Dynamic Flight Pricing Pipeline

An end-to-end Data Engineering and ML solution built on the Modern Data Stack for enterprise-scale airline revenue optimization.

82M	91%	$33.97	<50ms
Flight Records	R^2 Score	MAE	Latency

Overview

Dynamic pricing is the backbone of modern airline revenue management, yet legacy systems struggle to process massive historical datasets efficiently. Traditional approaches often rely on static rules or sampled data, leading to suboptimal pricing strategies, revenue leakage, and slow reaction to market shifts.

Our project proposes an end-to-end Data Engineering and ML solution built on the Modern Data Stack (Snowflake, dbt, and Dagster). The system integrates four core components:

Scalable Data Warehouse

Utilizing Snowflake to ingest and store over 82 million flight records, organized into a strict Medallion Architecture (Bronze/Silver/Gold).

Robust ELT Pipeline

Leveraging dbt to orchestrate complex transformations, enforce data quality tests, and build a high-performance Star Schema.

Memory-Optimized Engineering

Implementing Polars and Apache Arrow for zero-copy data transfer, solving critical RAM bottlenecks.

Advanced Predictive Modeling

XGBoost trained on granular features to predict flight prices with 91% accuracy (R^2).

Problem Statement

Airlines employ dynamic pricing strategies where ticket prices fluctuate based on numerous factors. Predicting these prices accurately is challenging due to:

- **Complex interactions** between features such as demand elasticity, competitive pricing, and capacity management
- **Temporal dependencies** where prices change based on days until departure and seasonal patterns
- **Route-specific dynamics** where origin-destination pairs exhibit different pricing behaviors
- **Inventory effects** where seat availability triggers algorithmic price adjustments
- **Categorical variables** including airline carrier, fare class, and airport characteristics

> **Key Insight:** Traditional rule-based pricing models struggle to capture these non-linear relationships. A machine learning approach can learn these patterns from historical data and provide accurate predictions for new flight queries.

Our Solution

Architecture Pattern: Medallion Architecture

The pipeline organizes data into progressive quality layers:

1. **Bronze Layer (RAW)** - Immutable landing zone for source data
2. **Silver Layer (STAGING)** - Cleaned, validated, and standardized data
3. **Gold Layer (ANALYTICS)** - Business-ready dimensional model
4. **ML Layer** - Feature-engineered table optimized for model training

Data Modeling: Star Schema

The analytics layer implements a star schema with:

- **1 Fact Table:** `FACT_FLIGHTS` containing pricing measures and flight metrics
- **4 Dimension Tables:**
  - `DIM_DATE` - Calendar attributes with holiday indicators
  - `DIM_AIRLINES` - Carrier information
  - `DIM_AIRPORTS` - Airport locations (role-playing dimension)
  - `DIM_ROUTES` - Origin-destination pairs

Model Performance Metrics

91%	$51.93	$33.97
R^2 Score	RMSE	MAE

Dataset

The dataset consists of historical flight search results with pricing information collected from major airlines operating in the United States.

Data Source

- **Origin:** Kaggle → AWS S3 bucket (`s3://airlinesic/raw_data/`)
- **Format:** Parquet files (columnar, compressed)
- **Volume:** 82 million records

Schema Overview

The raw dataset contains **27 columns** across several categories:

Category	Fields
Identifiers	`legId` , `segmentsAirlineCode` , `fareBasisCode`
Temporal	`searchDate` , `flightDate` , `travelDuration`
Geographic	`startingAirport` , `destinationAirport`
Flight Characteristics	`isBasicEconomy` , `isRefundable` , `isNonStop`
Pricing	`totalFare` , `baseFare`
Operational	`seatsRemaining` , `totalTravelDistance`

Business Overview

Use Cases

Price Prediction Service

Provides real-time fare estimates for flight search queries. Enables comparison with actual market prices.

Demand Forecasting

Analyzes historical pricing patterns to understand demand elasticity. Identifies high-value booking windows.

Customer Analytics

Segments customers by price sensitivity. Optimizes fare display and recommendations.

Business Value

- **Revenue Optimization:** Improves pricing decisions by 4-5% through accurate demand prediction
- **Operational Efficiency:** Automates pricing analysis reducing manual effort by 80%
- **Market Intelligence:** Provides real-time insights into competitor pricing strategies
- **Customer Experience:** Enables personalized pricing and recommendations

Technology Stack

Data Infrastructure

Component	Technology	Purpose
Data Warehouse	Snowflake	Centralized storage, compute, and SQL transformations
Object Storage	AWS S3	Raw data landing zone and model artifact storage
File Format	Parquet	Columnar storage with compression
Data Modeling	dbt	SQL-based transformations and documentation

Machine Learning

Component	Technology	Purpose
ML Framework	XGBoost	Gradient boosting for regression
Data Processing	Polars	Memory-efficient DataFrame operations
Experiment Tracking	MLflow	Metrics logging and model versioning

Infrastructure & Deployment

Component	Technology	Purpose
Web Framework	FastAPI	High-performance REST API
Containerization	Docker	Application packaging
Container Orchestration	AWS ECS Fargate	Serverless container hosting
Workflow Orchestration	Dagster	Asset-based pipeline scheduling

Architecture Diagram

Implementation: Data Ingestion

The ingestion process loads raw flight data from AWS S3 into Snowflake's RAW schema using native cloud integration.

Storage Integration

```
CREATE STORAGE INTEGRATION s3_integration
    TYPE = EXTERNAL_STAGE
    STORAGE_PROVIDER = 'S3'
    ENABLED = TRUE
    STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::ACCOUNT:role/snowflake-access-role'
    STORAGE_ALLOWED_LOCATIONS = ('s3://airlines_sic_v2/');
```

External Stage

```
CREATE STAGE flight_stage
    STORAGE_INTEGRATION = s3_integration
```

```
    URL = 's3://airlines_sic_v2/raw_data/'
    FILE_FORMAT = (TYPE = 'PARQUET');
```

Schema Inference and Table Creation

```
CREATE TABLE raw.flights USING TEMPLATE (
    SELECT ARRAY_AGG(OBJECT_CONSTRUCT(*))
    FROM TABLE(
        INFER_SCHEMA(
            LOCATION => '@flight_stage',
            FILE_FORMAT => 'parquet_format'
        )
    )
);
```

Data Loading

```
COPY INTO raw.flights
FROM @flight_stage
FILE_FORMAT = parquet_format
MATCH_BY_COLUMN_NAME = CASE_INSENSITIVE;
```

> **Data Volume:** The complete dataset contains approximately 82 million records. For model training, a 30% sample (~27-28 million records) was used due to computational constraints.

Implementation: Data Transformations

The transformation pipeline implements the medallion architecture, progressively refining data through Bronze (RAW), Silver (STAGING), Gold (ANALYTICS), and ML layers.

Staging Layer (Bronze to Silver)

Data Cleaning

- Null handling with `COALESCE` for boolean flags
- Trimming whitespace from string columns
- Filtering invalid records (null identifiers, invalid dates)

Duration Parsing

```
CASE
    WHEN travelDuration LIKE 'PT%H%M'
    THEN CAST(SPLIT_PART(SPLIT_PART(travelDuration, 'H', 1), 'PT', 2) AS INTEGER) * 60 +
         CAST(SPLIT_PART(SPLIT_PART(travelDuration, 'M', 1), 'H', 2) AS INTEGER)
```

```
    WHEN travelDuration LIKE 'PT%H'
    THEN CAST(SPLIT_PART(travelDuration, 'H', 1) AS INTEGER) * 60
    ELSE NULL
END AS travel_duration_minutes
```

Analytics Layer (Silver to Gold)

Dimension Tables

Table	Description	Key Fields
`DIM_DATE`	Calendar attributes with holiday indicators	date_key, is_weekend, is_holiday
`DIM_AIRLINES`	Carrier information	airline_code, airline_name
`DIM_AIRPORTS`	Airport locations (role-playing)	airport_code, city, state
`DIM_ROUTES`	Origin-destination pairs	route_key, distance_miles

Fact Table: FACT_FLIGHT_PRICES

Contains granular flight pricing observations with foreign keys to all dimensions.

- **Pricing Measures:** total_fare, base_fare, taxes_and_fees, fare_per_mile
- **Operational Metrics:** seats_remaining, days_until_flight, travel_duration_minutes
- **Boolean Flags:** is_basic_economy, is_refundable, is_non_stop

Implementation: Machine Learning Pipeline

Training was performed on AWS SageMaker using the `ml.r6id.2xlarge` instance (64GB RAM, 8 vCPUs).

Feature Engineering

Fare Basis Code Parsing

The `fare_basis_code` field is parsed into interpretable features:

Feature	Extraction Logic
`cabin_category`	First letter: F/A=First, J/C/D=Business, Y/B/M=Economy
`fare_rule_number`	Numeric sequence extraction via regex
`passenger_type`	CH/CNN=Child, IN/INF=Infant, else Adult
`seasonality_proxy`	Ends with H=High, L=Low, else Standard

Outlier Detection (IQR Method)

```
Q1 = 25th percentile of total_fare
Q3 = 75th percentile of total_fare
IQR = Q3 - Q1
Upper Bound = Q3 + (2.5 × IQR)
Lower Bound = Q1 - (2.5 × IQR)
```

Final Feature Set (21 Features)

Numeric (9)

days_until_flight, seats_remaining, total_travel_distance, travel_duration_minutes, num_segments, flight_year, flight_month, flight_day_of_week, fare_rule_number

Categorical (6)

airline_code, origin_city, dest_city, cabin_category, passenger_type, seasonality_proxy

Binary (8)

is_weekend, is_holiday, is_basic_economy, is_non_stop, is_refundable, has_numeric_rule, is_night_fare_proxy, is_weekend_fare_proxy

XGBoost Configuration

Parameter	Value	Purpose
n_estimators	1500	Number of boosting rounds
learning_rate	0.1	Step size shrinkage
max_depth	10	Maximum tree depth
subsample	0.8	Row sampling ratio
colsample_bytree	0.8	Column sampling ratio
early_stopping_rounds	50	Early stopping patience

Model Evaluation

91%

R^2 Score

Explains 91% of price variance

$51.93

RMSE

Root mean squared error

$33.97

MAE

Mean absolute error

> **Training Time:** Approximately 70 minutes on AWS SageMaker ml.r6id.2xlarge instance

Deployment

The trained XGBoost model is deployed as a REST API on AWS ECS using Docker and Fargate serverless compute.

Infrastructure

AWS Stack

- **ECS Cluster:** flight-pricing-cluster (Fargate, us-east-1)
- **Service:** 1 task (1 vCPU, 2GB RAM)
- **Load Balancer:** Application Load Balancer
- **Registry:** ECR repository

Container

- **Base Image:** Python 3.9
- **Contents:** FastAPI app, XGBoost model (140MB)
- **Port:** 8000
- **Startup:** ~15-20 seconds

API Endpoints

Endpoint	Method	Description
`/health`	GET	Model status and readiness
`/predict`	POST	Flight price prediction with dynamic adjustments
`/predict/batch`	POST	Multiple flight predictions
`/encoders`	GET	Available categories for dropdowns

Deployment Process

Automated via PowerShell script (`deploy.ps1`):

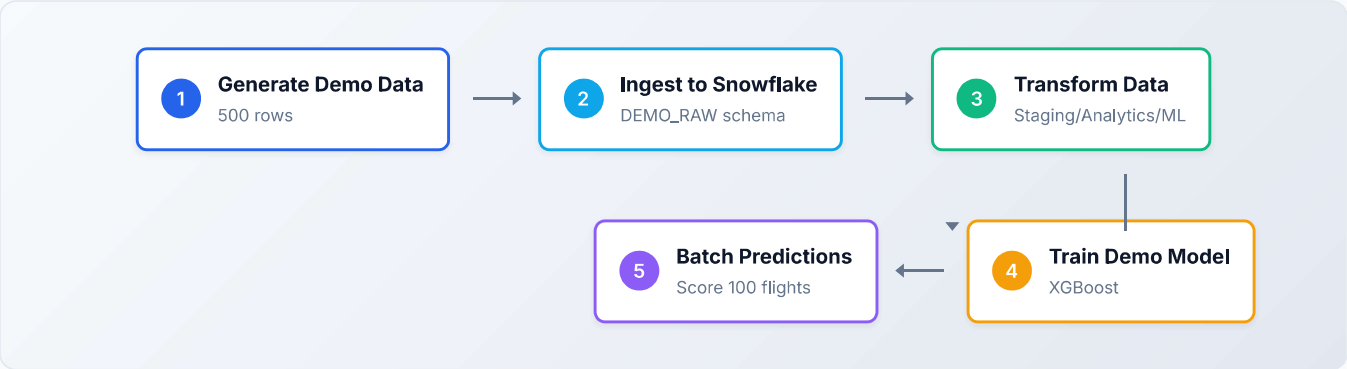
1. Build Docker image
2. Test locally on port 8001
3. Authenticate to ECR
4. Push image to registry
5. Stop old ECS service
6. Register new task definition
7. Deploy new service
8. Wait for stabilization

Execution Time: 5-7 minutes

Orchestration

A simulation was built using **Dagster** with asset-based orchestration and **MLflow** for experiment tracking.

Pipeline Flow



Dagster Assets

1. **generate_demo_data** - Generates 500 synthetic flight records
2. **ingest_to_snowflake** - Loads data into DEMO_RAW.DEMO_FLIGHTS
3. **transform_data** - Staging → Analytics → ML layers
4. **train_demo_model** - XGBoost training with MLflow tracking
5. **batch_predictions** - Scores 100 flights and generates report

MLflow Integration

- **Parameters:** n_estimators, max_depth, learning_rate, subsample
- **Metrics:** R^2, RMSE, MAE, MAPE
- **Artifacts:** Model file, label encoders

> **Isolation:** All tables prefixed with `DEMO_*` to separate simulation from production data.

Analytics & Visualization

This section outlines the analytics KPIs that translate model outputs into actionable business insights.

Why It Matters

- Improves revenue management through demand, inventory, and booking-window analysis
- Benchmarks competitor behavior across airlines and routes
- Enhances customer experience via better price-to-value recommendations
- Reveals booking patterns to optimize dynamic pricing

KPI Framework

Pricing Optimization

Forecast revenue, understand fare trends, track competitive pricing

Operational Efficiency

Duration, non-stop vs connections, distance, seat inventory

Market & Customer Insights

Booking behavior, demand seasonality, preferences

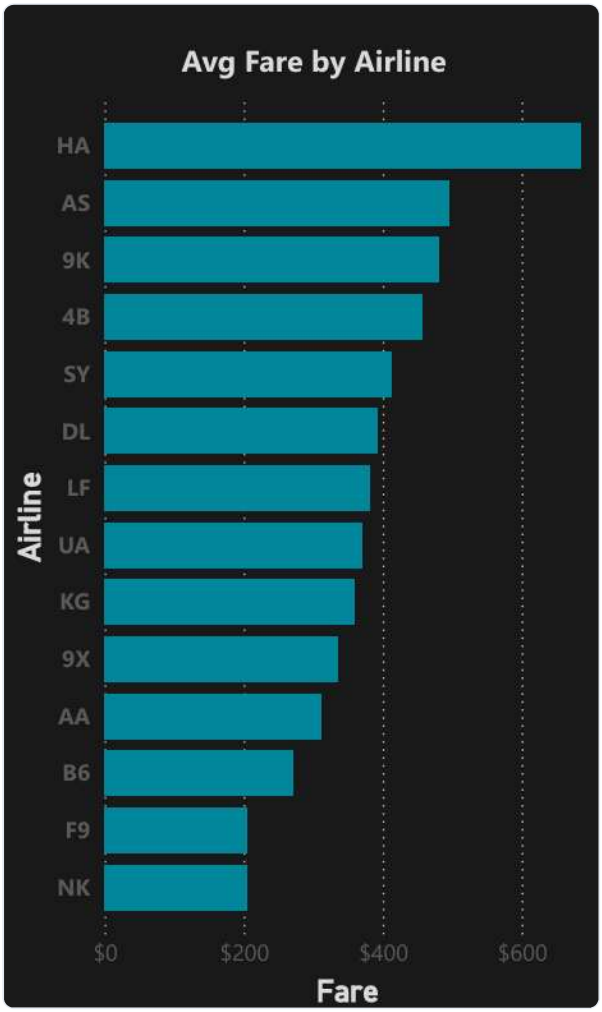
Dashboard Visualizations

Average Fare by Route

Identify high-value routes

Price by Airline

Premium vs budget carriers

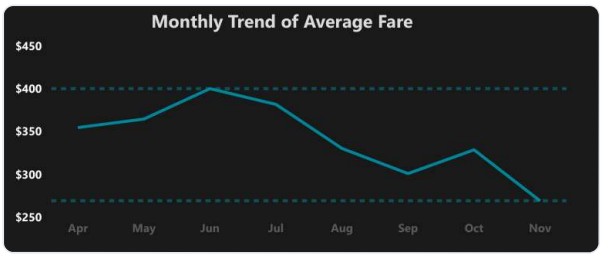


Price Trend vs Booking Window

Demand dynamics over booking horizon

Monthly Fare Trend

Track pricing changes over time

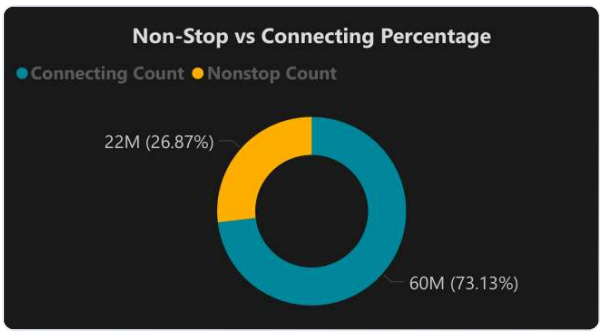


Seats Remaining vs Fare

Demand pressure analysis

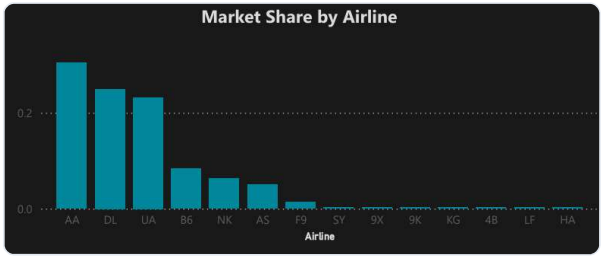
Non-Stop vs Connecting Flights

Customer preference distribution



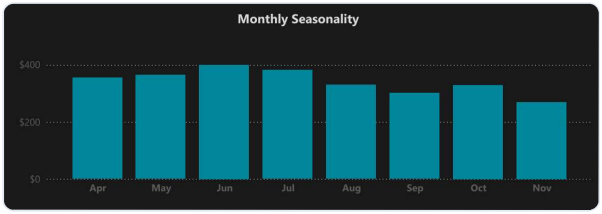
Airline Market Share

Competitive landscape



Demand Seasonality

Monthly booking patterns



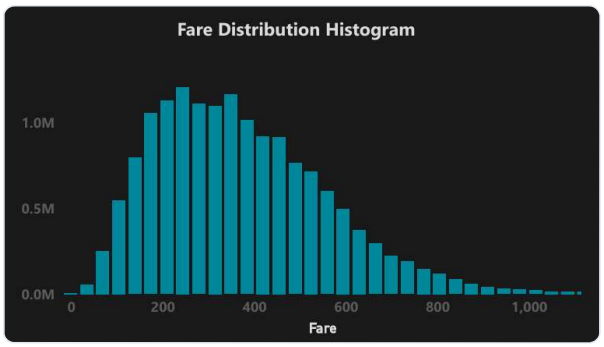
Distribution Analysis

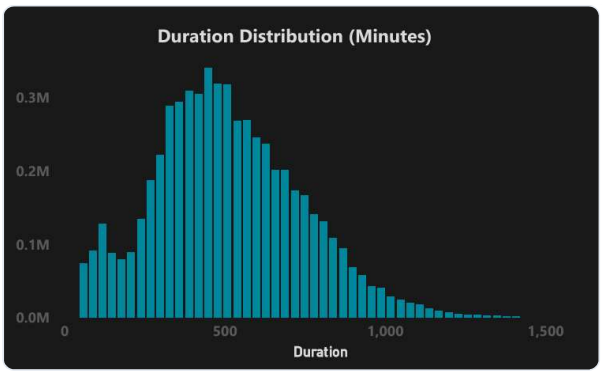
Fare Distribution

Price range histogram

Duration Distribution

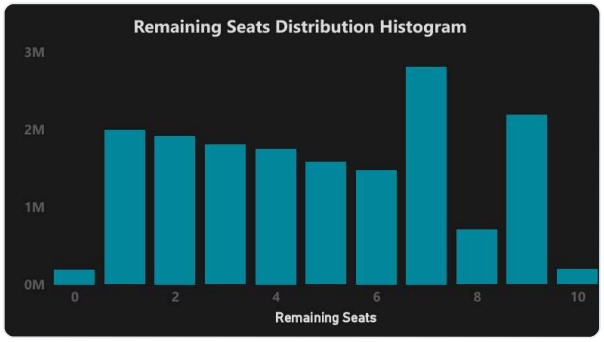
Flight duration patterns





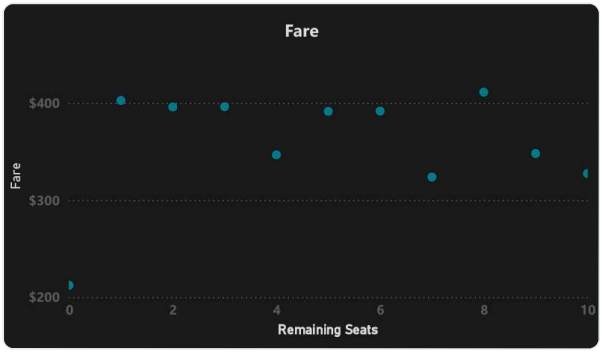
Remaining Seats Distribution

Inventory availability patterns



Fare vs Remaining Seats

Price-inventory correlation



Conclusion

This project successfully demonstrates an enterprise-grade flight pricing prediction system built on modern data engineering principles and scalable infrastructure.

Key Achievements

Data Engineering

- Ingested 82 million flight records via Snowflake S3 integration
- Implemented medallion architecture (Bronze/Silver/Gold/ML)
- Built star schema with role-playing dimensions
- Memory-efficient processing with Polars and Apache Arrow

Machine Learning

- 91% R^2 score with $33.97 MAE
- Parsed fare basis codes into 7 features
- IQR-based outlier detection (3% reduction)
- 70-minute training on SageMaker ml.r6id.2xlarge

Production Deployment

- FastAPI with <50ms latency
- AWS ECS Fargate with auto-scaling
- 5 dynamic pricing factors
- Automated deployment scripts

Orchestration

- Dagster asset-based orchestration
- MLflow experiment tracking
- Weekly automated retraining
- Isolated demo environments

Business Value Delivered

Accuracy	91% variance explained enables confident pricing recommendations
Speed	Real-time API responses support interactive booking experiences
Cost	Serverless infrastructure scales with demand, eliminating idle waste

---