电子科技大学信息与软件工程学院

**项 目 报 告**

课程名称 大数据分析与智能计算

理论教师 汤羽

实验教师 杨珊

学生信息：

|  |  |  |
| --- | --- | --- |
| 序号 | 学号 | 姓名 |
| 1 | 2021090921019 | 杨径骁 |
| 2 | 2021090921020 | 谭宇乔 |
| 3 | 2021090914029 | 卢秋男 |
| 4 | 2021090922023 | 张轩豪 |
| 5 | 2021090923001 | 赵胤翰 |

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**项 目 报 告**

**指导教师：杨珊 地点：二教110**

1. **项目名称：航空公司****延误和取消分析项目**
2. **项目时间：2023.11.22—2023.12.22**
3. **项目原理**

读入航班数据并处理（可以存入HDFS、HBase等），制作图形和表格来进行数据分析，建立预测模型，预测航班取消情况，分析至少包括以下内容：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短途航班和长途航班，哪种航班取消更严重？⑤建立机器学习算法模型，预测未来航班取消情况。

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。该数据集包含1987年10月到2008年12月美国境内所有商业航班的航班到达和离开详细信息。这是一个大型数据集：总共有近1.2亿条记录，占用了1.6 GB的压缩空间和12 GB的未压缩时空间。

Python是数据分析最常用的语言之一，而Apache Spark是一个开源的强大的分布式查询和处理引擎。本实验要求基于Python语言进行Spark Application编程，完成数据获取、处理、数据分析及可视化方面常用的数据分析方法与技巧，让学生掌握使用PySpark来分析数据。

模型可选择：支持向量机（SVM）、随机森林（Random Forest）、梯度提升树(GBDT)、线性判别分析（Linear Discriminant Analysis）、伯努利贝叶斯分类（BernoulliNB）、Adaboost、XGBoost等。

对于机器学习模型的选择，可以根据问题的特性和数据集的大小来决定。若选择深度学习方法，LSTM模型在处理涉及时间序列的数据时通常效果较好。

在建模过程中，需要使用准确率、召回率、ROC曲线、AUC等指标来评估模型性能，以便选择最适合数据集和问题的模型。最终，通过综合分析，可以提取有价值的信息，为航空公司延误和取消提供有效的预测和分析。

# 项目内容

1. PySpark的安装及测试
2. Jupyter Notebook安装及测试
3. 针对数据集，进行相应的数据分析
   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. 分析一天中延误最严重的飞行时间
   4. 分析一周中延误最严重的飞行时间。
   5. 短途航班和长途航班，哪种航班取消更严重？
   6. 建立机器学习算法模型，预测未来航班取消情况。

# 五、 需求分析与设计

1. 背景

近年以来，受各种因素的影响，航班的正常率一直维持在一个比较低的水平，航班延误已经成为制约航空发展的一个短板。同时航班作为公共交通运输服务的一个重要成员，其为社会公共服务的质量，一定程度上损害了公众利益。尤其是出现航班大面积延误时，旅客与航空运输企业的纠纷就越来越多，矛盾越来越突出，严重时甚至激化为冲突。

1. 任务概述
   1. 目标

本项目需分析航空公司延误和取消情况，以帮助航空公司了解延误和取消的原因，并提供预测模型以预测未来的航班取消情况。并提供数据可视化和统计分析结果，以便决策者和相关人员能够更好地理解和利用数据。项目选取对DataExpo2009数据集，通过Python、PySpark等工具，进行航空公司延误和取消分析，并在此基础上制作图形和表格来进行数据分析，建立预测模型，预测航班取消情况。针对数据集的规模和复杂性，考虑使用分布式计算框架进行大数据处理和分析，以提高性能和效率。考虑数据集的增长和未来需求的变化，设计可扩展的架构和算法，以便在需要时能够处理更大规模的数据和更复杂的分析任务。通过数据获取和处理、数据分析和建模、数据可视化等步骤实现对航空公司延误和取消情况的分析，并提供预测模型以预测未来航班取消情况。

* 1. 用户特点

用户特点：范围较广，暂无客观规律性

1. 运行环境规定

本项目是基于VMware虚拟机的ubuntu-20.04.3系统，采用Anaconda开发工具完成。

1. 用例图

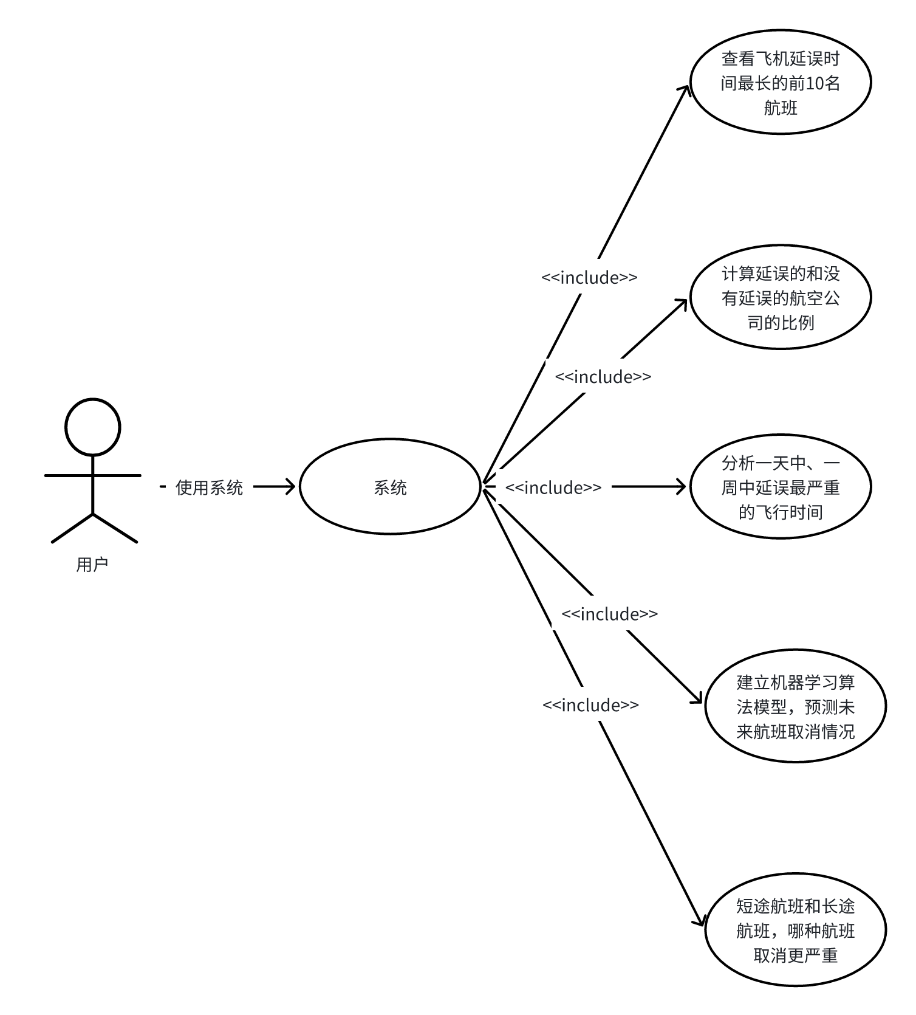


图1-1 系统用例图

|  |  |
| --- | --- |
| 要素 | 含义与要求 |
| 用例名称 | 查看飞机延误时间最长的前10名航班 |
| 简要描述 | 展示飞机延误时间最长的前10名航班的相关信息 |
| 参与者 | 用户、系统 |
| 前置流程 | 已经获取并处理了航班数据 |
| 基本流程 | * + - 1. 用户发出查看飞机延误时间最长的前10名航班的请求。       2. 系统根据航班延误时间对航班数据进行排序，选取延误时间最长的前10名航班。 |
| 备选流程 | 如果航班数据为空或不足10条：  系统向用户显示适当的消息，说明无法获取足够的航班数据进行展示。 |
| 后置条件 | 无 |

系统规约1-1 查找航班

|  |  |
| --- | --- |
| 要素 | 含义与要求 |
| 用例名称 | 计算延误和没有延误的航空公司的比例 |
| 简要描述 | 计算延误和没有延误的航空公司的比例，并展示结果。 |
| 参与者 | 用户、系统 |
| 前置流程 | 已经获取并处理了航班数据 |
| 基本流程 | 用户发出计算延误和没有延误的航空公司比例的请求。  系统统计延误的航班数量和没有延误的航班数量。  系统计算延误和没有延误的航空公司的比例。 |
| 备选流程 | 如果航班数据为空：  系统向用户显示适当的消息，说明无法计算延误和没有延误的航空公司比例。 |
| 后置条件 | 无 |

系统规约1-2 计算延误和没有延误航班

|  |  |
| --- | --- |
| 要素 | 含义与要求 |
| 用例名称 | 分析一天中、一周中延误最严重的飞行时间 |
| 简要描述 | 分析一天中和一周中延误最严重的飞行时间段，并展示结果。 |
| 参与者 | 用户、系统 |
| 前置流程 | 已经获取并处理了航班数据 |
| 基本流程 | * + - 1. 用户发出分析一天中和一周中延误最严重的飞行时间的请求。       2. 系统根据航班数据按照时间维度进行分析，计算一天中和一周中延误最严重的飞行时间段。 |
| 备选流程 | 如果航班数据为空：  系统向用户显示适当的消息，说明无法分析延误最严重的飞行时间。 |
| 后置条件 | 无 |

系统规约1-3 分析一天中、一周中延误最严重的飞行时间

|  |  |
| --- | --- |
| 要素 | 含义与要求 |
| 用例名称 | 短途航班和长途航班，哪种航班取消更严重 |
| 简要描述 | 比较短途航班和长途航班的取消情况，确定哪种航班的取消更严重，并展示结果。 |
| 参与者 | 用户、系统 |
| 前置流程 | 已经获取并处理了航班数据 |
| 基本流程 | 用户发出比较短途航班和长途航班取消情况的请求。  系统根据航班数据筛选出短途航班和长途航班，并统计它们的取消数量。  系统比较短途航班和长途航班的取消数量，确定哪种航班的取消更严重。 |
| 备选流程 | 如果航班数据为空：  系统向用户显示适当的消息，说明无法比较短途航班和长途航班的取消情况。 |
| 后置条件 | 无 |

系统规约1-4 短途航班和长途航班分析

|  |  |
| --- | --- |
| 要素 | 含义与要求 |
| 用例名称 | 建立机器学习算法模型，预测未来航班取消情况 |
| 简要描述 | 建立机器学习算法模型，使用历史航班数据预测未来航班的取消情况。 |
| 参与者 | 用户、系统 |
| 前置流程 | 已经获取并处理了航班数据 |
| 基本流程 | * + - 1. 用户发出建立机器学习算法模型预测航班取消情况的请求。       2. 系统根据历史航班数据，进行特征选择和模型训练。       3. 系统使用训练好的模型对未来航班数据进行预测，得出航班的取消概率。       4. 系统生成预测结果报告，并展示给用户。 |
| 备选流程 | 如果无法建立机器学习算法模型或模型训练失败：  系统向用户显示错误消息，并说明无法预测航班取消情况。 |
| 后置条件 | 无 |

系统规约1-5 建立机器学习算法模型

1. 数据流图

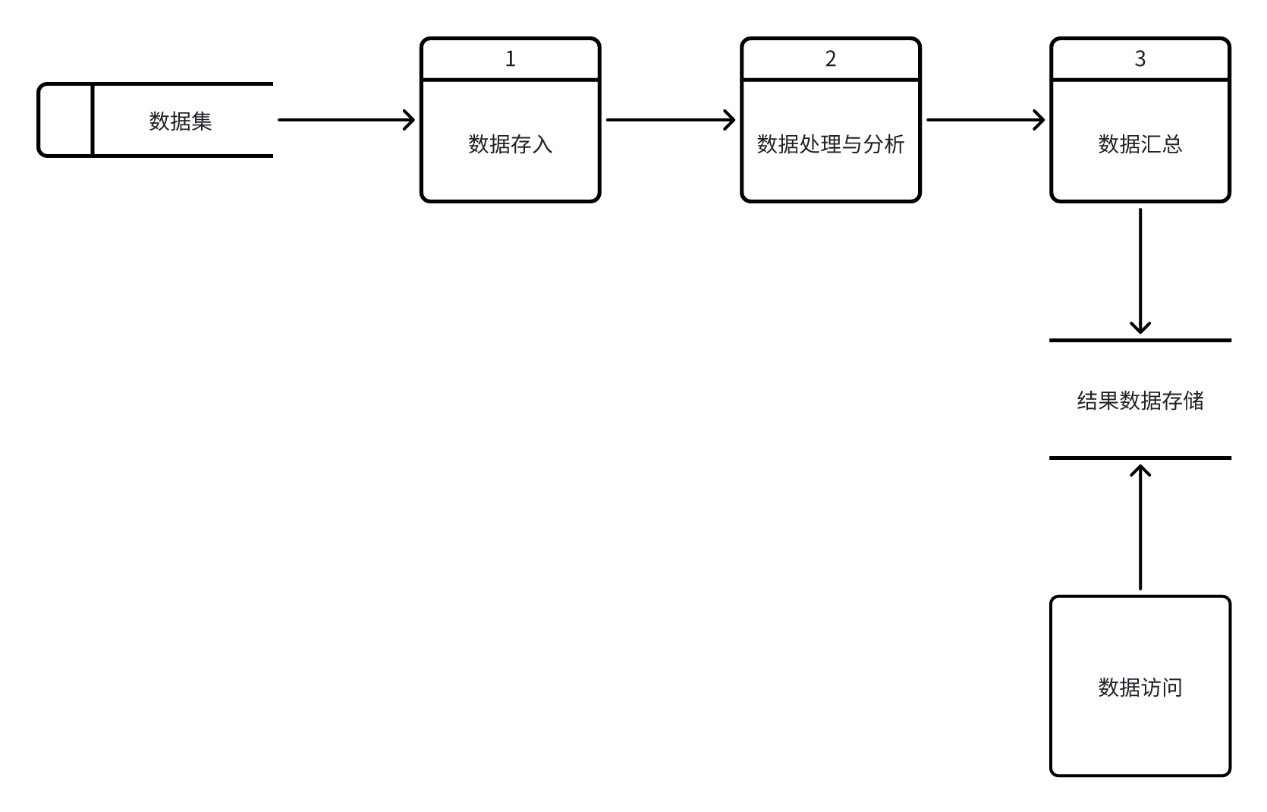
****

图1-2 数据流图

1. 顺序图

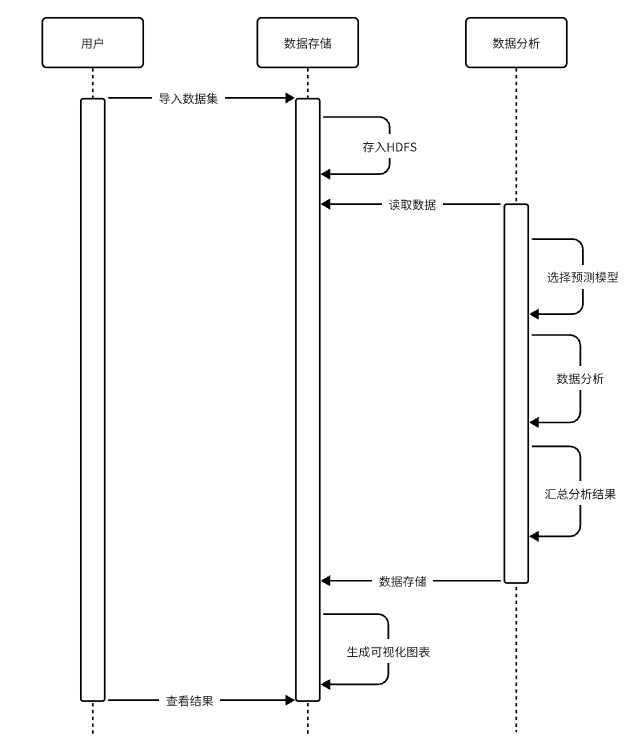


图1-3 顺序图

1. 流程图

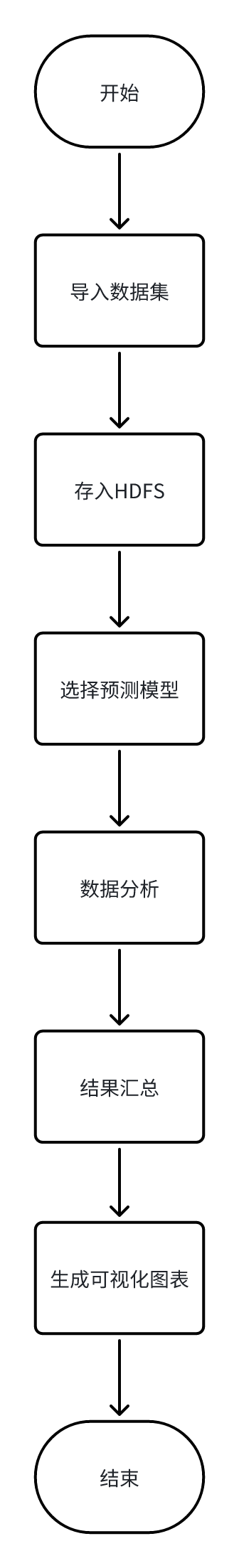
****

图1-4 流程图

# 项目计划

1. 进行需求分析，绘制用例图等详细描述需求分析
2. 环境配置，安装PySpark、Jupyter Notebook
3. 测试实验环境
4. 准备实验数据集
5. 针对数据集，进行相应的数据分析

# 七、项目环境配置管理

**7.1** **操作系统**：VMware虚拟机ubuntu-20.04.3系统针对数据集，进行相应的数据分析

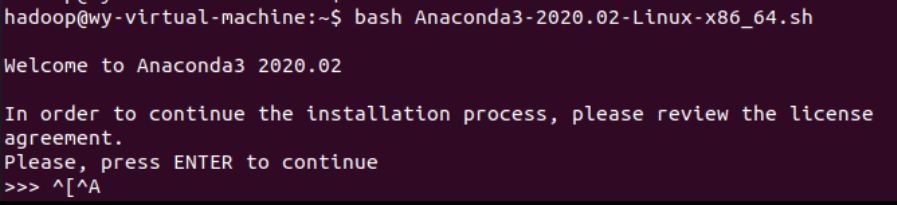
**7.2 开发工具**：Anaconda

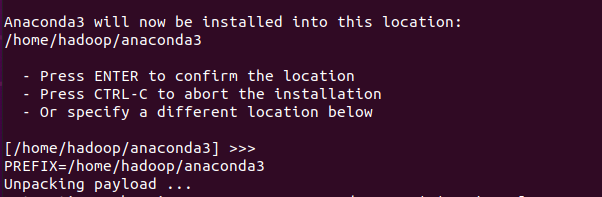
**7.3 配置过程**

1. **PySpark安装配置**
2. 下载Anaconda包

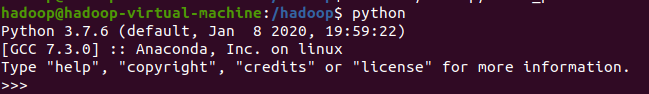


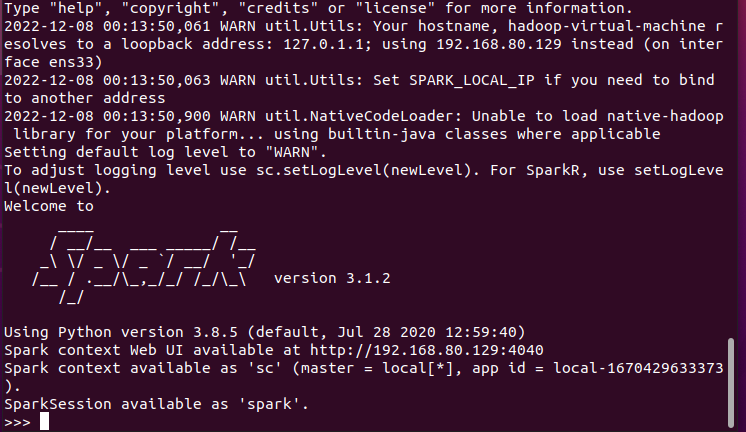
1. 安装Anaconda



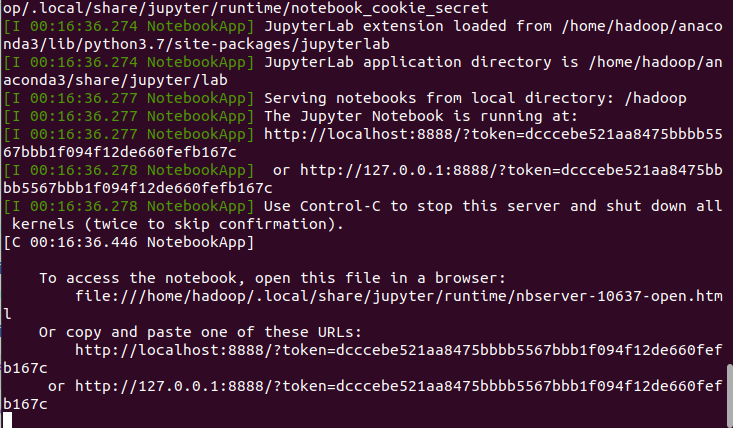


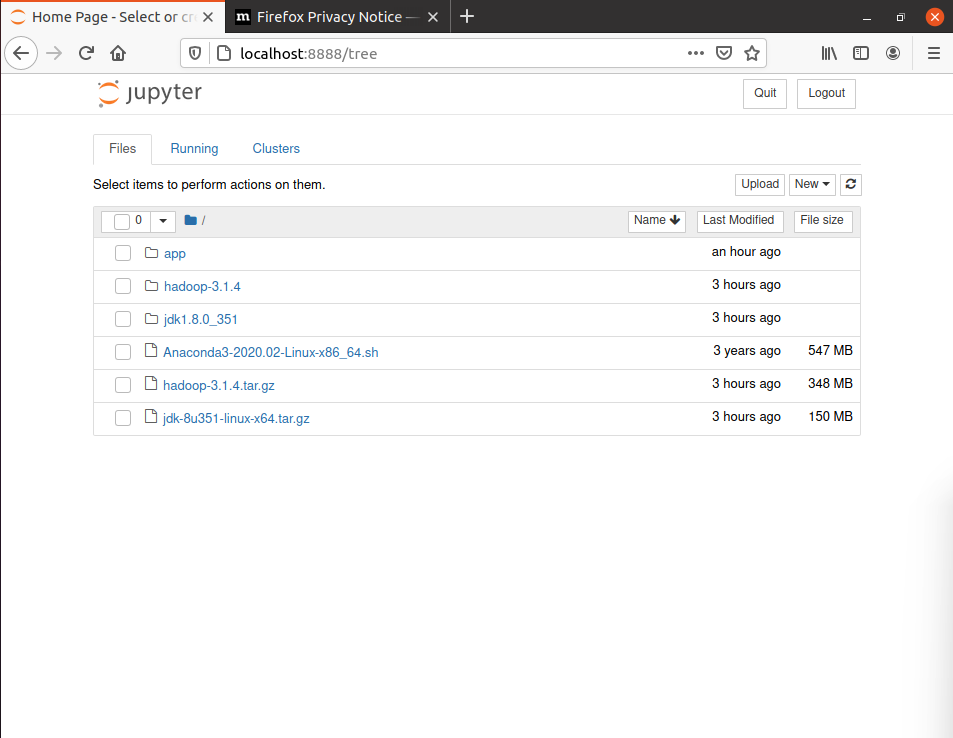
1. 验证python环境





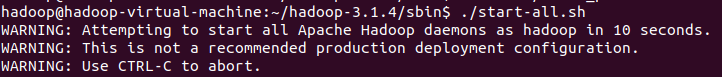
1. **Jupyter安装配置**
   1. 配置PySpark driver
   2. 启动Jupyter

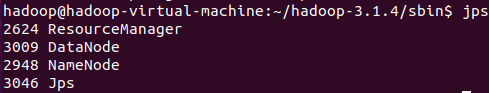




# 八、项目实践过程

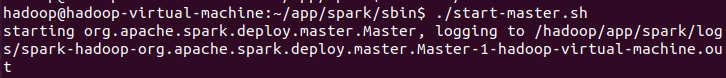
1. **准备过程**
2. 启动Hadoop





上图显示启动成功。

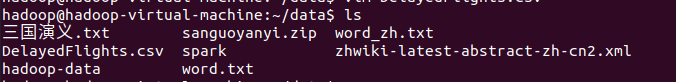
1. 启动Spark



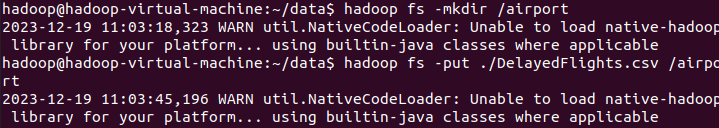
上图显示启动成功。

1. 上传文件

将数据集文件导入到hadoop用户中：



将数据集文件上传到hdfs系统中：



1. **问题一：“查看飞机延误时间最长的前10名航班”**

（1） 问题分析

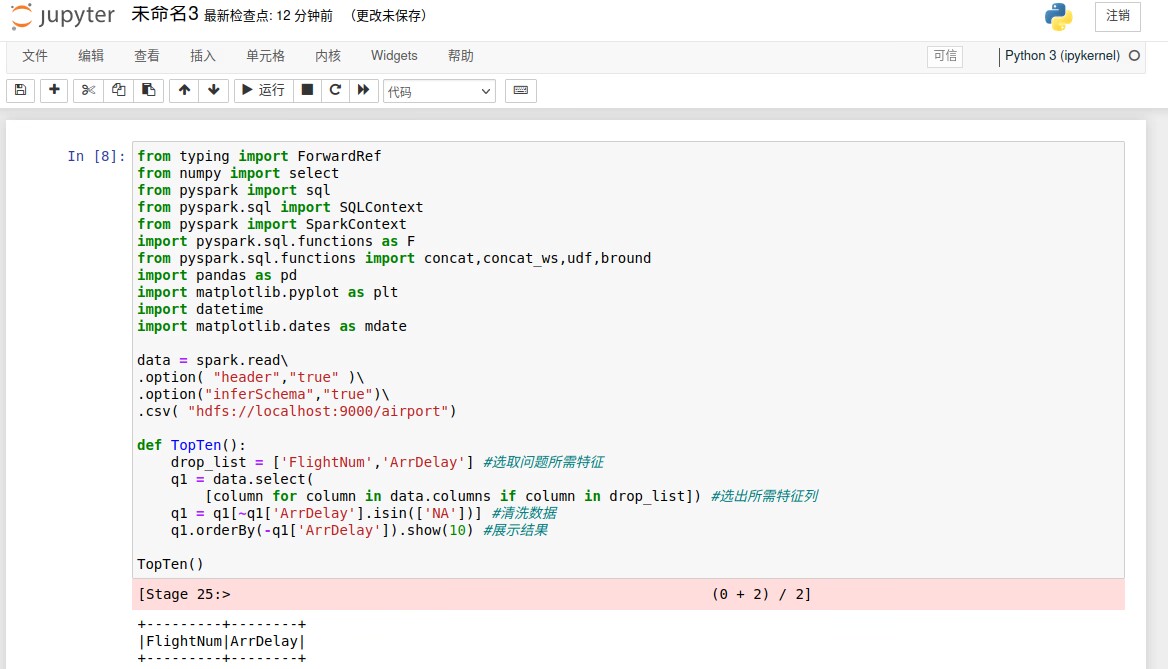
问题一需要“查看飞机延误时间最长的前10名航班”，经过讨论和分析，我们认为该问题应该按照抵达延误时间对航班进行排序，所以需要提取数据集中特征**航班号FlightNum、抵达延误时间ArrDelay，按照ArrDelay**大小进行排序。

（2）代码

代码 1

|  |
| --- |
| from typing import ForwardRef  from numpy import select  from pyspark import sql  from pyspark.sql import SQLContext  from pyspark import SparkContext  import pyspark.sql.functions as F  from pyspark.sql.functions import concat,concat\_ws,bround  import pandas as pd  import matplotlib.pyplot as plt  import datetime  import matplotlib.dates as mdate  data = spark.read\  .option("header","true")\  .option("inferSchema","true")\  .csv("hdfs://localhost:9000/airport")  def TopTen():  drop\_list = ['FlightNum','ArrDelay'] #选取问题所需特征  q1 = data.select(  [column for column in data.columns if column in drop\_list]) #选出所需特征列  q1 = q1[~q1['ArrDelay'].isin(['NA'])] #清洗数据  q1.orderBy(-q1['ArrDelay']).show(10) #展示结果  TopTen() |

运行页面如下：



1. **问题二：“计算延误的和没有延误的航空公司的比例”**
2. 问题分析

本问题为“计算延误的和没有延误的航空公司的比例”，在此问题中，建立特征为**航空公司编码UniqueCarrier**和**抵达的延误时间ArrDelay**，ArrDelay的作用是：根据判断ArrDelay是否大于零判断是否延误，继而分别进行统计、计算比例。

1. 代码

代码 2

|  |
| --- |
| from typing import ForwardRef  from numpy import select  from pyspark import sql  from pyspark.sql import SQLContext  from pyspark import SparkContext  import pyspark.sql.functions as F  from pyspark.sql.functions import concat,concat\_ws,bround  import pandas as pd  import matplotlib.pyplot as plt  import datetime  import numpy as np  data = spark.read\  .option("header","true")\  .option("inferSchema","true")\  .csv("hdfs://localhost:9000/airport/DelayedFlights.csv")  def rateOfDelay():  #建立特征  drop\_list\_2 = ['UniqueCarrier','ArrDelay']  q2 = data.select([column for column in data.columns  if column in drop\_list\_2])  q2 = q2[~q2['ArrDelay'].isin(['NA'])]  q2\_delay = q2.where(q2['ArrDelay'] > 0).groupBy('UniqueCarrier').count() #将延误时间大于零的航班分组，生成延误航班组  q2\_nodelay = q2.where(q2['ArrDelay'] <= 0).groupBy('UniqueCarrier').count() #将延误时间小于等于0的航班重新分组 ，生成未延误航班组  q2\_nodelay = q2\_nodelay.withColumnRenamed('count', 'NoDelayTimes') #重命名未延误航班列名  q2\_delay = q2\_delay.withColumnRenamed('count', 'DelayTimes') #重命名延误航班列名  q2\_merge = q2\_delay.join(q2\_nodelay, on='UniqueCarrier', how='left\_outer') #将延误航班与未延误航班列合并  q2\_merge = q2\_merge.withColumn('DelayRate', F.when(q2\_merge['NoDelayTimes'] != 0, q2\_merge['DelayTimes'] / q2\_merge['NoDelayTimes']).otherwise(1) )  return q2\_merge #返回新生成的数据集  def showTable():  q2\_merge=rateOfDelay()  q2\_merge.show()#展示结果    def reverseCapture():  q2\_merge=rateOfDelay()  pandas\_delay=q2\_merge.toPandas()# 转换为柱状图  pandas\_delay.DelayRate.plot(kind='barh')  plt.xlabel('DelayRate')  plt.ylabel('UniqueCarrier')  a=np.arange(len(pandas\_delay.UniqueCarrier))  plt.yticks(a,pandas\_delay.UniqueCarrier)#设置y轴  plt.xticks(rotation = 15)#设置x轴  plt.title("DelayRate")  plt.show()#展示柱状图  showTable()  reverseCapture() |

1. **问题三：分析一天中、一周中延误最严重的飞行时间”**

（1）问题分析

问题三要求我们分别“分析一天中、一周中延误最严重的飞行时间”，首先经过分析，我们理解题意为需要得到的数据结果为数据集内每一天、每一周中的最高延误时间，所以我们选取特征为年Year、月Month、月中的某日DayofMonth、星期几DayOfWeek、抵达延误时间ArrDelay，先算出每一天的最高延误时间，再据此算出每一周的最高延误时间。

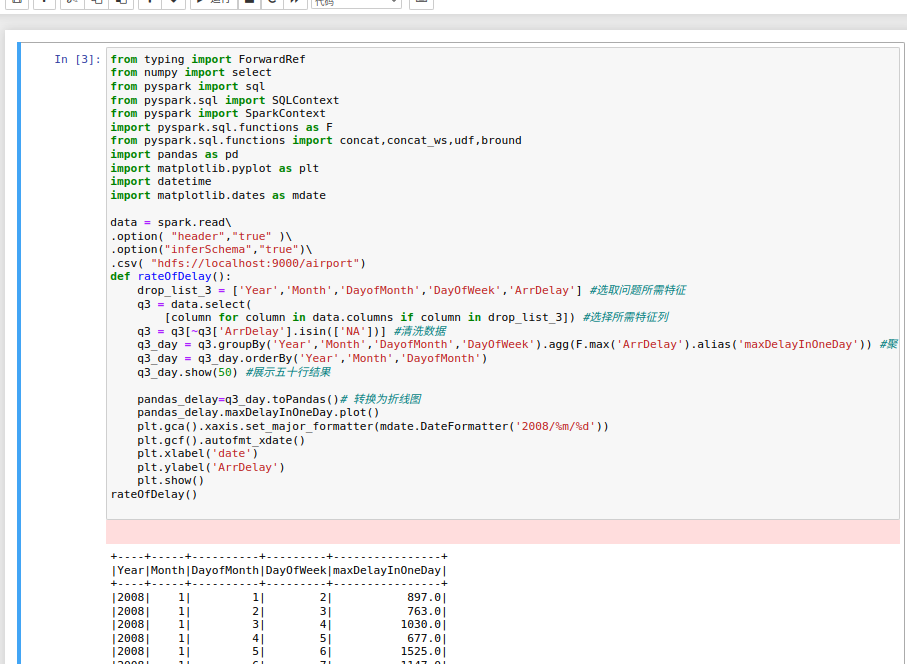
（2）代码

A.每一天的最高延误时间(只显示前50行)

代码 3

|  |
| --- |
| from typing import ForwardRef  from numpy import select  from pyspark import sql  from pyspark.sql import SQLContext  from pyspark import SparkContext  import pyspark.sql.functions as F  from pyspark.sql.functions import concat,concat\_ws,udf,bround  import pandas as pd  import matplotlib.pyplot as plt  import datetime  import matplotlib.dates as mdate  data = spark.read\  .option( "header","true" )\  .option("inferSchema","true")\  .csv( "hdfs://localhost:9000/airport")  def rateOfDelay():  drop\_list\_3 = ['Year','Month','DayofMonth','DayOfWeek','ArrDelay'] #选取问题所需特征  q3 = data.select(  [column for column in data.columns if column in drop\_list\_3]) #选择所需特征列  q3 = q3[~q3['ArrDelay'].isin(['NA'])] #清洗数据  q3\_day = q3.groupBy('Year','Month','DayofMonth','DayOfWeek').agg(F.max('ArrDelay').alias('maxDelayInOneDay')) #聚合每天的抵达延误时间进行比较，选出最大值加入“maxDelayInOneDay”列中  q3\_day = q3\_day.orderBy('Year','Month','DayofMonth')  q3\_day.show(50) #展示五十行结果    pandas\_delay=q3\_day.toPandas()# 转换为折线图  pandas\_delay.maxDelayInOneDay.plot()  plt.gca().xaxis.set\_major\_formatter(mdate.DateFormatter('2008/%m/%d'))  plt.gcf().autofmt\_xdate()  plt.xlabel('date')  plt.ylabel('ArrDelay')  plt.show()  rateOfDelay() |

运行界面如下：

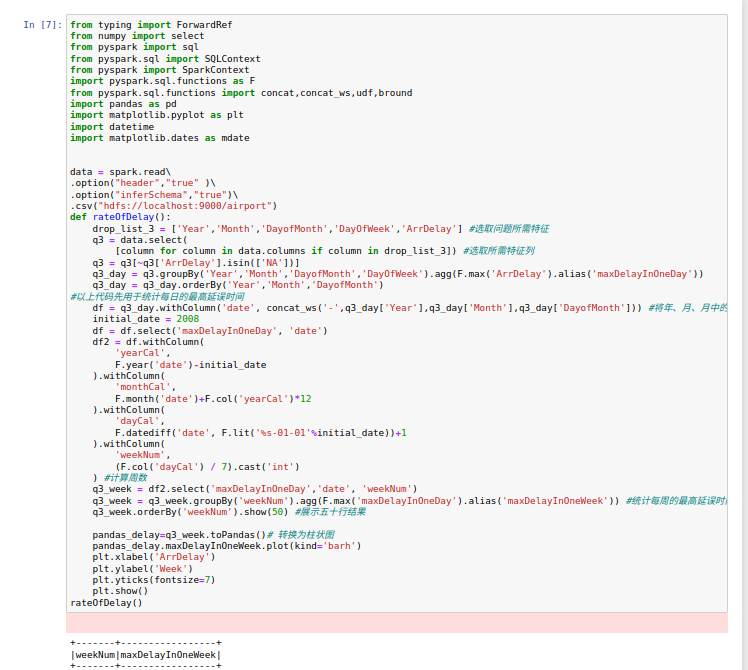


B. 每一周的最高延误时间(只显示前50行)

代码 4

|  |
| --- |
| from typing import ForwardRef  from numpy import select  from pyspark import sql  from pyspark.sql import SQLContext  from pyspark import SparkContext  import pyspark.sql.functions as F  from pyspark.sql.functions import concat,concat\_ws,udf,bround  import pandas as pd  import matplotlib.pyplot as plt  import datetime  import matplotlib.dates as mdate  data = spark.read\  .option("header","true" )\  .option("inferSchema","true")\  .csv("hdfs://localhost:9000/airport")  def rateOfDelay():  drop\_list\_3 = ['Year','Month','DayofMonth','DayOfWeek','ArrDelay'] #选取问题所需特征  q3 = data.select(  [column for column in data.columns if column in drop\_list\_3]) #选取所需特征列  q3 = q3[~q3['ArrDelay'].isin(['NA'])]  q3\_day = q3.groupBy('Year','Month','DayofMonth','DayOfWeek').agg(F.max('ArrDelay').alias('maxDelayInOneDay'))  q3\_day = q3\_day.orderBy('Year','Month','DayofMonth')  #以上代码先用于统计每日的最高延误时间  df = q3\_day.withColumn('date', concat\_ws('-',q3\_day['Year'],q3\_day['Month'],q3\_day['DayofMonth'])) #将年、月、月中的第几日用分隔符“-”连接，与date并列  initial\_date = 2008  df = df.select('maxDelayInOneDay', 'date')  df2 = df.withColumn(  'yearCal',  F.year('date')-initial\_date  ).withColumn(  'monthCal',  F.month('date')+F.col('yearCal')\*12  ).withColumn(  'dayCal',  F.datediff('date', F.lit('%s-01-01'%initial\_date))+1  ).withColumn(  'weekNum',  (F.col('dayCal') / 7).cast('int')  ) #计算周数  q3\_week = df2.select('maxDelayInOneDay','date', 'weekNum')  q3\_week = q3\_week.groupBy('weekNum').agg(F.max('maxDelayInOneDay').alias('maxDelayInOneWeek')) #统计每周的最高延误时间  q3\_week.orderBy('weekNum').show(50) #展示五十行结果    pandas\_delay=q3\_week.toPandas()# 转换为柱状图  pandas\_delay.maxDelayInOneWeek.plot(kind='barh')  plt.xlabel('ArrDelay')  plt.ylabel('Week')  plt.yticks(fontsize=7)  plt.show()  rateOfDelay() |

运行程序界面如下：



1. **问题四：“短途航班和长途航班，哪种航班取消更严重”**
2. 问题分析

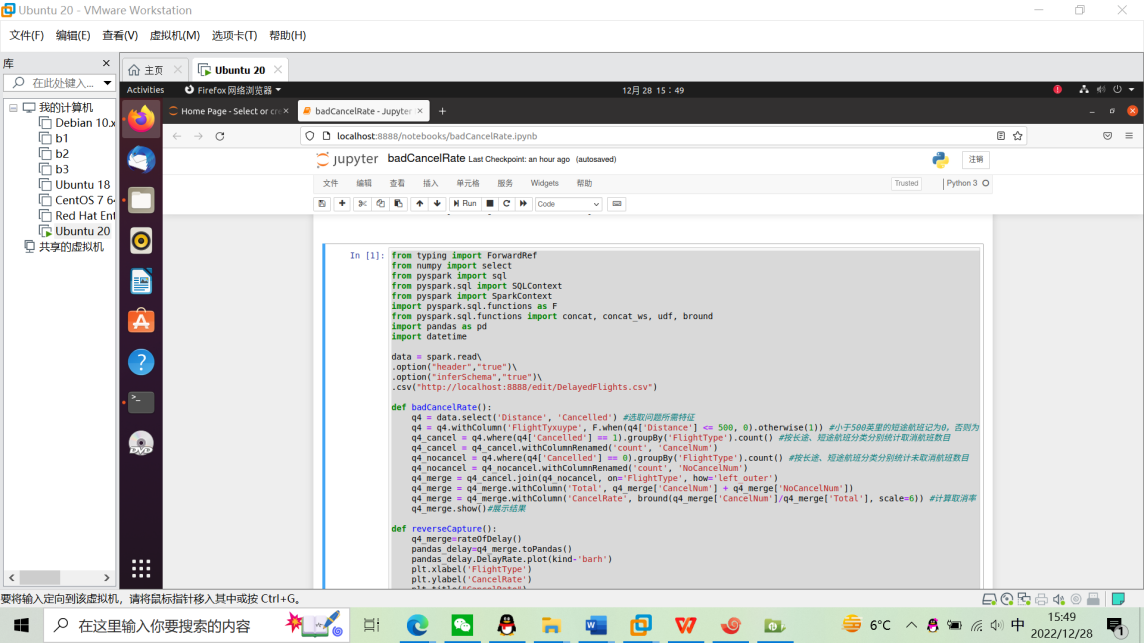
这一问要求判断“短途航班和长途航班，哪种航班取消更严重”，我们对该问题的分析为首先拟定短途和长途航班的距离分割界限为500英里，然后首先对所有航班进行短途和长途的分类，再分别同济两类中的是否取消情况，据此分析，我们需要的特征有**航行距离Distance、航班是否取消Cancelled**。

1. 代码

代码 5

|  |
| --- |
| from typing import ForwardRef  from numpy import select  from pyspark import sql  from pyspark.sql import SQLContext  from pyspark import SparkContext  import pyspark.sql.functions as F  from pyspark.sql.functions import concat, concat\_ws, udf, bround  import pandas as pd  import datetim0e  data = spark.read\  .option("header","true")\  .option("inferSchema","true")\  .csv("http://localhost:8888/edit/DelayedFlights.csv")  def badCancelRate():  q4 = data.select('Distance', 'Cancelled') #选取问题所需特征  q4 = q4.withColumn('FlightTyxuype', F.when(q4['Distance'] <= 500, 0).otherwise(1)) #小于500英里的短途航班记为0，否则为长途航班，记为1  q4\_cancel = q4.where(q4['Cancelled'] == 1).groupBy('FlightType').count() #按长途、短途航班分类分别统计取消航班数目  q4\_cancel = q4\_cancel.withColumnRenamed('count', 'CancelNum')  q4\_nocancel = q4.where(q4['Cancelled'] == 0).groupBy('FlightType').count() #按长途、短途航班分类分别统计未取消航班数目  q4\_nocancel = q4\_nocancel.withColumnRenamed('count', 'NoCancelNum')  q4\_merge = q4\_cancel.join(q4\_nocancel, on='FlightType', how='left\_outer')  q4\_merge = q4\_merge.withColumn('Total', q4\_merge['CancelNum'] + q4\_merge['NoCancelNum'])  q4\_merge = q4\_merge.withColumn('CancelRate', bround(q4\_merge['CancelNum']/q4\_merge['Total'], scale=6)) #计算取消率  q4\_merge.show()#展示结果    def reverseCapture():  q4\_merge=rateOfDelay()  pandas\_delay=q4\_merge.toPandas()  pandas\_delay.DelayRate.plot(kind-'barh')  plt.xlabel('FlightType')  plt.ylabel('CancelRate')  plt.title("CancelRate")  plt.show()# |

1. 运行界面



1. **问题五：建立机器学习算法模型，预测未来航班取消情况**

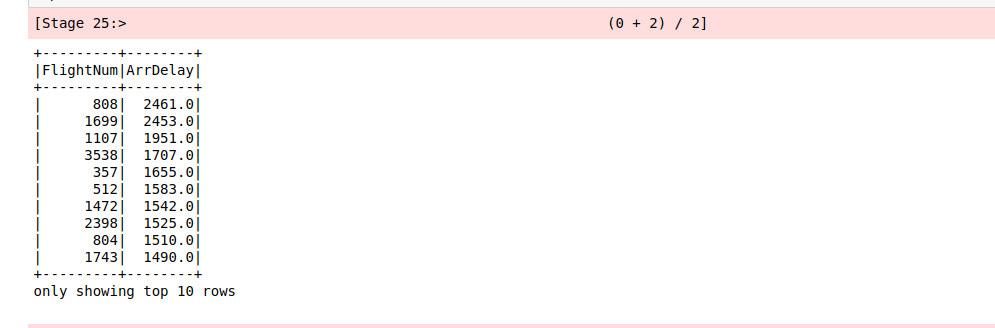
机器学习算法模型以预测未来航班取消情况的代码与运行界面如九-6所示。

**九、项目结果与分析（含重要数据结果分析或核心代码流程分析）**

**1. 查看飞机延误时间最长的前10名航班：**

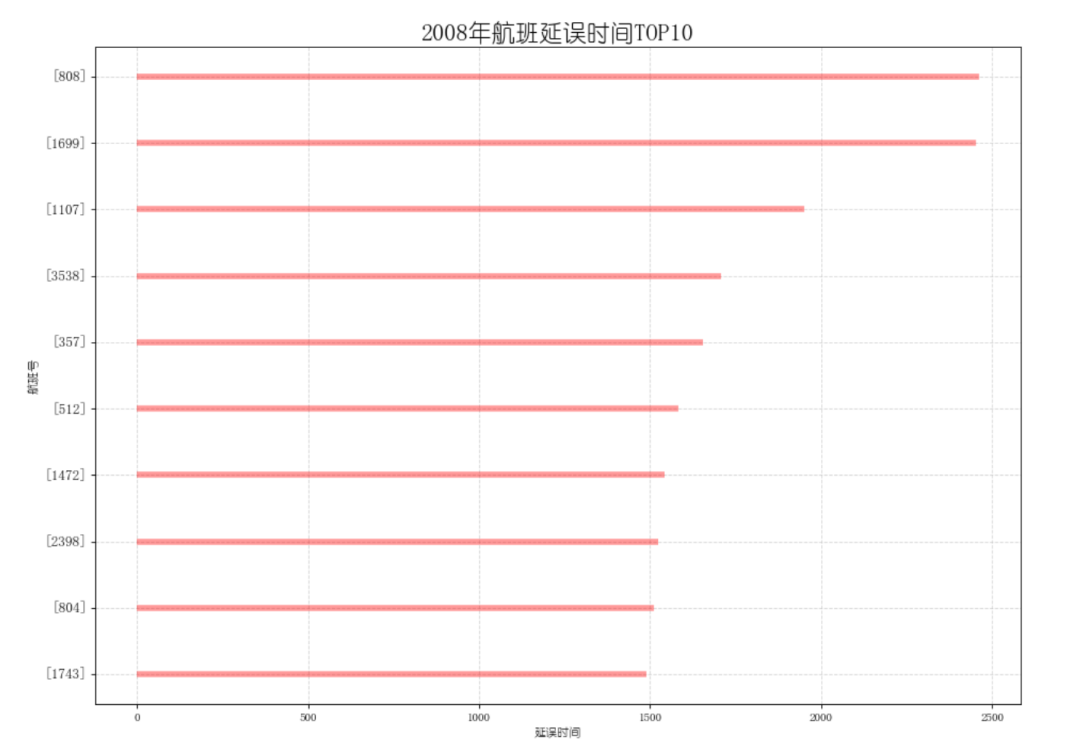
(1)重要数据结果分析

通过运行代码1，得到下图所示的结果。以下列表展示为数据集中前10名延误时间最长的航班，第一列为航班号，第二列为延误时间。



（2）柱状图表示

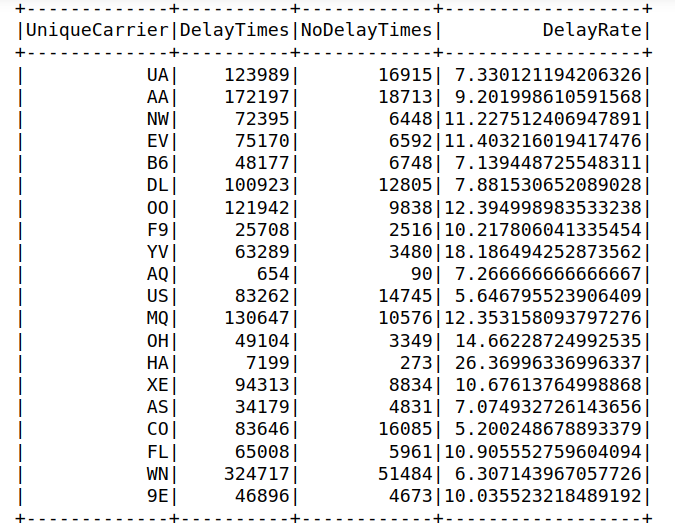
我们将上面的结果转换为可视化表格，将结果从pyspark下的dataframe结构转化为pandas下的dataframe，然后调用对pandas中对dataframe的画图接口plot()进行绘制。纵坐标为航班号，横坐标为延误时间，从上至下，可以一目了然地查看飞机延误时间最长的前10名航班，与上表符合，结果如下：



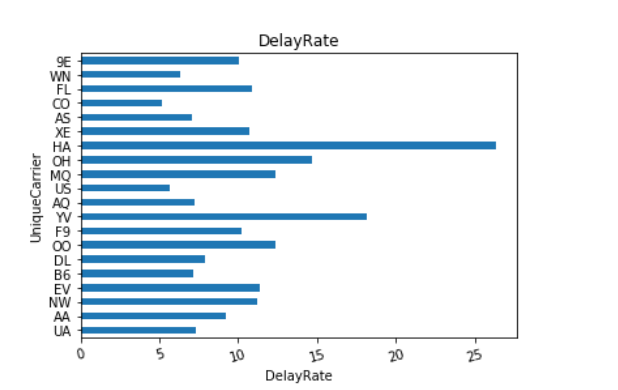
**2. 计算延误的和没有延误的航空公司的比例：**

(1)重要数据结果分析

通过运行代码二，得到下图所示的结果，展示了航空公司中延误与未延误的航空比例；第一列为航空公司编码，第二列为延误航班数，第三列为未延误航班数，最后一列未延误比例。通过结果可以看出HA航空公司延误比例最高，CO最低。



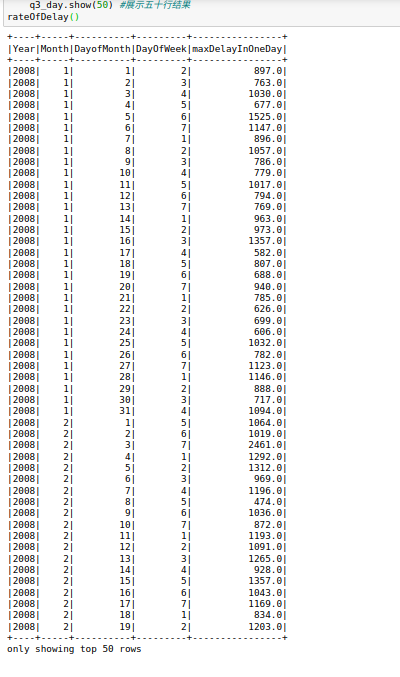
我们将上面的表格转换为柱状图，从柱状图中可以更明显地观察出HA的延误比例最高，CO最低，这个结果与上表相同。



**3. 分析****一天中延误最严重的飞行时间。**

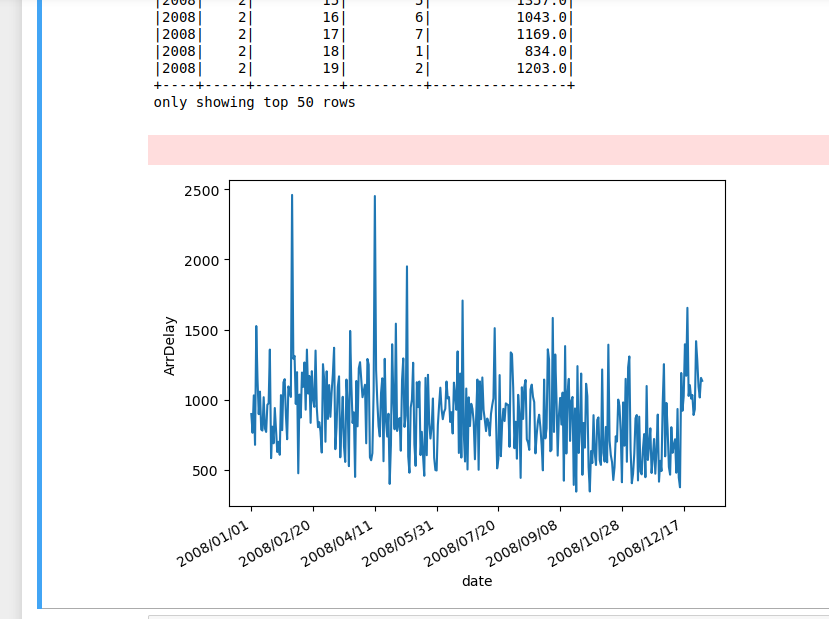
（1）重要数据结果分析

以下列表展示一年中每天最高的抵达延误时间，第一至四列分别为年、月、月中的第几日、星期几，第五列为该天最高的抵达延误时间。由于篇幅有限，此处仅展示部分结果（前50行）。



（2）折线图表示

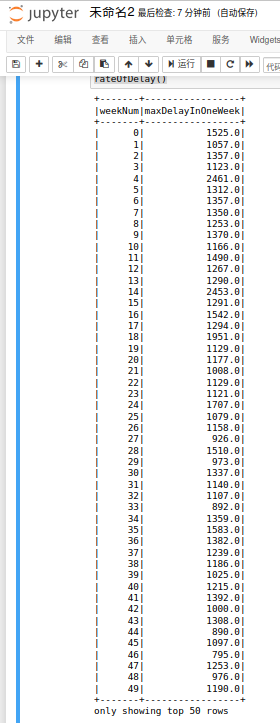
我们将上面的结果转换为可视化表格，调用toPandas()函数将pyspark下的dataframe结构转化为pandas下的dataframe，然后调用plot（）绘制折线图。该折线图用于每日延误的峰值，y轴代表的是延误时间，x轴代表的是日期，图形如下图所示：



**4. 分析一周中延误最严重的飞行时间。**

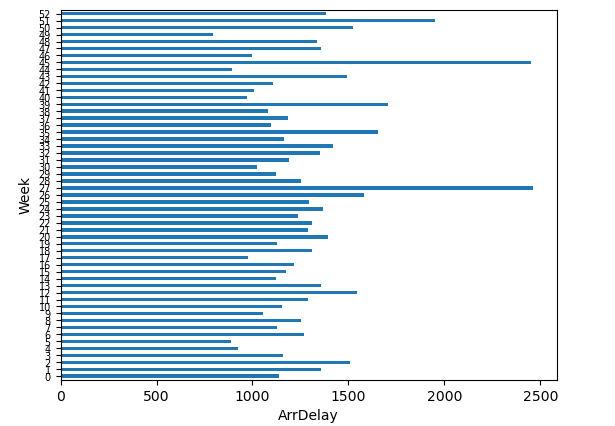
（1）重要数据结果分析

以下列表展示2008年中每一周中的最高抵达延误时间，第一列为周数，第二列为一周中的最高抵达延误时间。



（2）柱状图表示

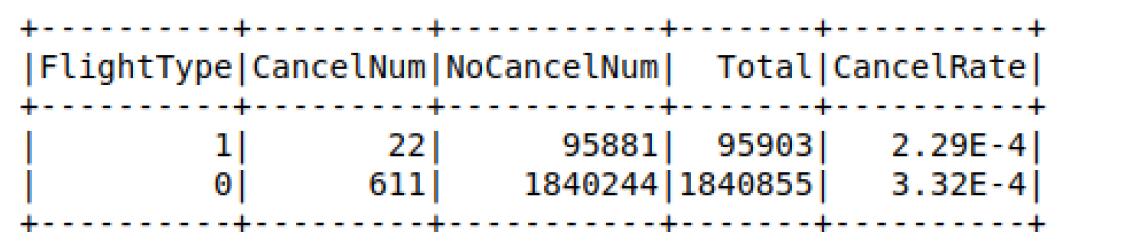
我们将上面的结果转换为可视化表格，调用toPandas()函数将pyspark下的dataframe结构转化为pandas下的dataframe，然后调用plot（kind='barh'）绘制柱状图。柱状图描述2008年每周航班延误峰值，y轴代表的是周数，x轴代表的是每周统计延误时间。柱状图如下图所示：



1. **短途航班和长途航班，哪种航班取消更严重？**

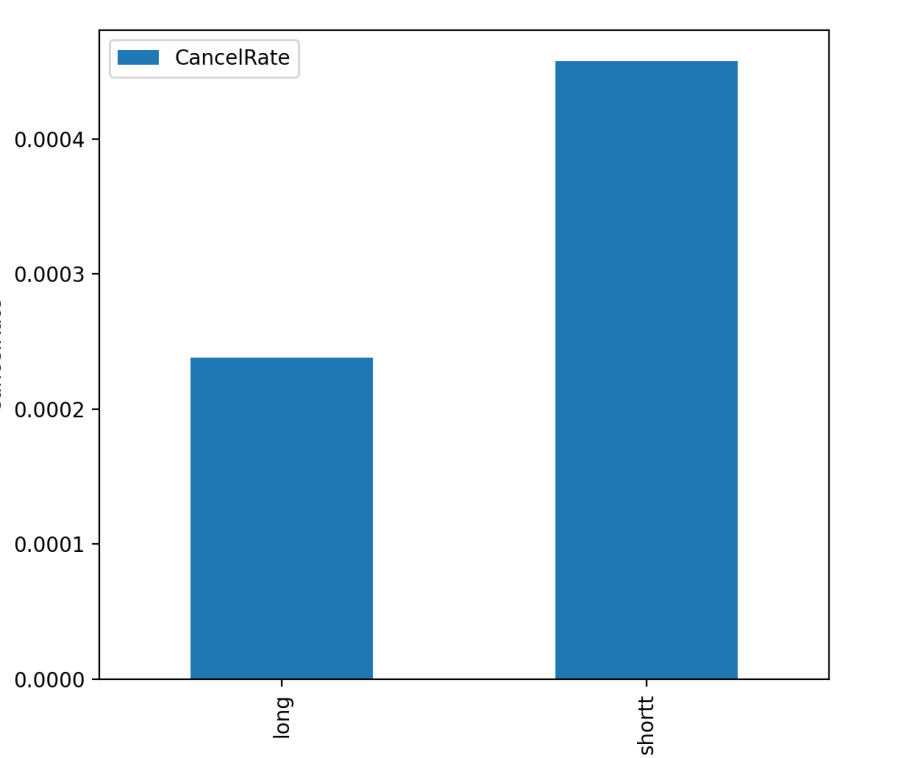
（1）重要数据结果分析

通过运行代码，可以得到以下结果。如图，第一列是航班类型，0为短途航班，1为长途航班，第二列为长途和短途航班分别对应的取消的数目，第三列为长途和短途航班对应没有取消的航班数目，第四列为短途和长途分别的航班总数，第五列为短途和长途航班分别的取消率，根据该结果能够明显看出，短途航班的取消率更高。



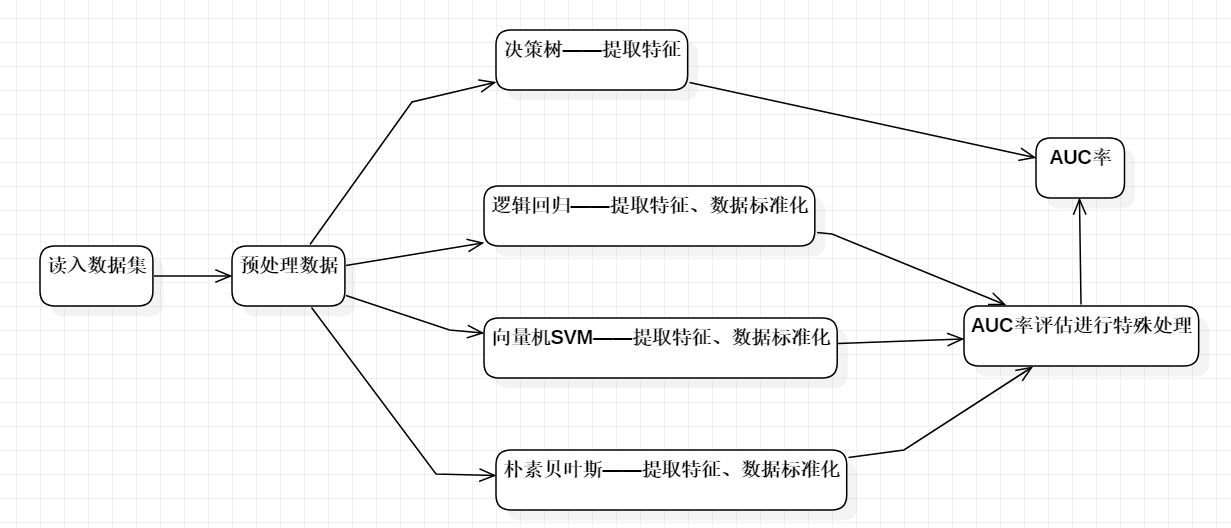
1. 柱状图表示

将结果转换成可视化表格，调用toPandas()函数将pyspark下dataframe结构转换为pandas下的dataframe，然后调用plot(kind=’barh’)绘制柱状图，x轴表示航班类型，y轴表示航班取消率，柱状图如下图所示：

****

1. **建立机器学习算法模型，预测未来航班取消情况。**
2. 核心代码流程分析

代码流程图：



数据预处理代码，提取特征操作，以决策树为例：

|  |
| --- |
| import numpy as np  from pyspark.mllib.regression import LabeledPoint  # 数据集路径  global Path  if sc.master[0:5] == "local":      Path = "file:/hadoop/pythonwork/PythonProject/"  else:      Path = "hdfs://master:9000/pythonwork/ex-data/"  print("Begin to get data")  # 去除数据集的头，也即数据集中的分类标题，同时以“,”分隔每项数据  rawDataWithHeader = sc.textFile(Path+"ex-data/DelayedFlights.csv")  header = rawDataWithHeader.first()  rawData = rawDataWithHeader.filter(lambda x:x != header)  lines = rawData.map(lambda x:x.split(","))  def extract\_category\_features(field, categoriesMap, num):      """      提取字符特征的函数      :param field: 每项数据      :param categoriesMap: 字典      :param num: 索引      :return:      """      categoryIdx = categoriesMap[field[num]]      categoryFeatures = np.zeros(len(categoriesMap))      categoryFeatures[categoryIdx] = 1      return categoryFeatures  def extract\_number\_features(field, num):      """      提取数字特征的函数      :param field: 每项数据      :param num: 索引      :return:      """      numericalFeatures=[float(field[num])]      return numericalFeatures  def merge\_features(field):      """      将所有特征整合到一起      :param field: 每项数据      :return:      """      UC\_features = extract\_category\_features(field, UC\_categoriesMap, 9)      O\_features = extract\_category\_features(field, O\_categoriesMap, 17)      D\_features = extract\_category\_features(field, D\_categoriesMap, 18)      Month\_feature = extract\_number\_features(field, 2)      DayOfMonth\_feature = extract\_number\_features(field, 3)      DayOfWeek\_feature = extract\_number\_features(field, 4)      FlightNum\_feature = extract\_number\_features(field, 10)      Diverted\_feature = extract\_number\_features(field, 24)      return np.concatenate((UC\_features,O\_features,D\_features, \                             Month\_feature,DayOfMonth\_feature,DayOfWeek\_feature, \                             FlightNum\_feature,Diverted\_feature))  # 9-UniqueCarrier 字典  UC\_categoriesMap = lines.map(lambda fields:fields[9]).distinct().zipWithIndex().collectAsMap()  # 17-Origin 字典  O\_categoriesMap = lines.map(lambda fields:fields[17]).distinct().zipWithIndex().collectAsMap()  # 18-Dest 字典  D\_categoriesMap = lines.map(lambda fields:fields[18]).distinct().zipWithIndex().collectAsMap()  def extract\_label(field):      """      提取标签 22-cancelled      :param field:      :return:      """      label = field[22]      return float(label)  # 创建 LabeledPoint 数据  labelpointRDD = lines.map(lambda r:                            LabeledPoint(extract\_label(r), merge\_features(r)))  # 以随机方式将数据以 8、1、1 分为 3 个部分  (trainData, validationData, testData) = labelpointRDD.randomSplit([8, 1, 1])  print("将数据分 trainData:" + str(trainData.count()) +        "   validationData:" + str(validationData.count()) +        "   testData:" + str(testData.count())) |

数据预处理代码，提取特征加标准化操作，以逻辑回归为例：

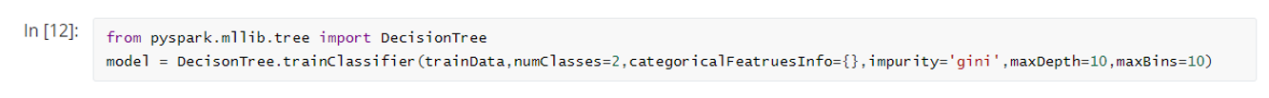
|  |
| --- |
| import sys  from time import time  import pandas as pd  import matplotlib.pyplot as plt  from pyspark import SparkConf, SparkContext  from pyspark.mllib.classification import LogisticRegressionWithSGD  from pyspark.mllib.regression import LabeledPoint  import numpy as np  from pyspark.mllib.evaluation import BinaryClassificationMetrics  from pyspark.mllib.feature import StandardScaler  def extract\_category\_features(field, categoriesMap, num):      categoryIdx = categoriesMap[field[num]]      categoryFeatures = np.zeros(len(categoriesMap))      categoryFeatures[categoryIdx] = 1      return categoryFeatures  def extract\_number\_features(field, num):      numericalFeatures=[float(field[num])]      return numericalFeatures  def merge\_features(field):      """      将所有特征整合到一起      """      UC\_features = extract\_category\_features(field, UC\_categoriesMap, 9)      O\_features = extract\_category\_features(field, O\_categoriesMap, 17)      D\_features = extract\_category\_features(field, D\_categoriesMap, 18)      Month\_feature = extract\_number\_features(field, 2)      DayOfMonth\_feature = extract\_number\_features(field, 3)      DayOfWeek\_feature = extract\_number\_features(field, 4)      FlightNum\_feature = extract\_number\_features(field, 10)      Diverted\_feature = extract\_number\_features(field, 24)      return np.concatenate((UC\_features,O\_features,D\_features,                             Month\_feature,DayOfMonth\_feature,DayOfWeek\_feature,                             FlightNum\_feature,Diverted\_feature))  def extract\_label(field):      """      提取标签-22-cancelled      :param field:      :return:      """      label = field[22]      return float(label)  # 数据集路径  global Path  if sc.master[0:5] == "local":      Path = "file:/hadoop/pythonwork/PythonProject/"  else:      Path = "hdfs://master:9000/pythonwork/ex-data/"  # 1.导入并转换数据  rawDataWithHeader = sc.textFile(Path+"ex-data/DelayedFlights.csv")  header = rawDataWithHeader.first()  rawData = rawDataWithHeader.filter(lambda x:x != header)  lines = rawData.map(lambda x:x.split(","))  # 2.建立训练评估所需数据 RDD[LabeledPoint]  # 9-UniqueCarrier 字典  UC\_categoriesMap = lines.map(lambda fields:fields[9]) \      .distinct().zipWithIndex().collectAsMap()  # 17-Origin 字典  O\_categoriesMap = lines.map(lambda fields:fields[17]) \      .distinct().zipWithIndex().collectAsMap()  # 18-Dest 字典  D\_categoriesMap = lines.map(lambda fields:fields[18]) \      .distinct().zipWithIndex().collectAsMap()  labelRDD = lines.map(lambda r: extract\_label(r))  featureRDD = lines.map(lambda r: merge\_features(r))  # 这里的 withMean 参数为 True，在朴素贝叶斯的模型立这个参数应当设置为 False  stdScaler = StandardScaler(withMean=True, withStd=True).fit(featureRDD)  ScalerFeatureRDD = stdScaler.transform(featureRDD)  labelpoint = labelRDD.zip(ScalerFeatureRDD)  labelpointRDD = labelpoint.map(lambda r: LabeledPoint(r[0], r[1]))  # 3.以随机方式将数据分为 3 个部分并且返回------------  (trainData, validationData, testData) = labelpointRDD.randomSplit([8, 1, 1])  print("将数据分 trainData:" + str(trainData.count()) +        "   validationData:" + str(validationData.count()) +        "   testData:" + str(testData.count())) |

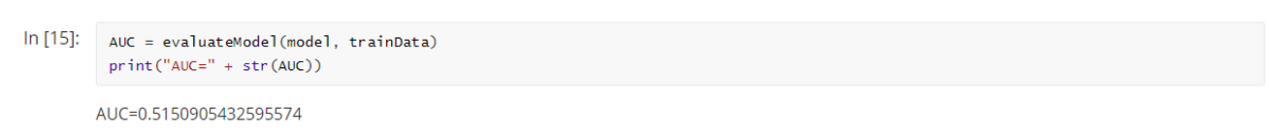
模型训练代码，以决策树为例：

|  |
| --- |
| from pyspark.mllib.evaluation import BinaryClassificationMetrics  def evaluateModel(model, validationData):      """      计算模型的 AUC 率      :param model:      :param validationData:      :return:      """      score = model.predict(validationData.map(lambda p:p.features))      scoreAndLabels = score.zip(validationData.map(lambda p:p.label))      metrics = BinaryClassificationMetrics(scoreAndLabels)      AUC = metrics.areaUnderROC      return(AUC)  def evalAllParameter(trainData, validationData,impurityList, maxDepthList, maxBinsList):      """      评估模型的所有参数对AUC率的影响并显示出来      :param trainData:      :param validationData:      :param impurityList:      :param maxDepthList:      :param maxBinsList:      :return: 带有最佳参数的最佳模型      """      metrics = [trainEvaluateModel(trainData, validationData,                                    impurity,maxDepth,  maxBins  )                 for impurity in impurityList                 for maxDepth in maxDepthList                 for  maxBins in maxBinsList ]      Smetrics = sorted(metrics, key=lambda k: k[0], reverse=True)      bestParameter=Smetrics[0]      print("调校后最佳参数：impurity:" + str(bestParameter[2]) + "  ,maxDepth:" + str(bestParameter[3]) +  "  ,maxBins:" + str(bestParameter[4])   + "  ,结果AUC = " + str(bestParameter[0]))      return bestParameter[5] |

1. 重要数据结果分析

决策树模型的训练参数及AUC率：

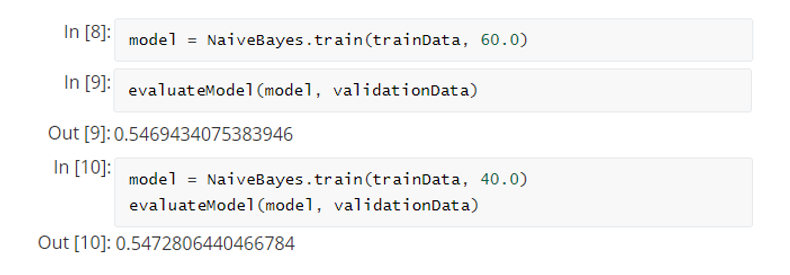




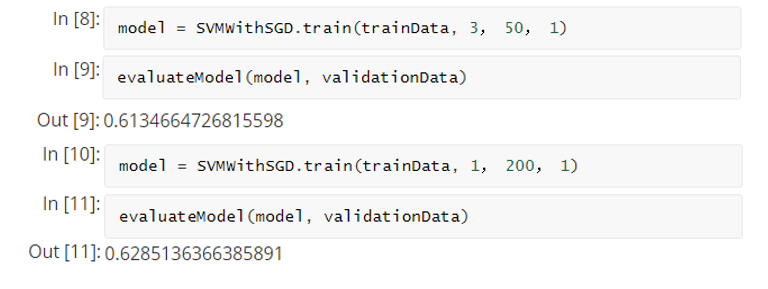
逻辑回归模型的训练参数及AUC率：



朴素贝叶斯模型的训练参数及AUC率：



向量机模型的训练参数及AUC率：



综上所述，当前最佳模型AUC率约为0.66。

# 十、总结及心得体会

在本次航空公司延误和取消分析实验中，我们小组展开了积极而有序的合作。通过小组会议，我们充分商议并分工合作，拆分任务，为实验的顺利进行奠定了基础。在合作过程中，我们共同克服了遇到的各种困难，通过细化和完善的汇总过程，进一步提高了报告的质量。

实验的关键在于对大数据分析工具的灵活运用，我们选择了使用 PySpark 进行数据分析。通过使用 Python 进行 Spark Application 编程，我们成功地完成了数据获取、处理、分析和可视化等步骤。这使得我们不仅复习了在理论课上学到的知识，而且在实践中加深了对大数据分析与计算相关内容的理解。

在实验的过程中，我们克服了各种小问题，这些问题的解决对最终结果的成功至关重要。我们在相互帮助和交流中，通过网络学习了一些新知识，逐步提高了团队的整体水平。

理论与实践的结合是本次实验的一大特点。在理论课堂上，我们学到了大数据计算系统的数据模型和处理算法等知识。然而，实际操作中，我们发现实验并不轻松，具有一定难度。通过个人思考和团队合作，我们成功地克服了这些难题，最终完成了实验。

在这个过程中，我们不仅提升了个人独立思考和交流合作的实践能力，还收获了对大数据分析与计算领域的深刻认识。感谢本次实验，让我们在理论与实践的交融中受益匪浅。

# 十一、对本项目过程及方法、手段的改进建议

可以进一步尝试使用深度学习模型，如Transformer等大参数量模型，以提高对未来航班取消情况的准确预测能力