Inflight: Practical 1

190018035

March 22, 2024

Contents

1 Introduction **1**
1.1 Project Objectives 1
1.2 Accomplishments 2
1.2.1 Implemented Features 2
1.3 Organization of the Report 2

2 Part 1: Core Features **3**
2.1 Feature Description 3

3 Part 2: Intermediate Features **5**
3.1 Feature Description 5
3.2 Difficulties Encountered 6

4 Part 3: Advanced Features **6**
4.1 Feature Description 6
4.2 Difficulties Encountered 6

5 Evaluation **6**

6 Conclusion **6**

1 Introduction

This project aims to provide an interactive application, InFlight, that allows users to explore and analyze the *Airline Reporting Carrier On-Time Performance* dataset [1]. The dataset contains information on the on-time performance of domestic flights operated by large air carriers in the United States between 1987 and 2020. The application is divided into three parts, each with a set of features that allow users to explore different aspects of the dataset. The following sections will detail the objectives, accomplishments, and features implemented in each part of the project.

1.1 Project Objectives

The main objectives of the project is to achieve the following:

1. Develop a user-friendly interface that provides visualizations and insights into the dataset.

2. Gain familiarity with PySpark and the PySpark SQL API for processing large datasets.

3. Gain valuable insights into the on-time performance of domestic flights in the United States.

1.2 Accomplishments

The following accomplishments have been achieved in the project:

1. Implemented a web-based application to interact with the dataset. This application utilizes Flask and PySpark as the backend and ReactJS and Tailwind CSS as the frontend.

2. Gained sufficient knowledge of PySpark and the PySpark SQL API by successfully completing all three parts of the project.

3. Developed a set of features that allow users to explore and analyze different aspects of the dataset, such as flight performance metrics, regional performance, and airline performance.

1.2.1 Implemented Features

The project is divided into three parts, each with a set of features that are implemented. The features implemented in each part are shown in Table 1 below.

Part 1	**Part 2**	**Part 3**
Read in and store the dataset using PySpark	Determine a "performance" metric for airports and display a radar chart to compare airport performance given a list of airports	
Display the total number of flights given a year	Chart region and state performance on on a US map	
Display the total number of flights given a range of years	Display a table of the best and worst performing states in each US region	
Display the total number of flights given comma-separated list of years		
Display the percentage of flights that departed on time, early and late		
Display the top reason for cancelled flights given a year		
Display the top 3 airports with the most punctual flights for a given year		
Determine a "performance" metric and display the top three worst performing airlines		

Table 1: Features implemented in each part of the project.

1.3 Organization of the Report

The remainder of this report is organized in the following manner:

- **Part 1: Core Features** - Details the core features implemented in the project, including reading in and storing the dataset, displaying flight statistics, and analyzing flight timeliness.
- **Part 2: Intermediate Features** - Details the intermediate features implemented in the project, including airport performance metrics, regional performance analysis, and state performance analysis.
- **Part 3: Advanced Features** -
- **Evaluation** - Evaluates the project based on the objectives and accomplishments of the project, along with a reflection on lessons learned and experience gained from the project.
- **Conclusion** - Concludes the report with a summary of the project and future work that can be done to improve the application.

2 Part 1: Core Features

This section details the core features implemented in the project. This includes converting the dataset to a suitable file format and summarizing a few key statistics about the dataset.

2.1 Feature Description

The following section details each of the core features implemented in the project.

Task 1: Read in and store the dataset using PySpark. This task involves reading in the dataset and storing it using PySpark.The dataset is read in from *airline.csv*, which has a file size of over >80GB, using PySpark. The dataset is then stored in in the Parquet file format [2]. Parquet is a columnar storage format the provides efficient storage and encoding of data. Parquet utilizes the algorithm outlined in Dremel, an interactive query system for analysis of read-only nested data [3], to represent nested structures. This algorithm can be broken down into two individual algorithms: the column stripping algorithm, which transforms complex records into a columnar format, and the record assembly algorithm, which reconstructs the original records from these columns when needed. Column stripping allows for better compression than traditional row-oriented storage formats, such as CSV, by exploiting the redundancy in the data found within columns of data, skipping non-relevant data very quickly [4]. Meanwhile, he row assembly algorithm ensures that despite the high compression, the data can be reconstructed quickly when needed, which is essential for large datasets such as the one used in the project, where I/O operations can be a bottleneck. By storing the dataset in Parquet format, the amount of data read from disk is minimized, and the data is stored in a more efficient manner, allowing for faster query times and better performance overall. The benefits of the Parquet format as opposed to CSV are outlined in Figure 1 below.

Dataset	Size on Amazon S3	Query Run Time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet Format	130 GB	6.78 seconds	2.51 GB	$0.01
Savings	87% less when using Parquet	34x faster	99% less data scanned	99.7% savings

Figure 1: Comparison of Parquet and CSV file formats in terms of file size, query run time, data scanned and cost [4].

Location: Code lines 13-27 in file `backend/scripts/convert.py`

Task 2: Display the total number of flights for a given year. This task allows the user to specify a year and view the total number of flights that occurred in that year. This is done by filtering the airline dataframe based on the specified year and counting the number of rows in the resulting dataframe.

Location: Code lines 26-36 in file `backend/part1/views.py`

Task 3: Display the total number of flights given a range of years. This task allows the user to specify a start year and end year and view the total number of flights that occurred within that range. This is done by filtering the airline dataframe based on the specified range of years and counting the number of rows in the resulting dataframe.

Location: Code lines 52-69 in file `backend/part1/views.py`

Task 4: Display the total number of flights given a list of years. This task allows the user to specify a list of years and view the total number of flights that occurred in each year. This is done by filtering the airline dataframe based on the specified list of years using the `isin()` function and counting the number of rows in the resulting dataframe for each year.

Location: Code lines 83-97 in file `backend/part1/views.py`

Task 4: Display the timeliness percentage of flights for a given year. This task allows the user to specify a year and view the percentage of flights that departed on time, early, and late. The 'DepDelay' column is used to determine the timeliness of flights, with flights that departed on time having a 'DepDelay' of 0, flights that departed early having a 'DepDelay' less than 0, and flights that departed late having a 'DepDelay' greater than 0. Flight with an unknown departure delay are determined by subtracting the number of flights that departed on time, early, and late from the total number of flights. The percentage of flights in each category is then calculated and displayed to the user in the form of a pie chart.

Location: Code lines 115-149 in file `backend/part1/views.py`

Task 5: Display the top reason for cancelled flights for a given year. This task allows the user to specify a year and view the top reason for cancelled flights in that year. This is done by filtering the airline dataframe based on the specified year and creating a new dataframe that groups the data by the 'CancellationCode' column and counts the number of occurrences of each cancellation code. The cancellation code with the highest count is then displayed to the user.

Location: Code lines 163-193 in file `backend/part1/views.py`

Task 6: Display the top 3 most punctual airports for a given year This task allows the user to specify a year and view the top 3 airports with the most punctual flights in that year. This is done by filtering the airline dataframe based on the specified year and creating a new dataframe aggregates this to data to calculate the median departure delay (using the 'DepDelay' column) for each airport. The median is used here as it is less affected by extreme values than the mean, making it more robust to outliers. The top 3 airports with the lowest median departure delay are then displayed to the user.

Location: Code lines 207-238 in file `backend/part1/views.py`

Task 7: Display the top 3 worst performing airlines This task allows the user to view the top 3 worst performing airlines of the twentieth century based on a performance metric. The performance metric takes into account three factors:

- The percentage of flights that were delayed when departing.
- The percentage of flights that were delayed when arriving.
- The percentage of flights that were cancelled.

The dataframe is first filtered to include only flights that occurred in the twentieth century. The performance metric is then calculated for each airline by aggregating the data based on the 'DOT_ID_Reporting_Airline' column. Finally, a composite percentage score is calculated for each airline based on the three factors mentioned above by taking the average of the three percentages. The top 3 airlines with the highest composite percentage score are then displayed to the user.

Location: Code lines 247-268 in file `backend/part1/views.py`

3 Part 2: Intermediate Features

3.1 Feature Description

Task 1: Compare the performance of airports across the dataset TEMP

Location: Code lines 65-102 in file `backend/part2/views.py`

Task 2: Chart and analyze region and state performance TEMP

Location:

- Main logic: Code lines 118-145 in file `backend/part2/views.py`
- Single metric score logic: Code lines 16-57 in file `backend/part2/helper.py`

- Composite score logic: Code lines 72-116 in file `backend/part2/helper.py`
- Region score logic: Code lines 169-176 in file `backend/part2/helper.py`

3.2 Difficulties Encountered

4 Part 3: Advanced Features

4.1 Feature Description

4.2 Difficulties Encountered

5 Evaluation

6 Conclusion

Word Count: 225

References

[1] United States Department of Transport. *Download Page*. 2023. URL: `https://www.transtats.bts.gov/DL_SelectFields.asp?gnoyr_VQ=FGJ&Q0_fu146_anzr=b0-gvzr`.

[2] *Apache Parquet*. Apache Software Foundation. Accessed: 2024-03-23. 2024. URL: `https://parquet.apache.org/`.

[3] Sergey Melnik et al. "Dremel: Interactive Analysis of Web-Scale Datasets". In: *Proc. of the 36th Int'l Conf on Very Large Data Bases*. 2010, pp. 330–339. URL: `http://www.vldb2010.org/accept.htm`.

[4] *What is Parquet?* Databricks. URL: `https://www.databricks.com/glossary/what-is-parquet`.