Airline Flight Delay Data Analysis and Prediction

Thejas Mandya Shashidhara
tm4388
Dept. of ECE
tm4388@nyu.edu
Spring 2025

Ajay Venkatesh
av3855
Dept. of ECE
av3855@nyu.edu
Spring 2025

Ruchit Jathania
rj1376
Dept. of ECE
rj1376@nyu.edu
Spring 2025

Abstract

In this project, we intend to analyze the significant financial and operational challenges the aviation industry faces due to flight delays. Delays result from weather conditions, air traffic congestion, operational inefficiencies, and malfunctions. With millions of commercial flights occurring annually, traditional prediction methods for large-scale real-time data processing are inadequate. We leverage big data to build a machine learning model for flight delay prediction. Using historical flight records and air traffic information, an ML model is trained to predict delays before departure accurately. The system will utilize PySpark for data processing, Apache Spark for distributed computing, and Apache Hive for structured data storage and querying, running on HDFS. The project aims to produce an ML model to predict delay categories for flights trained on historical data.

1. Problem Statement and Objectives

Flight delays are a significant challenge in the aviation industry, causing financial losses, passenger dissatisfaction, and operational inefficiencies. Traditional delay prediction methods often fail to account for real-time influencing factors such as weather conditions, air traffic congestion, airline operations, and airport-specific delays.

With millions of flights recorded annually, a Big Data-driven approach is necessary to efficiently process and analyze vast datasets. This project will leverage Apache Spark, PySpark MLlib, and Hadoop HDFS to develop a scalable machine learning model for predicting flight delays.

Considering this scenario, we intend to create an analysis to answer.

1. Which airports have the highest average delays or the most frequent delays?
2. Which flight routes are most prone to delays?
3. Are there patterns in delays based on time of day, day of week, or season?
4. What are the primary causes of delays?
5. How does the weather impact flight delays?
6. Which airlines are more reliable or delay-prone?

The insights generated will help airlines optimize schedules, passengers make informed travel decisions, and airports improve resource management.

Technologies Used

- **Dataset**: BTS - Bureau of Transportation Statistics
- **Storage**: Hadoop HDFS
- **Processing**: Apache Spark, PySpark, Spark SQL
- **Machine Learning**: PySpark MLlib
- **Visualization**: Seaborn, Matplotlib, Plotly
- **Environment**: Jupyter Notebook/Google Colab, Dataproc

2 Architecture

The architecture of this project is designed to efficiently process and analyze large-scale flight delay data using a robust Big Data pipeline. Historical flight data from the Bureau of Transportation Statistics (BTS) is ingested and stored in Hadoop HDFS, enabling distributed and scalable storage. PySpark performs data cleaning, transformation, and feature engineering, while Spark SQL and Hive are used for querying and managing preprocessed datasets. Machine learning models are developed using PySpark MLlib, where features are vectorized, and classifiers such as Decision Trees or Random Forests are trained to categorize flights into delay classes, including no delay, delay $\leq$ 30 minutes, delay $>$ 30 minutes, and cancellations. The results are then analyzed and visualized using Seaborn, Matplotlib, and Plotly to uncover trends such as delay patterns across routes, times, and weather conditions. This end-to-end architecture supports high-performance analytics and delivers actionable insights for operational improvements in the aviation industry.

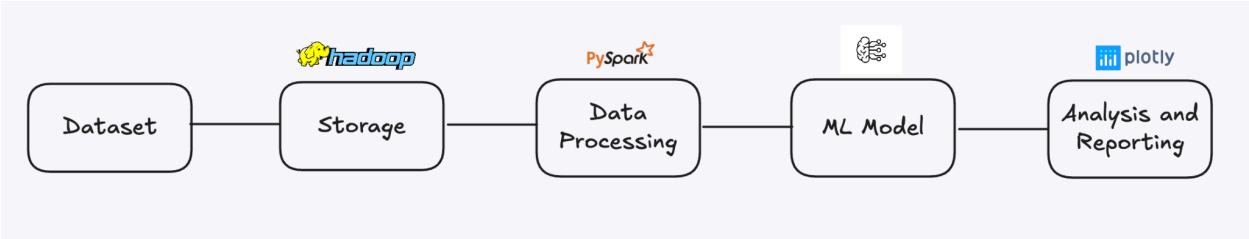


Figure 1: Architecture

3 Methodology

Considering the Problem Statement and the available resources, this project employs a Big Data-driven methodology to analyze and predict flight delays using scalable tools. The key steps in the method are outlined below:

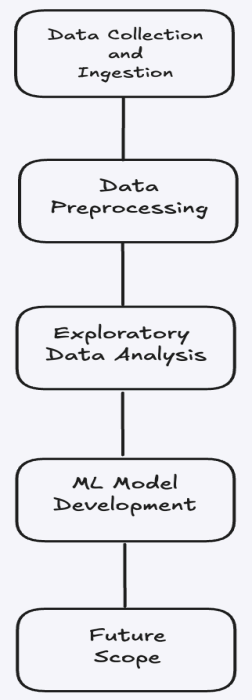


Figure 2: Flow of the Project

3.1 Data Collection and Ingestion

Flight Data is sourced from the Bureau of Transportation Statistics (BTS) and ingested into the Hadoop Distributed File System (HDFS) for distributed storage. The data covers all the airline details for 2024, including the airline name, origin, arrival time, departure time, destination, flight number, and factors of airline delay. This dataset contains 18 columns and 58 lakh rows.

3.2 Data Preprocessing

For the data preprocessing, we leveraged PySpark, and the raw flight datasets underwent a series of transformations to prepare them for data analysis and modeling. Initially, null values and duplicate records are identified and removed to ensure data quality. Additional features are engineered to capture meaningful patterns, such as extracting arrival and departure times, the hour of the day, and the day of the week. A crucial part of preprocessing is the creation of the target variable, which classifies each flight into one of four categories: No Delay, Delay less than or equal to 30 minutes, Delay greater than 30 minutes, and Cancelled. These preprocessing steps lay the foundation for practical exploratory analysis and machine learning model development.

3.3 Exploratory Data Analysis

In the Exploratory Data Analysis (EDA) phase, Spark SQL is employed to uncover patterns and trends in flight delay data across various dimensions. Through aggregated queries, airports and flight routes with the highest frequency or severity of delays are identified, helping pinpoint operational bottlenecks. Temporal analysis examines how delays vary by time of day, day of the week, and season, revealing cyclical trends or peak congestion periods. Additionally, airline-specific performance is assessed to determine which carriers are more prone to delays, and the impact of weather conditions is evaluated to understand external contributing factors. The insights derived from these analyses are visualized using Seaborn, Matplotlib, and plotly, enabling intuitive and impactful communication of findings to stakeholders.

3.4 Machine Learning Model Development

The processed and feature-engineered flight data is used to build a predictive model that classifies flight delays into four categories: No Delay, Delay less than or equal to 30 minutes, Delay greater than or equal to 30 minutes, and Cancelled. Using PySpark's MLlib, a scalable machine learning pipeline is created to efficiently handle large volumes of data. Relevant features such as origin, destination, scheduled departure time, airline, weather conditions, and day of the week are selected and transformed into a numerical format using one-hot encoding and vector assembly. The dataset is split into training and testing sets, and classification models such as Linear Regression and Random Forests are trained on the data. The model's performance is evaluated using accuracy, precision, recall, and F1-score metrics to ensure reliability. Once the model achieves satisfactory results, it generates predictions, providing actionable insights to help airlines and airports proactively anticipate and manage delays.

4 Data Sources and Results

The dataset, sourced from the Bureau of Transportation Statistics, contains detailed records of U.S. domestic flights for the year 2024. Using PySpark for large-scale data processing, we cleaned and transformed millions of flight records and conducted extensive exploratory data analysis (EDA) to uncover delay patterns by time, airline, and location. The processed data was used to train multiple classification models, and the outcomes were evaluated using confusion matrices.

4.1 Geospatial Analysis of Flight Delays

1. Top Delayed U.S. Airports

This visualization presents U.S. airports plotted on a map, where:

- Circle size indicates the overall traffic volume (number of flights).
- Circle color (from yellow to red) indicates the normalized average delay, with redder circles denoting higher delays.

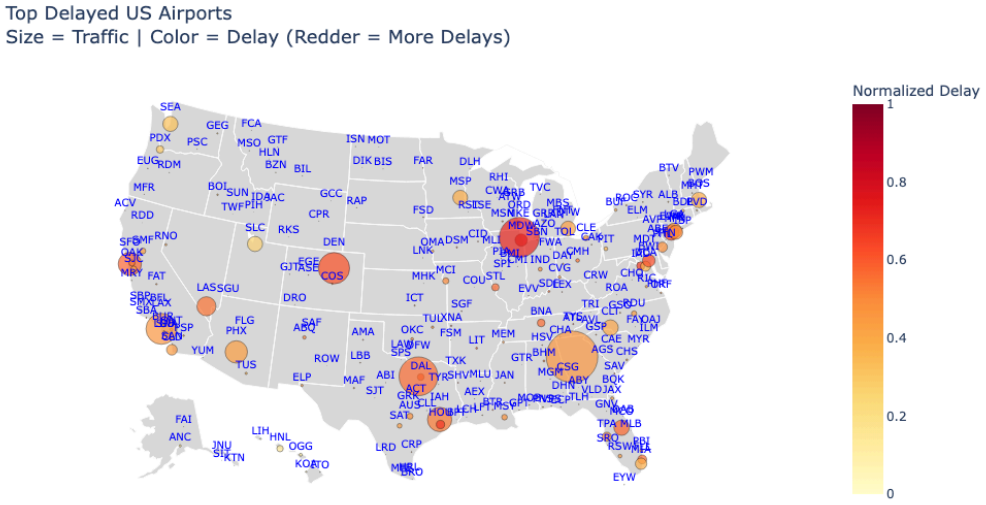


Figure 3: Top Delayed U.S. Airports

2. U.S. Flight Delay Routes Visualization

This map highlights flight route-level delays, with markers representing the severity of delay on specific connections:

- **Darker and larger dots** represent routes with higher normalized delay values.
- **Yellow to red** gradient shows escalating delay from low to high.

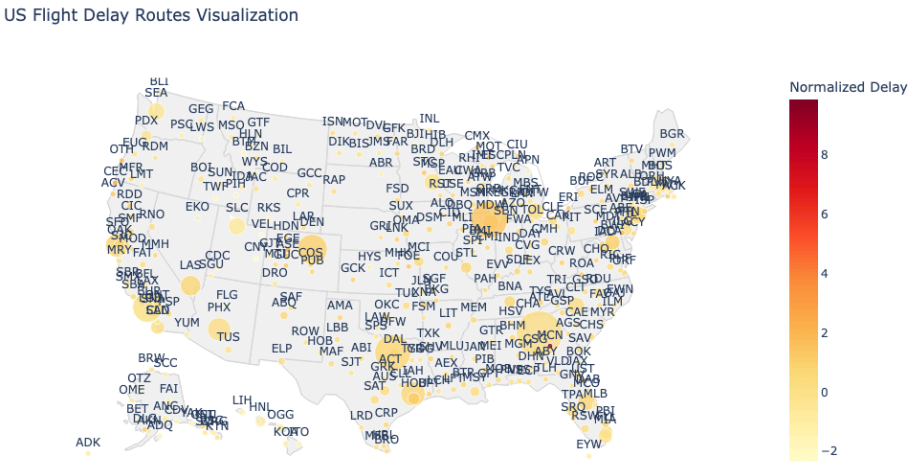


Figure 4 U.S. Flight Delay Routes Visualization

4.2 Monthly Delay Patterns by Carrier

The bar chart shows the average delay per month for three major carriers: AA, UA, and US.

Key observations include:

- Carrier AA exhibited the highest delays, especially in June and October, where delays peaked above 120 minutes.
- Carrier US maintained a more consistent delay profile across the year, but still faced spikes in February and August.
- UA generally had the lowest delays across all months, indicating more reliable schedule adherence.

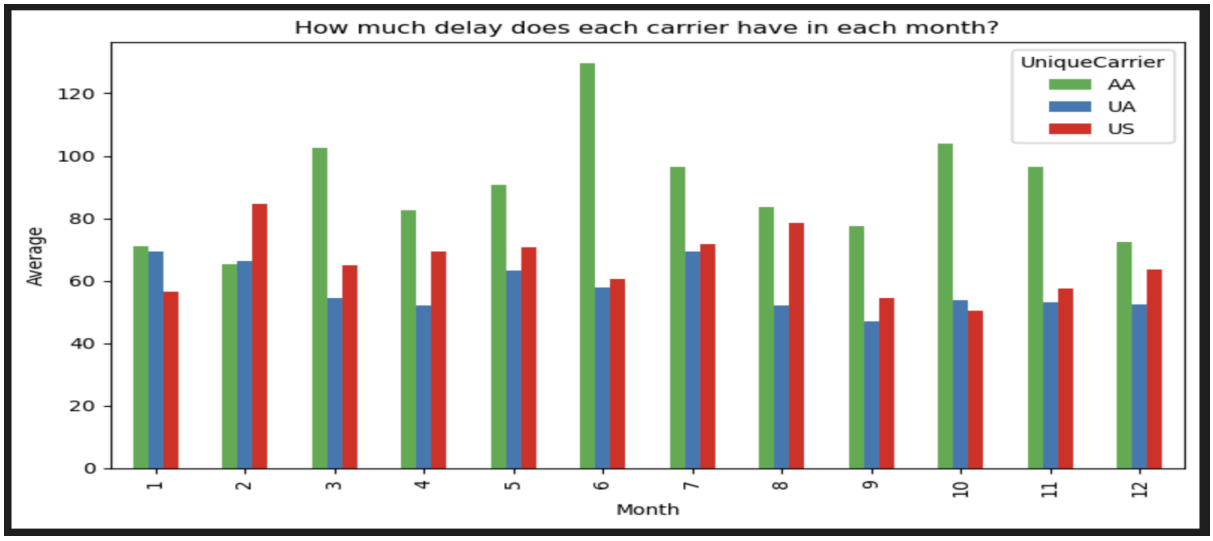


Figure 5: Monthly Delay Patterns by Carrier

4.3 Delay Patterns by Hour and Weekday

A heatmap was generated to examine the average delay per hour of the day across weekdays:

- High delays were observed early morning on Fridays (Day 5) and late-night Sundays (Hour 23).
- Mid-day and mid-week flights (Tuesdays to Thursdays) showed relatively lower delays.
- These insights suggest that flight scheduling and traffic density at certain times contribute to delays.

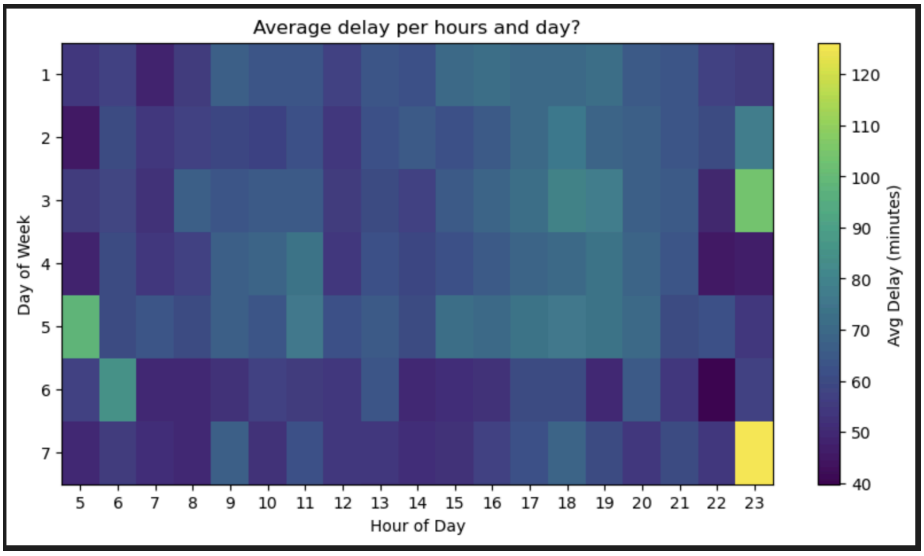


Figure 6: Average Delay per hours and day

4.4 Model Performance Comparison

We evaluated four classifiers to predict delay categories:

1. **Logistic Regression:**

- Achieved good accuracy for the No Delay class but struggled with short and long delays, misclassifying many ≤30min delays as No Delay.
- Indicates linear boundaries are insufficient for complex delay distributions.

2. **Decision Tree:**

- Improved precision for all classes compared to Logistic Regression.
- Captured non-linear relationships, with better separation between ≤30min and >30min delays.

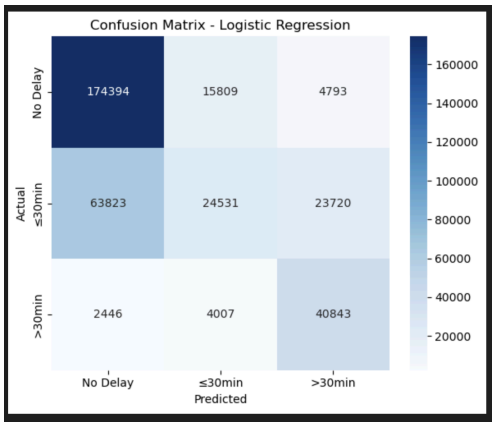
3. **Multilayer Perceptron(MLP):**

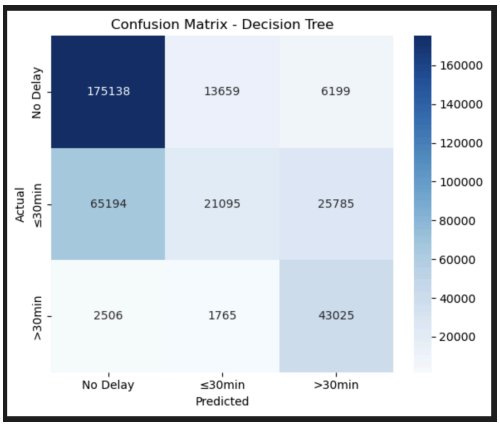
- Overfitted heavily to the No Delay class, predicting all test samples as No Delay, indicating poor generalization or incorrect label encoding.

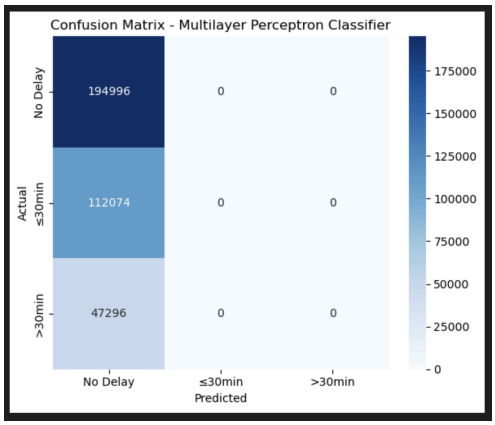
4. **Random Forest:**

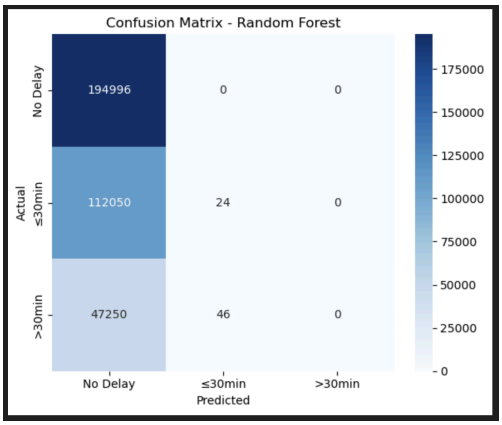
- Similar to MLP, but slightly better, with minimal prediction in ≤30min and >30min classes.
- Likely suffered from class imbalance or lack of tuning.

here , are the confusion matrix achieved by all the machine learning models









Conclusion:

Decision Tree performed best among tested models for multi-class delay classification.

5 Technological Challenges

During the development of this flight delay prediction system, we encountered several technical hurdles:

- **Handling Large Datasets Efficiently**: The raw BTS flight datasets were massive in size, often exceeding system memory limits when handled with traditional tools. We used **Hadoop HDFS** for distributed storage and **PySpark** to process data in parallel, which significantly improved performance and scalability.

- **Data Quality and Cleaning**: The datasets contained numerous null values, inconsistent date/time formats, and duplicate records. Implemented PySpark-based ETL pipelines that automatically filtered out invalid rows, standardized timestamps, and filled missing values with appropriate defaults or indicators.

- **Feature Engineering Complexity**: Extracting meaningful features such as delay buckets, time-of-day, and seasonal indicators required careful transformation of temporal data. Used PySpark's withColumn, hour(), dayofweek(), and month() functions to systematically generate new features while retaining pipeline performance.

- **Model Tuning in a Distributed Setup**: Tuning machine learning models in MLlib required understanding Spark's parallel execution model and the constraints of its APIs. Began with simpler models (Decision Tree) to validate pipeline correctness before moving to more complex classifiers and evaluating them using MLlib's built-in metrics.

- **Visualization Integration**: Combining PySpark with Python visualization tools (like Seaborn, Plotly) was tricky due to the need to convert Spark DataFrames to Pandas.
  Used .toPandas() after filtering and sampling Spark DataFrames to generate clear, performance-friendly plots.

6 Lessons Learned

- Big Data Tools are Essential for Scalability: Using PySpark and HDFS enabled us to process gigabytes of flight data that would otherwise be unmanageable with Pandas or Excel.
- Data Preprocessing is Critical: The success of machine learning models heavily depended on robust feature engineering and data cleaning—more than the choice of algorithm itself.
- Start Simple, Then Scale: Beginning with small subsets of data and simple models helped us debug the pipeline early and iterate faster before scaling to the full dataset.
- Visualization Bridges the Gap: Tools like Plotly and Seaborn were vital in making the data and model outputs interpretable, especially for stakeholders unfamiliar with code.
- Collaboration and Versioning Help: Using GitHub for code versioning and modular script development helped streamline collaboration and made the workflow more reproducible.

7 Future Scope

- Real-time Data Integration: Incorporate live weather, air traffic, and flight tracking data for real-time delay predictions
- Advanced ML models: Integrating the real-time data traffic with deep learning models such as LSTM, XGBoost to improve prediction accuracy.
- Scalability: developing and extending the model to handle international and multi-leg flight data.
- Integration with airport systems: enabling interoperability with airport resource planning and traffic control systems.
- Delay Cause Analysis: Enhancing the model to predict delays and explain contributing factors such as weather, crew, or mechanical issues.

8 References

- **Link** to the dataset
- Guide to Spark - ML
- Spark - SQL
- Library - Plotly
- https://ieeexplore.ieee.org/document/9994427

- https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-dataset-2019-2023/data?select=flights_sample_3m.csv

9 Code Execution Instructions

- Create your 2024 project folder on Dataproc (if not already):

```
mkdir -p ~/flight_project_2024
```

- Transfer all 2024_*.csv files + airports.dat from your Mac to the VM, From you local terminal, run the following command, (change your local path)

```
gcloud compute scp ~/Desktop/Subjects/Sem2/Big\ Data/Final\
Project/dataset/2024_*.csv nyu-dataproc-m:~/flight_project_2024/
--zone=us-central1-f

gcloud compute scp ~/Desktop/Subjects/Sem2/Big\ Data/Final\
Project/dataset/airports.dat nyu-dataproc-m:~/flight_project_2024/
--zone=us-central1-f
```

- SSH into the VM and prepare the full 2024 CSV

```
cd ~/flight_project_2024
head -n 1 2024_1.csv > full_2024.csv
tail -n +2 -q 2024_*.csv >> full_2024.csv
```

- Upload to HDFS

```
hdfs dfs -mkdir -p /user/your_path/flight_data_2024
hdfs dfs -put -f full_2024.csv /user/your_path/flight_data_2024/
```

- Launch Jupyter Notebook inside folder:

```
cd ~/flight_project_2024
jupyter notebook --no-browser --port=8888
```

- Then go to http://localhost:8889 on your browser (from your Mac) and paste the token from your terminal.
- Clone the github repository (link) and upload the jupyter notebook to the folder and then run the notebook. (use the same command used to upload files)