度的界定标准。我们认为飞行时间小于等于 180 分钟的航班为短途航班；大于180分钟的航班为长途航班。我们计算航班取消的比率，作为取消航班严重性的衡量标准，结果如图 3.8 所示。

|  |
| --- |
| 图图形用户界面, 文本, 应用程序  描述已自动生成 3.8 长短途航班取消严重性 |

1. 建立机器学习算法模型，预测未来航班取消情况
2. 数据预处理

首先要查看 Dataframe 中列的类型，可以使用 printSchema() 方法，它将以树格式打印模式，结果如图 3.9 所示。

|  |
| --- |
| 图图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成 3.9 打印树模式 |

针对航班是否取消的问题，我们认为其中的 Year、Month、DayofMonth、DayofWeek、CRSDepTime、CRSArrTime、UniqueCarrier\_index、DepDelay、Distance 特征与航班是否取消有较大的关系。

根据打印模式的结果、我们发现 UniqueCarrier 特征为 String 数据类型，因此需要转换数据类型，使其可以作为特征向量输入到模型之中。因此，我们将数据中的 Null 值用0填充，应用 StringIndexer 将类别特征转换为可计算的数值，再使用 OneHotEncoder 编码得到新的映射值；然后，使用 VectorAssembler 变换器，将之前选择的多个特征变为一个特征向量，数据处理后如图 3.10 所示。

|  |
| --- |
| 图 3.10 数据预处理 |

处理好数据后，我们将‘Cancelled’变量重命名为‘label’，作为标签值，如图 3.11 所示。

|  |
| --- |
| 图 3.11 提取特征向量 |

之后我们分割数据集，将 80% 的数据作为训练集，20% 的数据作为测试集，打印训练集和测试集的长度，结果如图 3.12 所示。

|  |
| --- |
| 图图形用户界面, 应用程序  描述已自动生成 3.12 查看数据集长度 |

1. 搭建模型

在这里我们选择搭建LogisticRegression、DecisionTreeClassifier、GBTClassifier 三种模型，对数据集进行训练。结果如图 3.13 所示。

|  |
| --- |
| 图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成（1）LogisticRegression |
| （2）DecisionTreeClassifier图形用户界面, 文本, 应用程序, 电子邮件  描述已自动生成 |
| 图形用户界面, 文本, 应用程序  描述已自动生成（3）GBTClassifier  图 3.13 模型训练结果 |

1. 评估模型

首先，使用 BinaryClassificationEvaluator 评估器对模型进行评估，结果如图 3.14 所示。

|  |
| --- |
| 图图形用户界面, 文本  描述已自动生成 3.14 评估模型 |

然后，我们使用准确率和召回率、ROC 曲线、AUC 值对模型进行评估。自定义 calculate\_metrics() 函数，将预测结果转换为 Pandas 中的 DataFrame 结构，调用 metric 包中的内置函数，打印评估结果，结果如图 3.15 所示。

|  |
| --- |
| 图图形用户界面, 文本, 应用程序  描述已自动生成 3.15 评估模型 |

1. 完整代码

完整代码见附件 delayFlight.html。

1. **总结及心得体会：**

PySpark 与我平常学习的机器学习有所不同，它具有支持两个机器学习库：mllib及ml，区别在于 ml 主要操作的是 DataFrame，而 mllib 操作的是 RDD，即二者面向的数据集不一样。相比于 mllib 在 RDD 提供的基础操作，ml 在DataFrame 上的抽象级别更高，数据和操作耦合度更低。

我在实验中所运用的主要是 ml 库，pyspark.ml 训练机器学习库主要包括：Transformer、Estimator、Pipeline。其中：

1. Transformer 实现了算法训练前的一系列的特征预处理工作，例如实验中所运用的 OneHot Encoder，主要对应操作为 transform；
2. Estimator对应各种机器学习算法，主要区分分类、回归、聚类和推荐算法4大类，主要对应操作为fit；
3. Pipeline是为了将一些列转换和训练过程形成流水线的容器，类似于RDD在转换过程中形成DAG的思路一致，分阶段调用transformer中的transform操作或estimator中的fit操作。

通过完整的运用上面的三个板块，我体会到了以下几点有关 PySpark 较传统机器学习不同之处：

1. 延迟执行：延迟执行是基于 DAG 实现，也是 Spark 实现运行效率优化的一大关键。即在中间转换过程时仅记录逻辑转换顺序，而直到遇有产出非结果时才真正执行，例如评估和预测等；
2. DataFrame增加列：DataFrame 是不可变对象，所以在实际各类transformer处理过程中，处理的逻辑是在输入对象的基础上增加新列的方式产生新对象，所以多数接口需指定 inputCol 和 outCol 参数；
3. 算法与模型：个人认为这是spark.ml中比较好的一个细节，即严格区分算法和模型的定义边界，而这在其他框架或大多数学习者的认知中是一个模糊的概念。在 Spark 中，算法是通常意义下的未经过训练的机器学习算法，例如逻辑回归算法、随机森林算法，由于未经过训练，所以这里的算法是通用的；而模型则是经过训练后产出的带有参数配置的算法，经过训练后可直接用于预测和生产。所以，从某种意义上讲，模型=算法+配套参数。在 Spark中，模型在相应算法命名基础上带有 Mode l后缀，例如 LinearSVC 和 LinearSVCModel，前者是算法，后者则是模型。