

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 罗瑜

实验教师 罗瑜

学生信息：

序号	学号	姓名
1	***	***
2	***	***
3	***	***
4	***	***
5	***	***
6	***	***

电子科技大学

项目报告

指导教师： 罗瑜

地点：二教 205

一、项目名称：航空公司延误和取消分析项目

二、项目时间：2024. 12. 2—2024. 12. 19

三、项目原理

本项目基于 PySpark 这一强大的工具，融合了 Python 的灵活性与 Apache Spark 的分布式计算能力，旨在实现对大规模数据集的高效处理与深入分析。PySpark 框架使得数据分析任务，如数据提取、转换、聚合和可视化等，可以通过熟悉的 Python 接口在 Spark 的分布式环境中执行，极大地提升了处理大数据的速度和便捷性。

此外，通过在 Hadoop 集群的每个节点上配置 Python 运行环境，我们能够确保 Python 编写的 Spark 应用程序能够在分布式节点上无缝运行，实现对海量数据的快速处理和分析，这对于从大数据中提取有价值的信息、优化业务决策和提高运营效率具有重要意义。

四、项目内容

项目需要读入并处理数据（可以存入 HDFS、HBase 等），并通过生成图表和表格进行数据解析，以便能更清楚地了解这个数据集中的内容。具体分析内容包括：

- （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。

五、需求分析与设计

5.1 项目背景

在当前的航空运输行业中，航班延误和取消对航空公司的运营效率和旅客的出行计划造成了显著影响。随着大数据技术的发展，航空公司和机场管理者越来越依赖数据分析来优化运营决策、提高服务质量，并增强对航班延误和取消的预测能力。本项目在这样的背景下应运而生，旨在通过分析航空数据集，揭示航班延误和取消的关键因素，评估不同航空公司的表现，并预测未来的航班取消情况。通过运用 PySpark 等大数据处理技术，我们能够处理和分析大规模的航班数据，为航空公司提供数据驱动的见解，帮助他们更好地管理风险，优化资源配置，并改善客户体验。本项目不仅有助于提升航空行业的运营效率，也为旅客提供了更加可靠的出行信息，具有重要的实际应用价值和广阔的发展前景。

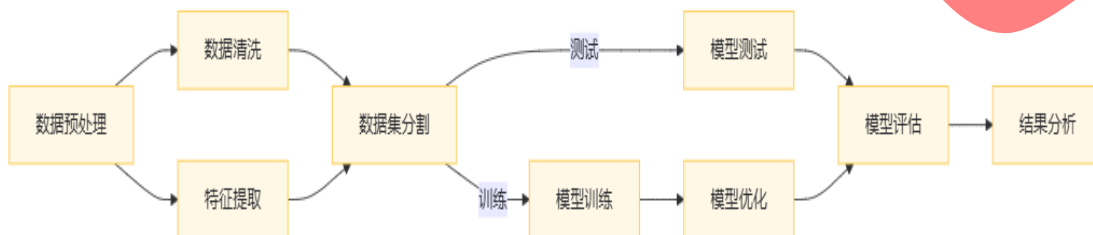
5.2 需求概述

本项目的核心目标是满足航空公司和旅客对航班取消预测的需求。为实现这一目标，项目将重点进行以下需求分析：

- 数据预处理：**对航班数据进行清洗和整理，以确保数据质量，从而为分析航班延误和取消的原因奠定基础。
- 数据分析与可视化：**深入探讨航班延误和取消的潜在因素，分析延误数据，包括延误时间最长的前 10 名航班、一天和一周中延误最严重的时间段等，以便为航空公司提供优化运营策略的依据。
- 机器学习模型构建：**基于数据分析的结果，建立预测航班取消的机器学习模型，以满足航空公司和旅客在提前预警和决策支持方面的需求。

通过这些分析和模型构建，项目旨在为航空公司提供数据驱动的见解，帮助其更好地管理航班运营，提高服务质量。

5.3 图示说明



本流程图总结了“航空公司延误和取消分析项目”的主要步骤，从数据预处理开始，包括对数据进行清洗以及对特征进行提取，再到数据分割，交给吗，模型训练，模型通过损失函数以及梯度下降进一步优化模型，然后再到测试集上测试，对模型进行评估，最后进行结果分析。

5.4 人员分工

本组将项目进行拆解，每人完成其中一个或几个部分。每个人的报告反映了其本人的工作量情况，不同的人的报告内容不同，合起来为一个完整项目。

小组成员	工作量
***	<ul style="list-style-type: none">分析一天中延误最严重的飞行时间并可视化分析说明使用梯度提升树(GBDT)模型对航班取消进行预测
***	<ul style="list-style-type: none">计算延误的和没有延误的航空公司的比例并可视化分析说明使用随机森林模型对航班取消情况进行预测分析
***	<ul style="list-style-type: none">查看飞机延误时间最长的前 10 名航班并可视化分析说明使用 SVM 支持向量机模型对航班取消进行预测
***	<ul style="list-style-type: none">分析一周中延误最严重的飞行时间并可视化分析说明使用线性判别分析模型对航班取消进行预测
***	<ul style="list-style-type: none">分析短途航班和长途航班，哪种航班取消更严重？并完成可视化
***	<ul style="list-style-type: none">使用逻辑回归模型对航班取消情况进行预测分析

六、项目计划

首先进行需求分析，明确项目目标和预期成果。接着配置基础环境，为后续的数据准备和分析奠定基础。随后，我们将深入分析数据集，识别影响航班延误的关键因素，并依据这些分析选择合适的标签和模型进行预测。

在模型训练阶段，我们将利用历史数据优化模型参数，以提升预测准确性。训练完成后，通过一系列评估指标对模型性能进行测试，确保模型在预测航班取消时的可靠性。

最后，我们将总结实验过程、挑战和成果，并撰写项目报告，记录分析和模型评估结果，为未来的研究和应用提供参考。

七、项目环境配置管理

7.1 操作系统

本项目选用 Linux Ubuntu 22.04

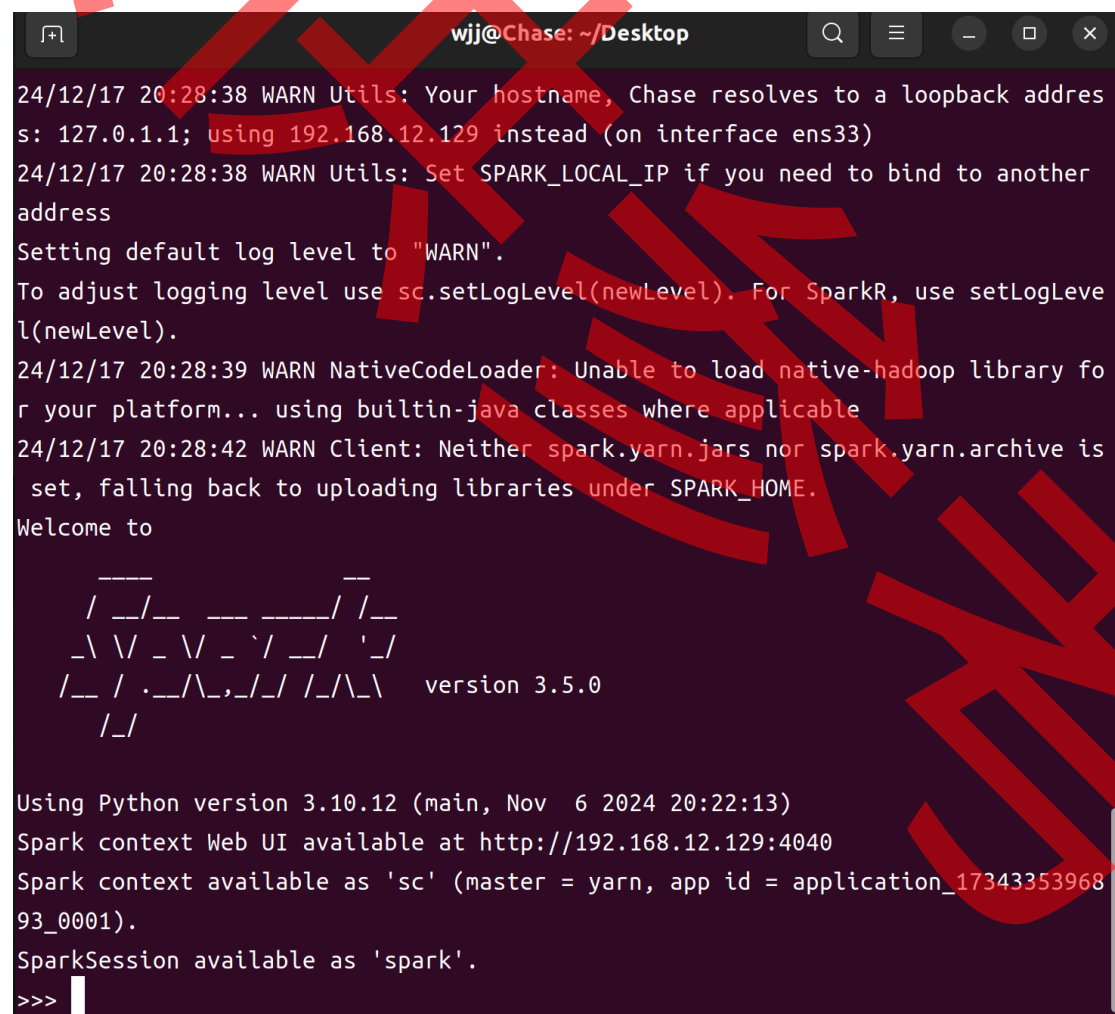
7.2 开发工具

主要选择的开发工具包括：Pyspark, Jupyter Notebook, Pycharm, vmware

7.3 配置过程

1. pyspark 的安装与测试

在实验一和实验二中已经安装好了 pyspark, 这里只对其进行测试。输入 `pyspark --master yarn --deploy-mode client` 指令, 在 Hadoop YARN 上运行 PySpark, 结果如下图所示, 可以看到运行正常。



```
wjj@Chase: ~/Desktop
24/12/17 20:28:38 WARN Utils: Your hostname, Chase resolves to a loopback address: 127.0.1.1; using 192.168.12.129 instead (on interface ens33)
24/12/17 20:28:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/12/17 20:28:39 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/12/17 20:28:42 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Welcome to

  ____      _
 / ___|    / \
 \  ___|  _/ _ \
  \___|_|_/_/ \_\

version 3.5.0

Using Python version 3.10.12 (main, Nov 6 2024 20:22:13)
Spark context Web UI available at http://192.168.12.129:4040
Spark context available as 'sc' (master = yarn, app id = application_1734335396893_0001).
SparkSession available as 'spark'.
>>>
```

2. Jupyter Notebook 安装及测试

首先安装 anaconda，安装结果如下：

```
==> For changes to take effect, close and re-open your current shell. <==

If you'd prefer that conda's base environment not be activated on startup,
set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Anaconda3!

=====

Working with Python and Jupyter is a breeze in DataSpell. It is an IDE
designed for exploratory data analysis and ML. Get better data insights
with DataSpell.

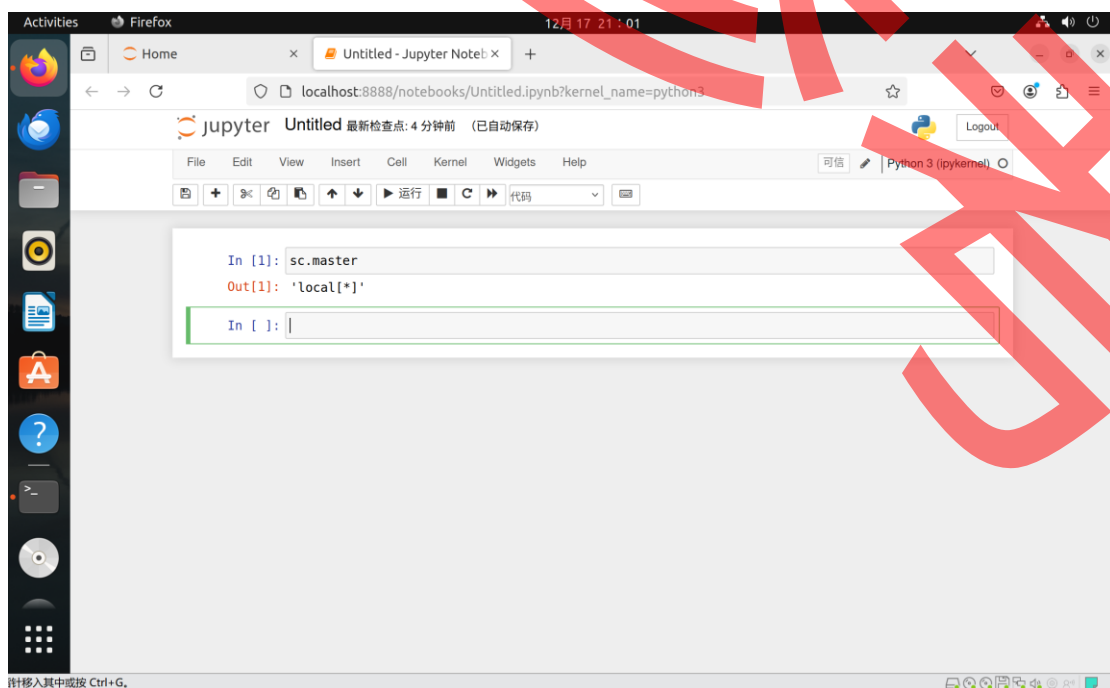
DataSpell for Anaconda is available at: https://www.anaconda.com/dataspell
```

然后对 jupyter 进行配置，配置完成后输入 pyspark 指令得到如下网址：

```
[C 20:54:21.981 NotebookApp]

To access the notebook, open this file in a browser:
    file:///hadoop/.local/share/jupyter/runtime/nbserver-7800-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=229b9a2ff4df6c8a0c3d2aeccf689cc5d81bfad4925
fc6f5
```

在浏览器中打开，输入代码 `sc.master`，即可得到下图运行结果：



八、项目实践过程

8.1 读取数据

首先将数据下载好，放在虚拟机/hadoop/data 文件夹下，再将其上传至 hadoop，

如下图：

```
hadoop@Chase:~$ hadoop fs -put /hadoop/data/DelayedFlights.csv /flight
2024-12-18 11:08:31,398 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop@Chase:~$ hadoop fs -ls -R /flight
2024-12-18 11:08:51,291 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
-rw-r--r--  1 hadoop supergroup  247963212 2024-12-18 11:08 /flight/DelayedFlig
hts.csv
hadoop@Chase:~$
```

编写如下代码，读取航班数据，并展示数据的列名以及总行数。

代码 1: 读取数据并统计总行数

```
from numpy import select
from pyspark import sql
from pyspark.sql import SQLContext
from pyspark import SparkContext
import pyspark.sql.functions as F
from typing import ForwardRef
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

# 读取数据
data = spark.read\
    .option("header","true")\
    .option("inferSchema","true")\
    .csv("hdfs://localhost:9000/flight")
# 展示 data 的列名
print("Columns:")
print(data.columns)

# 计算 data 的总行数并展示
total_rows = data.count()
print(f"Total number of rows: {total_rows}")
```

相关运行结果图如下：


```

In [11]: from pyspark.sql.functions import col
def calculate_delay_stats():
    """
    Task2: 计算有延误和没有延误的航空公司数量及其比例。
    """
    # 计算有延误的航空公司数量
    delay_airlines_count = data.filter(col("ArrDelay") > 0).select("UniqueCarrier").distinct().count()

    # 计算总航空公司数量
    total_airlines_count = data.select("UniqueCarrier").distinct().count()

    # 计算没有延误的航空公司数量
    no_delay_airlines_count = total_airlines_count - delay_airlines_count

    # 计算比例
    if no_delay_airlines_count > 0:
        ratio = delay_airlines_count / no_delay_airlines_count
    else:
        ratio = float('inf') # 如果没有未延误的航空公司, 则比例是无限大

    # 返回结果
    return {
        "delay_airlines_count": delay_airlines_count,
        "no_delay_airlines_count": no_delay_airlines_count,
        "ratio": ratio
    }

# 假设data是已经存在的DataFrame
# 调用函数并打印结果
stats = calculate_delay_stats()
print(f"Number of airlines with delay: {stats['delay_airlines_count']}")
print(f"Number of airlines without delay: {stats['no_delay_airlines_count']}")
print(f"The proportion of delayed to non-delayed airlines: {stats['ratio']:.2f}")

[Stage 20:>                                     (0 + 2) / 2]

Number of airlines with delay: 20
Number of airlines without delay: 0
The proportion of delayed to non-delayed airlines: inf

```

8.2 计算延误的和没有延误的航空公司的比例

为了计算延误的和没有延误的航空公司的比例。我们只需要统计有延误的航司（即存在 `ArrDelay>0`）数量，然后用总数减去有延误的航司数量，就可以得到无延误的航司数量，进一步可以得到比例。相关代码如下——

代码 2：计算延误的和没有延误的航空公司的比例

```

from pyspark.sql.functions import col
def calculate_delay_stats():
    """
    Task2: 计算有延误和没有延误的航空公司数量及其比例。
    """
    # 计算有延误的航空公司数量
    delay_airlines_count = data.filter(col("ArrDelay") > 0).select("UniqueCarrier").distinct().count()

    # 计算总航空公司数量
    total_airlines_count = data.select("UniqueCarrier").distinct().count()

    # 计算没有延误的航空公司数量
    no_delay_airlines_count = total_airlines_count - delay_airlines_count

    # 计算比例
    if no_delay_airlines_count > 0:
        ratio = delay_airlines_count / no_delay_airlines_count
    else:
        ratio = float('inf') # 如果没有未延误的航空公司, 则比例是无限大

    # 返回结果

```



```

    return {
        "delay_airlines_count": delay_airlines_count,
        "no_delay_airlines_count": no_delay_airlines_count,
        "ratio": ratio
    }
# 假设 data 是已经存在的 DataFrame
# 调用函数并打印结果
stats = calculate_delay_stats()
print(f"Number of airlines with delay: {stats['delay_airlines_count']}")
print(f"Number of airlines without delay: {stats['no_delay_airlines_count']}")
print(f"The proportion of delayed to non-delayed airlines: {stats['ratio']:.2f}")

```

这段代码定义了一个函数 `calculate_delay_stats`，用来计算航班延误和非延误的航空公司数量及其比例。它通过筛选和统计 `data DataFrame` 中的数据，得出有延误的航空公司数量，然后从总航空公司数量中减去这个值，得到无延误的公司数量，并计算两者的比例。最后，函数返回包含这些统计信息的字典，并通过打印语句输出结果。代码运行结果如下：

```

In [2]: from pyspark.sql.functions import col
def calculate_delay_stats():
    """
    Task2: 计算有延误和没有延误的航空公司数量及其比例。
    """
    # 计算有延误的航空公司数量
    delay_airlines_count = data.filter(col("ArrDelay") > 0).select("UniqueCarrier").distinct().count()
    # 计算总航空公司数量
    total_airlines_count = data.select("UniqueCarrier").distinct().count()
    # 计算没有延误的航空公司数量
    no_delay_airlines_count = total_airlines_count - delay_airlines_count
    # 计算比例
    if no_delay_airlines_count > 0:
        ratio = delay_airlines_count / no_delay_airlines_count
    else:
        ratio = float('inf') # 如果没有未延误的航空公司，则比例是无限大
    # 返回结果
    return {
        "delay_airlines_count": delay_airlines_count,
        "no_delay_airlines_count": no_delay_airlines_count,
        "ratio": ratio
    }
# 假设data是已经存在的DataFrame
# 调用函数并打印结果
stats = calculate_delay_stats()
print(f"Number of airlines with delay: {stats['delay_airlines_count']}")
print(f"Number of airlines without delay: {stats['no_delay_airlines_count']}")
print(f"The proportion of delayed to non-delayed airlines: {stats['ratio']:.2f}")

[Stage 11:=====> (5 + 3) / 8]
Number of airlines with delay: 20
Number of airlines without delay: 0
The proportion of delayed to non-delayed airlines: inf

```

但是我们发现，事实上，所有航司或多或少都存在延误的航班，不存在完全没有延误的航司。因此求比例无意义。为了更好探究这个问题，我们进一步探究每个公司延误的和没有延误的航班比例。

为了计算每个公司延误的和没有延误的比例，我们将关注于两个关键特征：航空公司的唯一编码（`UniqueCarrier`）和航班的抵达延误时间（`ArrDelay`）。通过判断 `ArrDelay` 字段值是否大于零来确定航班是否延误。

基于这一标准，我们将对数据进行分类统计，并计算出每个航空公司延误航班与未延误航班的次数，然后将延误次数与没有延误次数相除，即可得到比例。

代码 3：计算每个航司延误的和没有延误的航班比例并绘图

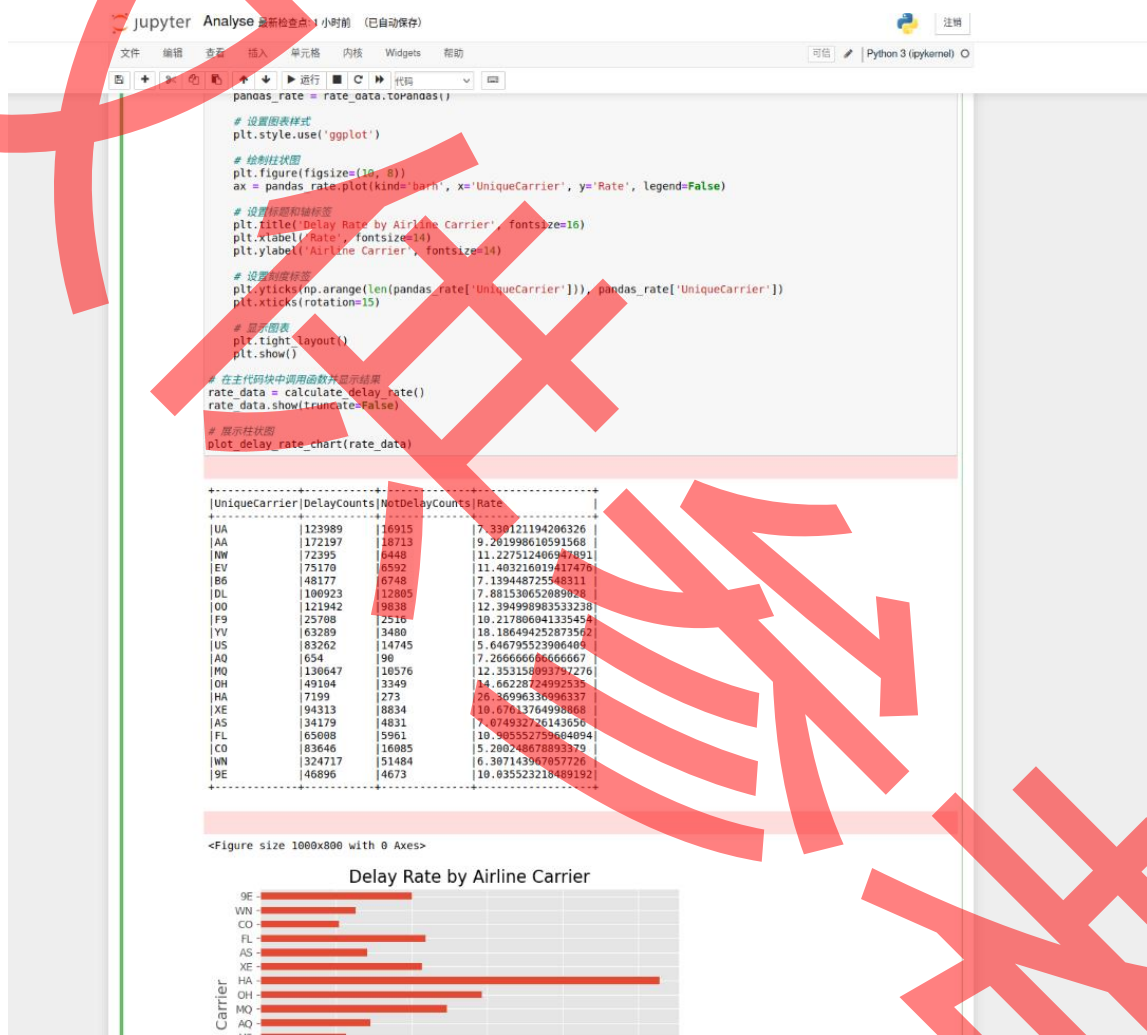
```
import numpy as np
def calculate_delay_rate():
    # 定义需要的列
    required_columns = ['UniqueCarrier', 'ArrDelay']
    # 选择需要的列
    selected_data = data.select([column for column in data.columns if column in
required_columns])
    # 计算每个航空公司的延误航班数
    delay_counts = selected_data.filter(selected_data['ArrDelay'] >
0).groupBy('UniqueCarrier').count()
    delay_counts = delay_counts.withColumnRenamed('count', 'DelayCounts')
    # 计算每个航空公司的没有延误的航班数
    not_delay_counts = selected_data.filter(selected_data['ArrDelay'] <=
0).groupBy('UniqueCarrier').count()
    not_delay_counts = not_delay_counts.withColumnRenamed('count', 'NotDelayCounts')
    # 将延误航班数和没有延误的航班数连接起来
    joined_data = delay_counts.join(not_delay_counts, on='UniqueCarrier', how='left_outer')
    # 计算延误率
    rate_data = joined_data.withColumn('Rate', (joined_data['DelayCounts'] /
joined_data['NotDelayCounts']))
    return rate_data

# 在主代码块中调用函数并显示结果
rate_data = calculate_delay_rate()
rate_data.show(truncate=False)
# 展示柱状图
pandas_rate = rate_data.toPandas()
# 设置图表样式
plt.style.use('ggplot')
# 绘制柱状图
plt.figure(figsize=(10, 8))
ax = pandas_rate.plot(kind='barh', x='UniqueCarrier', y='Rate', legend=False)
# 设置标题和轴标签
plt.title('Delay Rate by Airline Carrier', fontsize=16)
plt.xlabel('Rate', fontsize=14)
plt.ylabel('Airline Carrier', fontsize=14)
# 设置刻度标签
plt.yticks(np.arange(len(pandas_rate['UniqueCarrier'])), pandas_rate['UniqueCarrier'])
plt.xticks(rotation=15)
# 显示图表
```

```
plt.tight_layout()
plt.show()
```

这段代码通过定义 `calculate_delay_rate` 函数来计算航空公司的航班延误率，并将其可视化。函数从数据集中筛选出航空公司代码和到达延误时间，分别统计每个航空公司的延误和非延误航班数量，然后计算延误率。最后，代码将结果转换为 `Pandas DataFrame`，并使用 `Matplotlib` 绘制出每个航空公司的延误率柱状图。

运行过程如图：



同样，我们也可以画出关于航空公司延误比例的饼图，能更直观地感受到各个航空公司之间的比例关系。

```
import numpy as np
import matplotlib.pyplot as plt
from pyspark.sql.functions import col

def calculate_delay_rate():
    # 定义需要的列
```

```

required_columns = ['UniqueCarrier', 'ArrDelay']
# 选择需要的列
selected_data = data.select([column for column in data.columns if column in
required_columns])
# 计算每个航空公司的延误航班数
delay_counts = selected_data.filter(selected_data['ArrDelay'] >
0).groupBy('UniqueCarrier').count()
delay_counts = delay_counts.withColumnRenamed('count', 'DelayCounts')
# 计算每个航空公司的没有延误的航班数
not_delay_counts = selected_data.filter(selected_data['ArrDelay'] <=
0).groupBy('UniqueCarrier').count()
not_delay_counts = not_delay_counts.withColumnRenamed('count', 'NotDelayCounts')
# 将延误航班数和没有延误的航班数连接起来
joined_data = delay_counts.join(not_delay_counts, on='UniqueCarrier', how='left_outer')
# 计算延误率
rate_data = joined_data.withColumn('Rate', (joined_data['DelayCounts'] /
joined_data['NotDelayCounts']))
return rate_data.fillna(0) # 确保没有 NaN 值

# 在主代码块中调用函数并显示结果
rate_data = calculate_delay_rate()
rate_data.show(truncate=False)

# 转换为 Pandas DataFrame
pandas_rate = rate_data.toPandas()

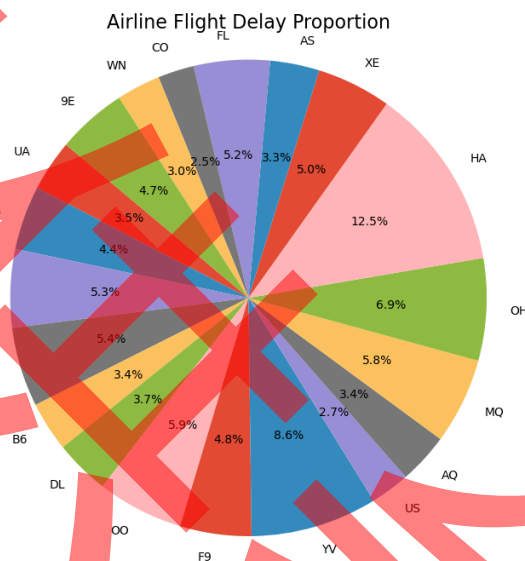
# 绘制饼图
plt.figure(figsize=(10, 8))
plt.pie(pandas_rate['Rate'], labels=pandas_rate['UniqueCarrier'], autopct='%1.1f%%',
startangle=140)
plt.title('Airline Flight Delay Proportion', fontsize=16) # 饼图的标题
plt.axis('equal') # 确保饼图是圆形的
plt.show()

```

这段代码与上面画柱状图相似，首先计算每个航空公司的延误和非延误航班数，然后计算延误率，并且使用 `fillna(0)` 确保所有的延误率都被计算，即使某些航空公司没有非延误航班。最后，将结果转换为 `Pandas DataFrame` 以便绘图。使用 `Matplotlib` 库，代码绘制了一个饼图，展示每个航空公司的航班延误比例，其中每个扇区的标签为航空公司代码，扇区大小表示延误率的百分比，图表标题为“Airline Flight Delay Proportion”，确保饼图形状为圆形。

运行结果如下：

UniqueCarrier	DelayCounts	NotDelayCounts	Rate
UA	123989	16915	7.330121194206326
AA	172197	18713	9.20198610591568
NW	72395	6448	11.227512406947891
EV	75170	6592	11.403216019417476
B6	48177	6748	7.139448725548311
DL	100923	12805	7.881530652089028
OO	121942	9838	12.39499898353238
F9	25788	2516	10.217806041335454
VV	63289	3480	18.186494252873562
US	83262	14745	5.646795523906409
AQ	654	90	7.266666666666667
MQ	130647	10576	12.353158093797276
OH	49104	3349	14.66228724992535
HA	7199	273	26.36996336996337
XE	94313	8834	10.67613764998868
AS	34179	4831	7.074932726143656
FL	65008	5961	10.905552759604094
CO	83646	16085	5.200248678893379
WN	324717	51484	6.307143967057726
9E	46896	4673	10.035523218489192



除上述两种图之外，我们还可以画一个水平条形图，展示了不同航空公司的航班延误和非延误的数量，一边直观地比较航班延误与非延误的数量以及航空公司之间的延误情况。

```
import numpy as np
import matplotlib.pyplot as plt
from pyspark.sql.functions import col

def calculate_delay_counts():
    # 计算每个航空公司的延误航班数
    delay_counts = data.filter(data['ArrDelay'] > 0).groupBy('UniqueCarrier').count()
    delay_counts = delay_counts.withColumnRenamed('count', 'DelayCounts')

    # 计算每个航空公司的没有延误的航班数
    not_delay_counts = data.filter(data['ArrDelay'] <= 0).groupBy('UniqueCarrier').count()
    not_delay_counts = not_delay_counts.withColumnRenamed('count', 'NotDelayCounts')

    # 将延误航班数和没有延误的航班数连接起来
    joined_data = delay_counts.join(not_delay_counts, on='UniqueCarrier', how='outer')
```

```

# 填充可能的空值
joined_data = joined_data.fillna(0)

return joined_data

# 在主代码块中调用函数并显示结果
delay_data = calculate_delay_counts()
delay_data.show(truncate=False)

# 转换为 Pandas DataFrame
pandas_delay_data = delay_data.toPandas()

# 设置图表样式
plt.style.use('ggplot')

# 绘制水平条形图
fig, ax = plt.subplots(figsize=(12, 6))
bar_width = 0.35
index = np.arange(len(pandas_delay_data['UniqueCarrier']))

bar1 = ax.barh(index, pandas_delay_data['DelayCounts'], bar_width, label='Delayed',
color='orange')
bar2 = ax.barh(index, pandas_delay_data['NotDelayCounts'], bar_width,
left=pandas_delay_data['DelayCounts'], label='Not Delayed', color='blue')

# 添加标签和标题
ax.set_xlabel('Flight Counts')
ax.set_ylabel('Airline Carrier')
ax.set_title('Flight Delay and Non-Delay Counts by Airline Carrier')
ax.set_yticks(index)
ax.set_yticklabels(pandas_delay_data['UniqueCarrier'])
ax.legend()

# 显示图表
plt.tight_layout()
plt.show()

```

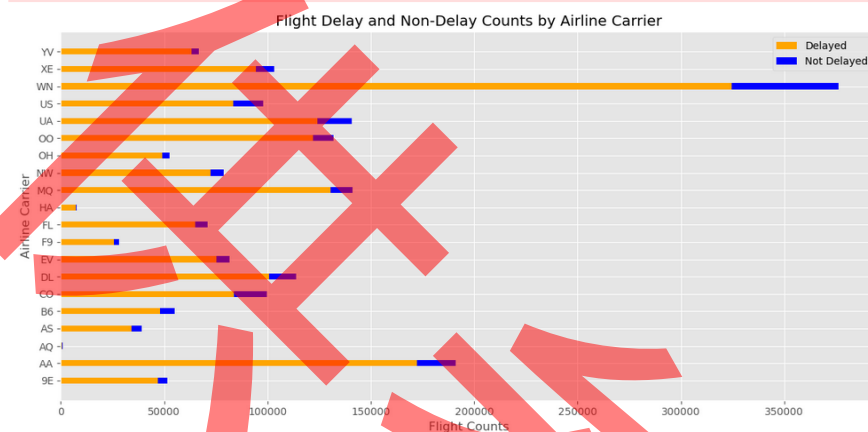
这段代码通过 PySpark 计算了不同航空公司的航班延误和非延误数量，并将结果转换为 Pandas DataFrame 以便于绘制水平条形图。代码使用 Matplotlib 库设置了一个图表样式，并绘制了一个水平条形图，其中延误航班用橙色表示，非延误航班用蓝色表示，直观展示了各航空公司的航班准时性。

运行结果如下：

```
ax.set_title('Flight Delay and Non-Delay Counts by Airline Carrier')
ax.set_yticks(index)
ax.set_yticklabels(pandas_delay_data['UniqueCarrier'])
ax.legend()

# 显示图表
plt.tight_layout()
plt.show()
```

UniqueCarrier	DelayCounts	NotDelayCounts
9E	46896	4673
AA	172197	18713
AQ	654	90
AS	34179	4831
B6	48177	6748
CO	83646	16085
DL	100923	12805
EV	75170	6592
F9	25708	2516
FL	65008	5961
HA	7199	273
MQ	130647	10576
NW	72395	6448
OH	49104	3349
OO	121942	9838
UA	129989	16915
US	83262	14745
WN	324717	51484
XE	94313	8834
YV	63289	3480



8.7 建立机器学习算法模型，预测未来航班取消情况

我使用随机森林（Random Forest）机器学习算法进行预测。

代码 4：数据预处理与训练集划分

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
# 加载数据集
flight_data = pd.read_csv('./DelayedFlights.csv')
print(flight_data.head(10))
features = [
    'Year', 'Month', 'DayOfMonth', 'DayOfWeek', 'CRSDepTime', 'CRSArrTime',
    'UniqueCarrier', 'FlightNum', 'TailNum', 'CRSElapsedTime', 'Distance',
    'Origin', 'Dest'
]
target = 'Cancelled'
# 提取特征和目标变量
```



```

X = flight_data[features].copy()
y = flight_data[target].copy()
# 移除含有缺失值的行
X.dropna(inplace=True)
y = y.loc[X.index] # 保持特征和目标索引一致
# 区分数值特征和类别特征
numeric_features = [
    'Year', 'Month', 'DayofMonth', 'DayOfWeek', 'CRSDepTime', 'CRSArrTime',
    'CRSElapsedTime', 'Distance'
]
categorical_features = ['UniqueCarrier', 'FlightNum', 'TailNum', 'Origin', 'Dest']
# 设置预处理流程
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])
# 应用预处理流程
X_preprocessed = preprocessor.fit_transform(X)
# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X_preprocessed, y, test_size=0.2, random_state=42
)

```

(2) 随机森林进行预测

首先，导入所需的库和模块，包括随机森林分类器、管道、评估指标等。然后，定义一个预处理管道，包含数值特征标准化和类别特征一键编码。接着，创建一个随机森林分类器，并设置类别权重以应对数据不平衡。将预处理和分类器组合成管道，并用训练集训练模型。模型训练后，用测试集进行预测并计算准确率、分类报告、混淆矩阵以及 ROC 曲线和 AUC 值。最后，进行预测。具体代码如下——

代码 5：随机森林训练与预测

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, accuracy_score
import numpy as np
# 创建预处理管道
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

```

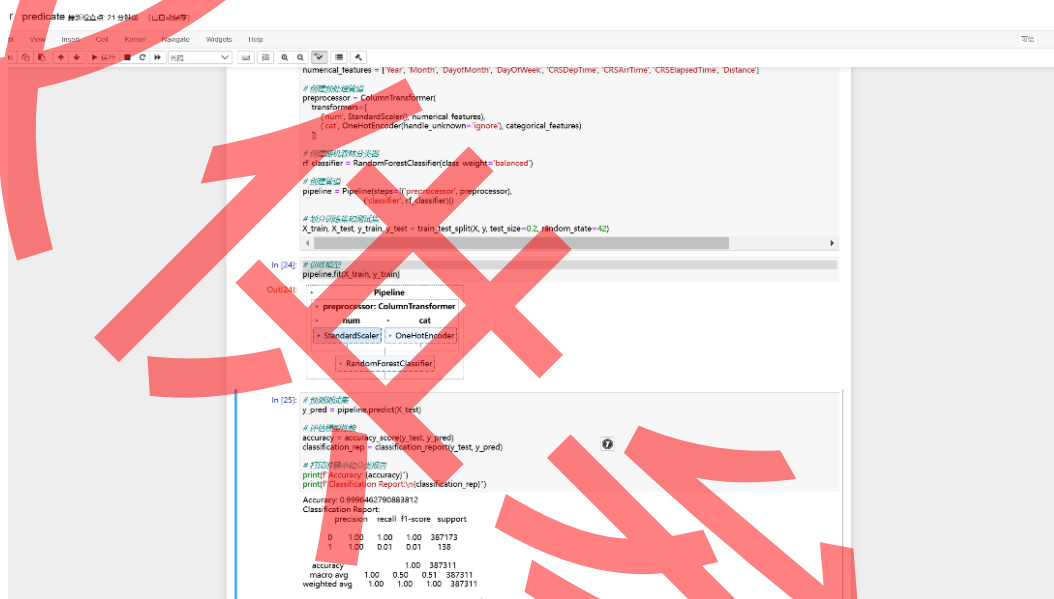
```

    ])
# 创建随机森林分类器
rf_classifier = RandomForestClassifier(class_weight='balanced')
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                            ('classifier', rf_classifier)])

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# 训练模型
pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)

```



```

numerical_features = ['Year', 'Month', 'DayOfMonth', 'DayOfWeek', 'CRSDepTime', 'CRSAirTime', 'CRSElapsedTime', 'Distance']
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])
# 随机森林分类器
rf_classifier = RandomForestClassifier(class_weight='balanced')
# 组合管道
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                            ('classifier', rf_classifier)])
# 划分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# 训练模型
pipeline.fit(X_train, y_train)

In [24]:
Out[24]:
Pipeline
  • preprocessor: ColumnTransformer
    • num
    • cat
    • StandardScaler
    • OneHotEncoder
  • classifier: RandomForestClassifier

In [25]:
y_pred = pipeline.predict(X_test)

# 计算准确率
accuracy = accuracy_score(y_test, y_pred)
classification_report = classification_report(y_test, y_pred)
# 打印结果
print('Accuracy:', accuracy)
print('Classification Report:', classification_report)

Accuracy: 0.999462796883812
Classification Report
precision  recall  f1-score  support
0         1.00    1.00    1.00    387173
1         1.00    0.01    0.01     138
accuracy          1.00    0.99    0.51    387311
macro avg         1.00    0.50    0.51    387311
weighted avg         1.00    1.00    1.00    387311

```

具体的分析结果和评测指标将在第九部分详细阐述。

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）

Task 2. 计算延误的和没有延误的航空公司的比例。

```

Number of airlines with delay: 20
Number of airlines without delay: 0
The proportion of delayed to non-delayed airlines: inf

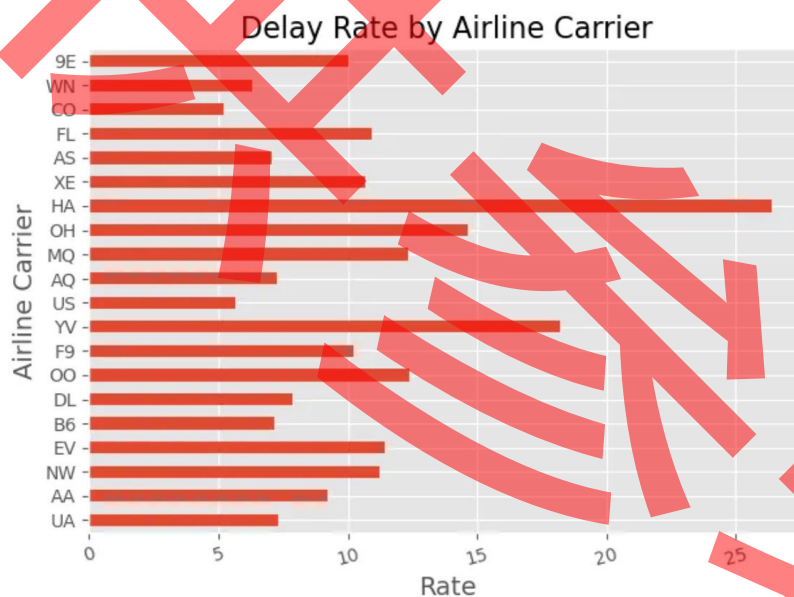
```

根据我们的代码运行，没有完全不延误的航司。即所有航司都或多或少有航班存在延误状态。于是，我们统计各航空公司延误与未延误方面的航班比例。

下图展示了各航空公司延误与未延误的航班情况比例列表。第一列为航空公司编码，第二列为延误航班数（DelayCounts），第三列为未延误航班数（NotDelayCounts），最后一列为该航司对应的延误与未延误比例。

UniqueCarrier	DelayCounts	NotDelayCounts	Rate
UA	123989	16915	7.330121194206326
AA	172197	18713	9.201998610591568
NW	72395	6448	11.227512406947891
EV	75170	6592	11.403216019417476
B6	48177	6748	7.139448725548311
DL	100923	12805	7.881530652089028
OO	121942	9838	12.394998983533238
F9	25708	2516	10.217806041335454
YV	63289	3480	18.186494252873562
US	83262	14745	5.646795523906409
AQ	654	90	7.266666666666667
MQ	130647	10576	12.353158093797276
OH	49104	3349	14.66228724992535
HA	7199	273	26.36996336996337
XE	94313	8834	10.67613764998868
AS	34179	4831	7.074932726143656
FL	65008	5961	10.905552759604094
CO	83646	16085	5.200248678893379
WN	324717	51484	6.307143967057726
9E	46896	4673	10.035523218489192

从表格中可以看出，不同航空公司的航班延误和未延误情况存在显著差异。可以看出 HA 航空公司延误比最高，达到了 26.36；CO 航空公司最低，仅 5.2。我们还可以用图形化表示，更直观反应各航司的延误比。

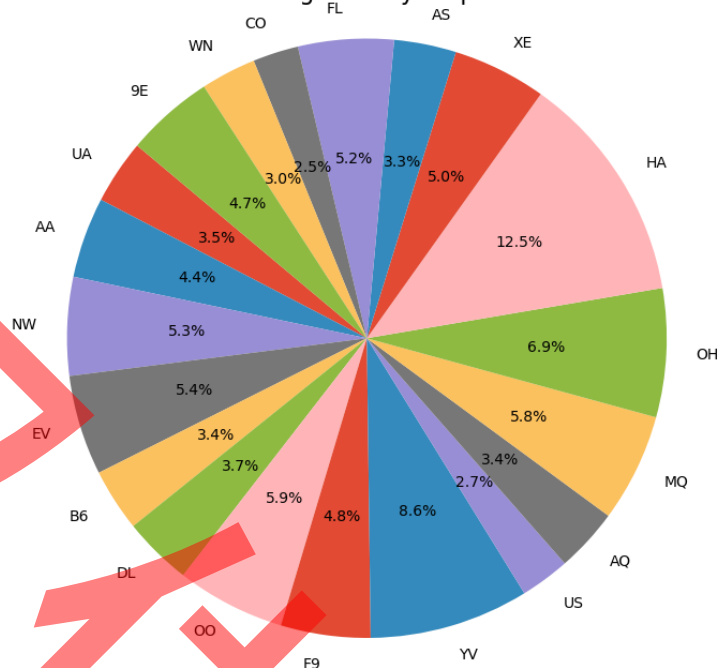


如上图所示的水平条形图，展示了不同航空公司的航班延误率。横轴表示延误率，纵轴列出了各个航空公司的代码。条形图的长度代表每个航空公司的航班延误率，数值越高，表示该公司的航班延误情况越严重。

从图中可以看出：

- 航空公司“HA”具有最高的延误率，远远超过其他航空公司，这可能表明该公司在航班调度或运营上存在问题。
- 航空公司“YV”和“OH”也显示出较高的延误率，但低于“HA”。
- 其他航空公司的延误率相对较低，分布较为均匀，但仍然可以看出一些航空公司的延误率略高于其他公司。

Airline Flight Delay Proportion

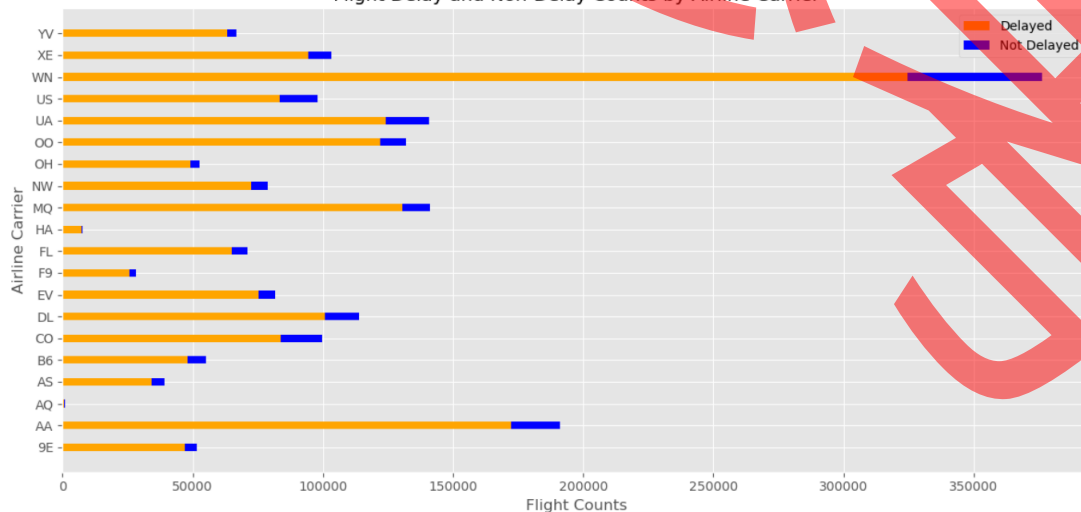


同样，如上图所示的饼图，图中每个扇形代表一个航空公司，扇形的大小表示该航空公司航班延误占总延误航班比例。

从图中可以看出：

- 航空公司“HA”占据了最大的扇形区域，约占12.5%，这表明它的航班延误比例最高。
- 航空公司“OH”和“YV”也占据了较大的比例，分别约为6.9%和8.6%。
- 其他航空公司如“UA”、“AA”、“NW”等的延误比例则相对较小，分布在3.0%到5.9%之间。
- 航空公司“CO”的延误比例最小，仅为2.5%。

Flight Delay and Non-Delay Counts by Airline Carrier



上图是一个水平条形图，图中每个航空公司的延误航班（橙色）和非延误航

班（蓝色）数量并排显示，以直观比较。

从图中可以看出：

- 航空公司“WN”的延误航班数量最多，远超过其他航空公司，同时它的非延误航班数量也相当高，显示出其航班数量基数大。
- 航空公司“UA”、“AA”和“CO”等也显示出较高的延误航班数量，但与“WN”相比，数量较少。
- 非延误航班数量在大多数航空公司中相对较少，这可能表明这些航空公司的航班更容易出现延误。

Task 6. 建立机器学习算法模型，预测未来航班取消情况。

● 随机森林模型

下图给出了随机森林的预测情况。随机森林模型在预测航班取消情况时表现出很高的准确率（0.9996），这意味着模型在大多数情况下都能正确预测航班是否会取消。然而，模型在处理正类（航班取消）的预测上存在明显不足。

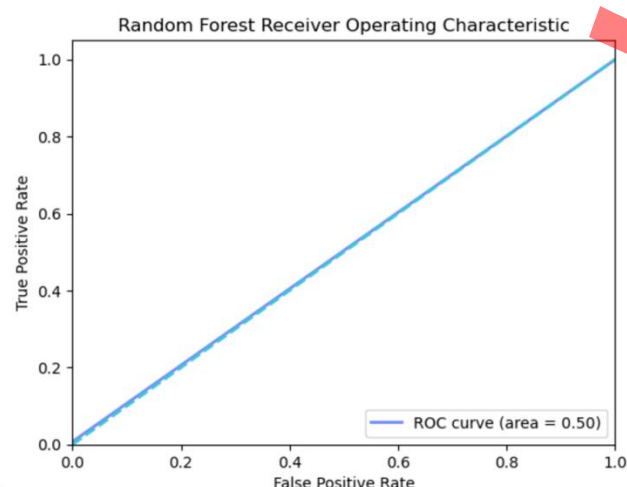
```
Accuracy: 0.9996462790883812
Classification Report:
      precision    recall  f1-score   support

0       1.00      1.00      1.00    387173
1       1.00      0.01      0.01      138

 accuracy          1.00      1.00      1.00    387311
 macro avg         1.00      0.50      0.51    387311
 weighted avg      1.00      1.00      1.00    387311
```

具体来说，模型的召回率仅为 0.0072，这表明模型在所有实际取消的航班中，只识别出了极少数。虽然精确度为 1.00，但由于正类样本数量极少，这个指标并不能很好地反映模型性能。

```
Recall: 0.007246376811594203
AUC: 0.5036231884057971
```



AUC 值为 0.5036，接近 0.5，随机森林模型在预测航班取消情况时存在明显

的偏向，需要进一步优化以提高对正类样本的识别能力。

十一、总结及心得体会

项目总结：

本项目通过使用 PySpark 对航空公司的航班延误和取消数据进行了深入分析，旨在揭示航班延误和取消的关键因素，并预测未来的航班取消情况。通过对数据的预处理、分析和可视化，我们发现不同航空公司的航班延误情况存在显著差异，其中 HA 航空公司的延误率最高，而 CO 航空公司的延误率最低。此外，我们还建立了随机森林机器学习模型来预测航班取消情况，尽管模型在整体准确率上表现良好，但在识别正类（航班取消）样本上存在不足。

心得体会：

- 团队合作的重要性：** 本项目的成功完成依赖于团队成员之间的紧密合作和分工。每个成员都在自己的领域内发挥了专长，共同推动了项目的进展。
- 数据处理能力的提升：** 通过本项目，我们对大数据技术有了更深入的理解和实践，特别是在使用 PySpark 进行数据处理和分析方面积累了宝贵的经验。
- 机器学习模型的挑战：** 在建立机器学习模型时，我们意识到模型的构建和优化是一个复杂的过程，需要不断地调整和测试以提高模型的性能。
- 问题解决能力：** 面对数据中的异常值和不平衡问题，我们学会了如何灵活运用不同的数据处理技术和机器学习策略来解决问题。

十二、对本项目过程及方法、手段的改进建议

本项目总体完成不错，但还存在可改进的地方，如下：

- 模型的选择：** 除了选择随机森林模型，我们还可以选择基于深度学习的 LSTM，这种模型可能对航班延误的预测效果更好。
- 模型调优：** 对于机器学习模型，我们可以探索更多的参数调优方法，如网格搜索或随机搜索，以找到最优的模型参数。
- 特征工程：** 进一步探索特征工程的可能性，比如特征选择和特征转换，以提高模型的预测能力。