Building a Scalable Pipeline for Pre-Flight Delay Prediction

1. Introduction and Objectives

Flight delays significantly impact passenger satisfaction, airline operational costs, and airport management. Accurately predicting delays enables proactive measures such as resource reallocation, passenger notifications, and improved scheduling.This project aims to build a scalable, cloud-based machine learning pipeline to predict flight arrival delays using large-scale flight data. The objectives include:

- Leveraging PySpark for distributed data processing on big datasets.
- Implementing a Random Forest classifier to capture non-linear relationships.
- Utilizing AWS services (S3, Athena) for efficient data storage and querying.
- Delivering a pipeline capable of training and scoring on millions of flight records.
- Producing actionable insights through feature importance and model evaluation metrics.

2. Dataset Overview

The dataset consists of airline flight records with detailed delay causes, covering multiple years and millions of entries.

Data Source: Raw CSV files stored on AWS S3 at `s3://us-flight-delay-data/raw/Airline_Delay_Cause.csv`.

Storage Format: Converted to Parquet for efficient querying via AWS Athena.

2.1 Key Variables:

Features were categorized based on when they are known—either before the flight takes off or only after it has landed. This distinction is critical for building a predictive model.

Pre-Flight Features (Predictors): These are known before a flight is scheduled to depart and are used for prediction.

Column	Description
month	Month of the flight (1–12)
carrier	Airline carrier code
Route	Flight route identifier (e.g., JFK-LAX)

Post-Flight Information (Leakage Risks): These are known only after a flight has landed. They describe the outcome and cause of a delay and must be excluded from a predictive model to avoid label leakage.

Column	Description
Delay Columns	arr_delay, carrier_delay, weather_delay, nas_delay, security_delay, late_aircraft_delay
Delayed_Arrival	Binary target: 1 if flight delayed, 0 if on-time

Data Size:Millions of records(Jan 2020 – Feb 2025), requiring distributed processing.

3. Data Preprocessing and Transformation

Handling such a large and complex dataset required careful preprocessing:

- **Type Conversion:** Cast delay columns from string to numeric, replacing invalid or missing values with zeros or null-safe defaults.
- **Handling Missing Values:** Used PySpark functions to fill or ignore nulls appropriately.

Categorical Feature Encoding:

- Applied `StringIndexer` to `carrier` and `Route` columns to convert categorical strings into numeric indices.
- Utilized `handleInvalid="keep"` to gracefully handle unseen or invalid categories in test data.

Feature Assembly:

Selected delay-related numerical features and encoded categorical indices.

Combined all features into a single vector column using `VectorAssembler`.

4. Exploratory Data Analysis (EDA)

Delay Distribution: Majority of delays stem from `late_aircraft_delay`, `nas_delay` (National Airspace System), and `carrier_delay`.

Class Distribution: The dataset exhibits class imbalance, with delayed flights outnumbering on-time flights.

Correlation Analysis:

- arr_delay` strongly correlated with delay status, indicating possible label leakage.
- Other delay cause variables showed moderate correlations with the target.

Insights: Delay causes vary by route and carrier

5. Machine Learning Model Development

Model Choice:RandomForestClassifier from PySpark MLlib chosen for:

- Ability to handle large feature sets and data volumes.
- Robustness to feature correlations and missing values.
- Interpretability through feature importance metrics.

Pipeline Construction:

1. Categorical Encoding (`StringIndexer` for `carrier`, `Route`).

2. Feature Vector Assembly (`VectorAssembler`).

3. Random Forest Model Training.

Model Parameters:

- Label column: `Delayed_Arrival`.
- Features column: `features`.
- maxBins` set to 400 for handling high-cardinality categorical features.
- Number of trees and max depth optimized through experimentation.

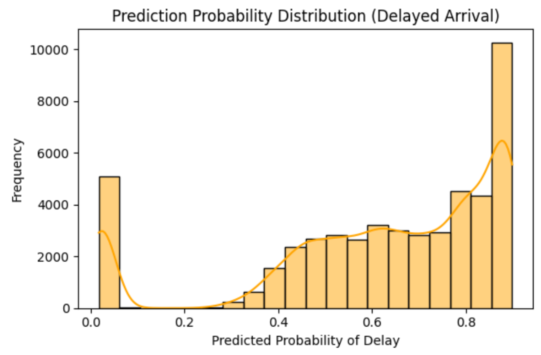
6. Model Evaluation and Results

Metric	Value
Accuracy	0.780
Precision (Delayed)	0.772
Recall (Delayed)	0.920
F1 Score	0.840

Confusion Matrix:

	Predicted On-Time (0)	**Predicted Delayed (1)**
Actual On-Time (0)	10,051	8,330
Actual Delayed (1)	2,455	28,258

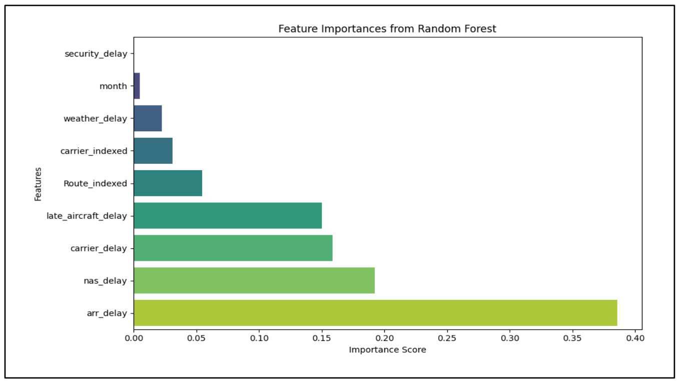
Interpretation: The model excels at detecting delayed flights (high recall), with some false positives in on-time flights predicted as delayed.



Operational Implication of False Positives
While the model shows some false positives—predicting delays for flights that were actually on time—this can still be beneficial operationally. In flight management, it is often safer to overestimate delays rather than underestimate them. Early warning systems help passengers prepare for potential disruptions and allow airlines to proactively adjust crew scheduling, gate assignments, or resource allocations. The cost of a false alert is significantly lower than the consequences of a missed delay prediction, making high recall a valuable trait for this application.

Top 5 Feature Importances:

Feature	importance
Arr delay	39.0
nas delay	19.1
Late_aircraft delay	16.1
Carrier delay	15.0
Route indexed	4.8



7. Cloud-Based Architecture Diagram

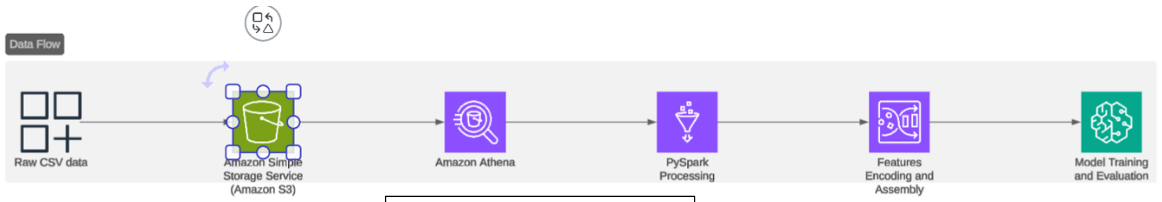

Data Flow
Raw CSV data
Amazon Simple Storage Service (Amazon S3)
Amazon Athena
PySpark Processing
Features Encoding and Assembly
Model Training and Evaluation

Figure: Data Flow Overview

This diagram outlines the core data flow of the pipeline from raw data ingestion in S3 to model training and evaluation highlighting the sequential transformation and processing steps.

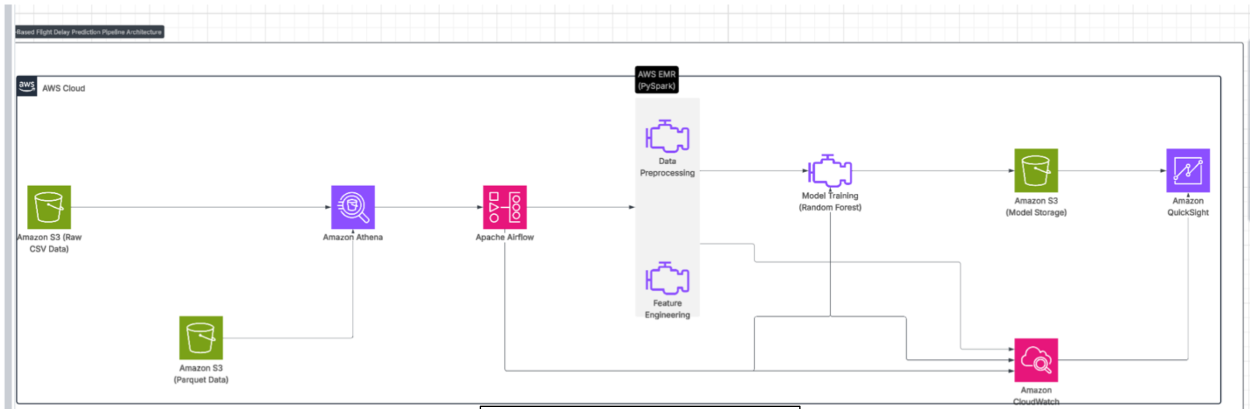


Figure: Full Production-Ready Pipeline

This diagram shows the complete pipeline with orchestration using Apache Airflow. It automates daily runs of data ingestion, processing with PySpark on EMR, model training, and batch predictions, ensuring scalability and continuous performance monitoring.

8. Challenges and Solutions

Challenges	solution
Class Imbalance	Used precision, recall, and F1-score to evaluate beyond accuracy. Also plan to apply SMOTE or undersampling.
Label Leakage (from `arr_delay`)	Identified and flagged for exclusion in future iterations to improve model robustness.
High Cardinality in Categorical Variables	Set `maxBins=400` in Random Forest; used `handleInvalid="keep"` in `StringIndexer`.
Large Dataset Size	Utilized distributed processing with PySpark and AWS cloud infrastructure.

9. Conclusion and Future Work

This project successfully developed a scalable, distributed flight delay prediction model with strong performance, especially high recall for identifying delays, which is critical for operational responsiveness.

Future Enhancements:

- Label Leakage Mitigation: Exclude `arr_delay` and related direct delay indicators to prevent target leakage.
- Data Balancing:Implement SMOTE or undersampling to address class imbalance.
- Model Tuning:Use cross-validation and hyperparameter optimization (e.g., `CrossValidator` in PySpark).
- Alternative Models:Experiment with gradient boosting algorithms such as XGBoost or LightGBM for potentially higher accuracy.
- Real-Time Pipeline: Integrate AWS Lambda and QuickSight for real-time delay prediction and visualization dashboards.
- Feature Engineering: Incorporate weather, air traffic control, and other external data sources.