Flight Price Analytics — Project Report

1) Project Overview

Built an end-to-end analytics stack to answer a traveler's core question: **"Buy now or wait?"** The solution ingests daily fare snapshots, cleans and models them, and exposes **actionable pricing signals** (lead-time sweet spots, weekday vs. weekend effects, and route alerts) through a **one-page Power BI dashboard**. Structure and deliverable style mirror the author's prior retail analytics project to keep outputs recruiter-friendly and decision-ready.

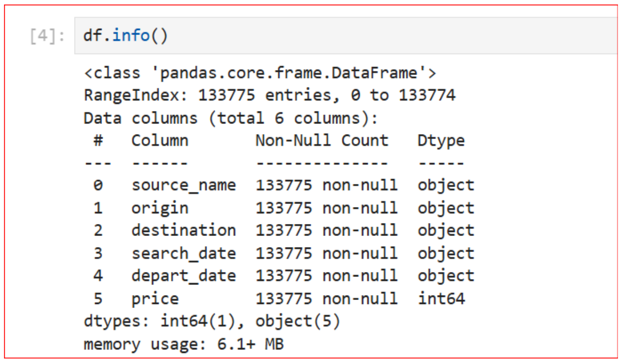
2) Dataset Summary

- **Grain**: one row per (route, search_date, depart_date) fare snapshot
- **Span**: ~Jun-2025 → Nov-2025 (sample shown; scalable to longer history)
- **Rows / Cols:** 133776 rows and 9 columns
- **Core fields**: origin, destination, route, price, search_date, depart_date, lead_time_days
- **Derived fields**: route key, **lead-time bucket** (0–7, 8–14, 15–30, 31–60, 60+), **is_weekend** (by depart_date), 7-day baseline of price for forecast vs. actual, and 7-day % change for alerting

3) Exploratory Data Analysis (Python)

Stack: pandas / numpy / matplotlib.
Steps

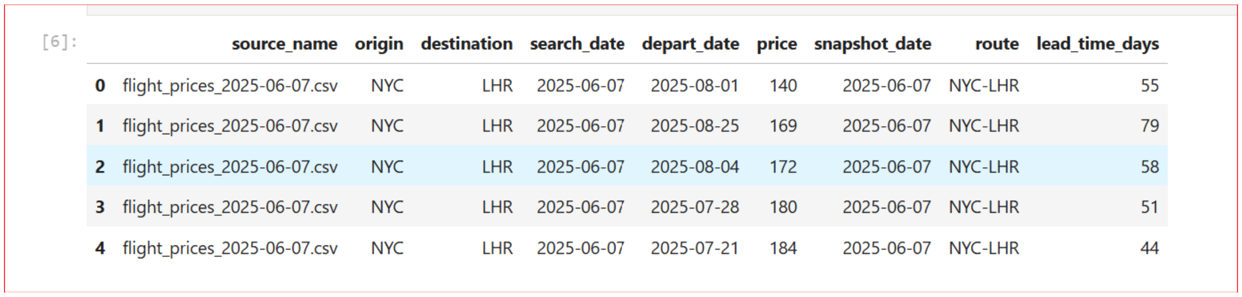
- **Health**: .info(), NA scan, duplicate scan at (route, depart_date, snapshot_date); clipped non-positive prices.



```
[4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 133775 entries, 0 to 133774
Data columns (total 6 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   source_name  133775 non-null  object
 1   origin       133775 non-null  object
 2   destination  133775 non-null  object
 3   search_date  133775 non-null  object
 4   depart_date  133775 non-null  object
 5   price        133775 non-null  int64
dtypes: int64(1), object(5)
memory usage: 6.1+ MB
```

- **Feature engineering:**

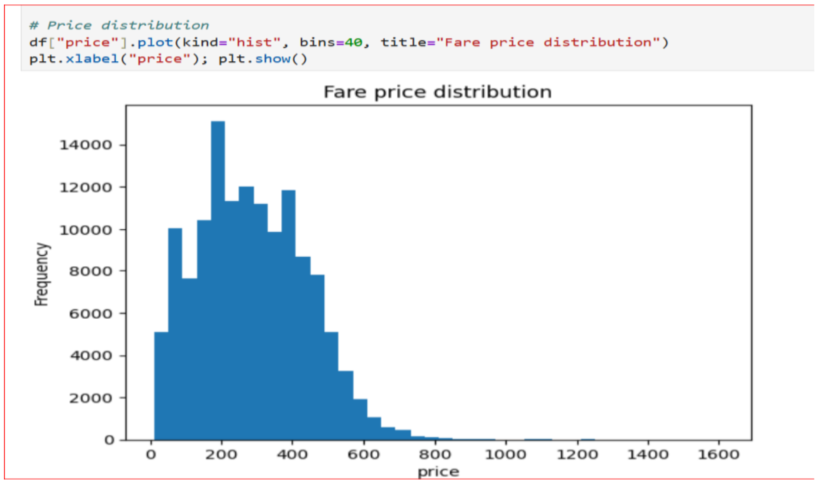


[6]:

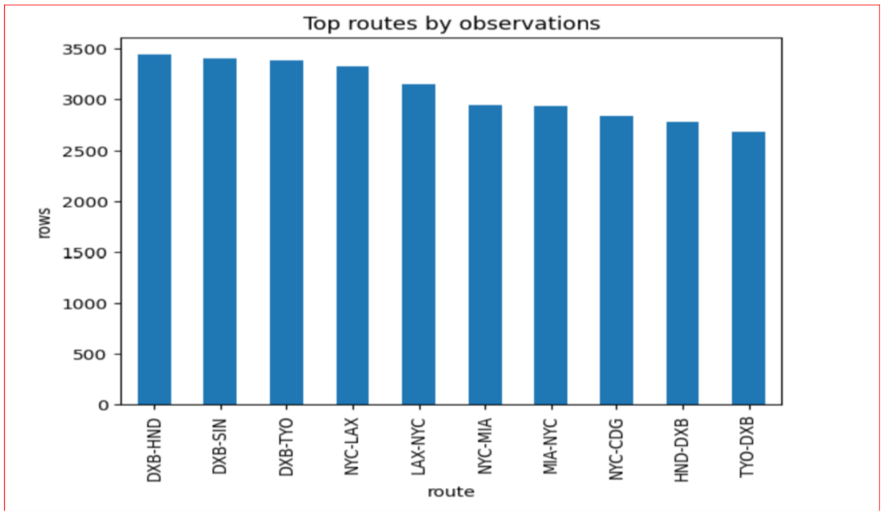
	source_name	origin	destination	search_date	depart_date	price	snapshot_date	route	lead_time_days
0	flight_prices_2025-06-07.csv	NYC	LHR	2025-06-07	2025-08-01	140	2025-06-07	NYC-LHR	55
1	flight_prices_2025-06-07.csv	NYC	LHR	2025-06-07	2025-08-25	169	2025-06-07	NYC-LHR	79
2	flight_prices_2025-06-07.csv	NYC	LHR	2025-06-07	2025-08-04	172	2025-06-07	NYC-LHR	58
3	flight_prices_2025-06-07.csv	NYC	LHR	2025-06-07	2025-07-28	180	2025-06-07	NYC-LHR	51
4	flight_prices_2025-06-07.csv	NYC	LHR	2025-06-07	2025-07-21	184	2025-06-07	NYC-LHR	44

- lead_time_days = (depart_date - search_date)
- lead_bucket via SWITCH/CASE bands for charting
- is_weekend = WEEKDAY(depart_date, 2) >= 6 (Sat/Sun)
- 7-day rolling **baseline** of price to compare against last snapshot

- **Quick EDA** (by route):



```
# Price distribution
df["price"].plot(kind="hist", bins=40, title="Fare price distribution")
plt.xlabel("price"); plt.show()
```



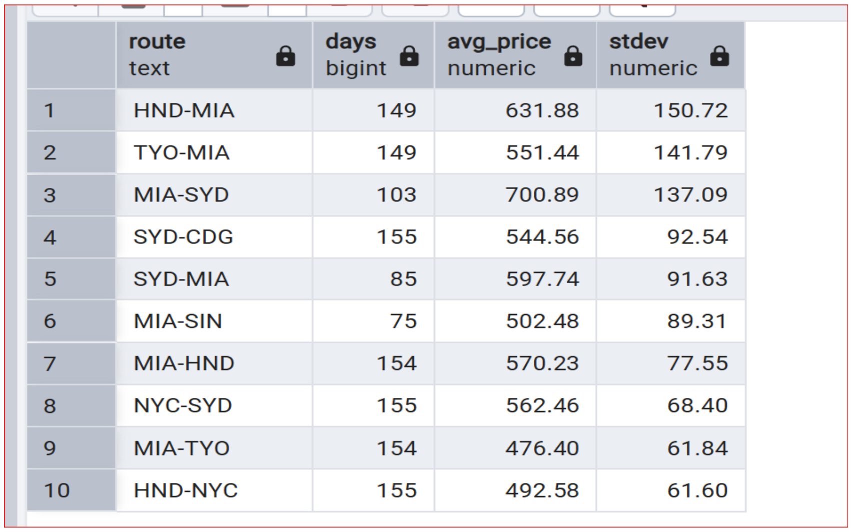
- **Seasonality**: visible month-to-month waves in average search-day price
- **Lead-time curve**: some routes show **U-shaped** price vs. lead time (cheap around 15–30 days, higher very early/very late)
- **Weekend effect**: for several routes, **weekend mean < weekday mean**.

4) Data Analysis using SQL

Reusable SQL answered the business questions below (snippets are in the project SQL file).

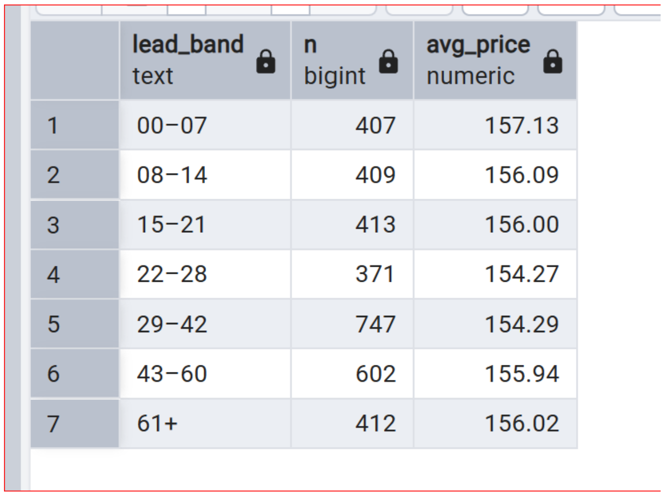
--1) Which routes are the most price-volatile?

--Why: volatile routes are good candidates for alerts and "Buy now" nudges.



	route text	days bigint	avg_price numeric	stdev numeric
1	HND-MIA	149	631.88	150.72
2	TYO-MIA	149	551.44	141.79
3	MIA-SYD	103	700.89	137.09
4	SYD-CDG	155	544.56	92.54
5	SYD-MIA	85	597.74	91.63
6	MIA-SIN	75	502.48	89.31
7	MIA-HND	154	570.23	77.55
8	NYC-SYD	155	562.46	68.40
9	MIA-TYO	154	476.40	61.84
10	HND-NYC	155	492.58	61.60

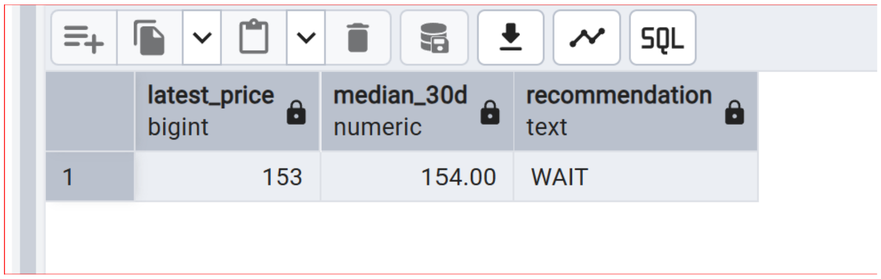
--2) What does the lead-time price curve look like for a route?



	lead_band text	n bigint	avg_price numeric
1	00–07	407	157.13
2	08–14	409	156.09
3	15–21	413	156.00
4	22–28	371	154.27
5	29–42	747	154.29
6	43–60	602	155.94
7	61+	412	156.02

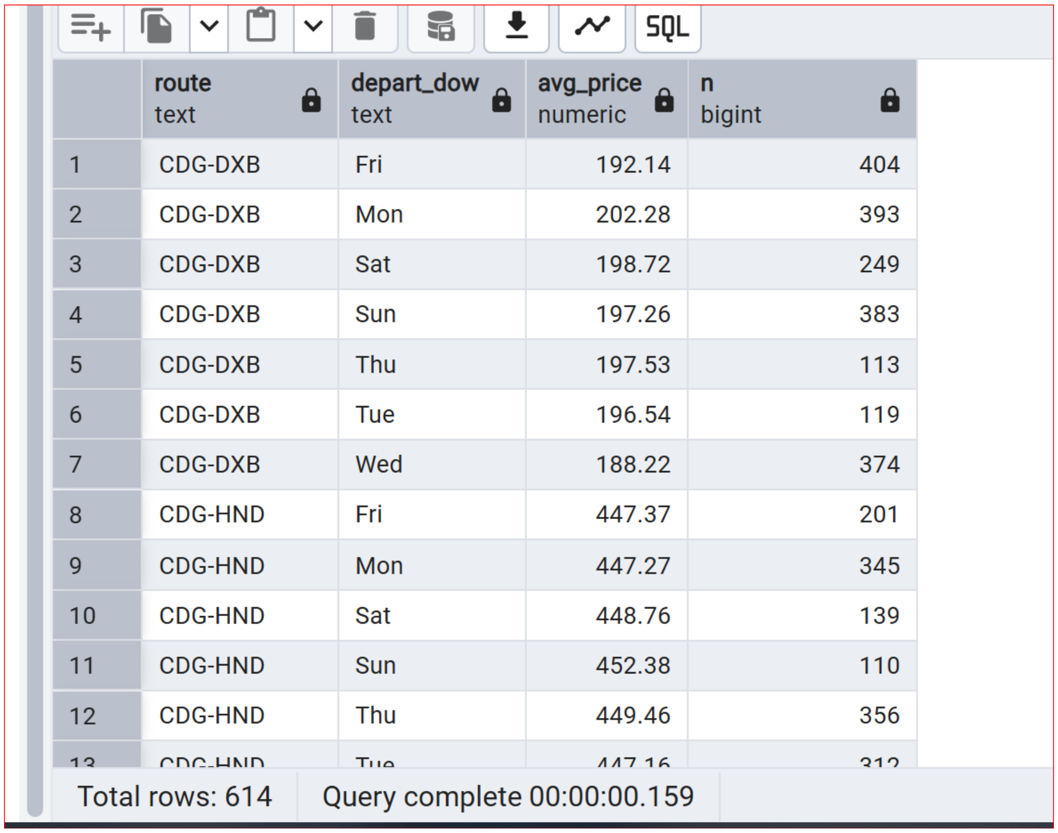
--3) “Buy or Wait” for a given route & departure?

--Compare **latest snapshot’s** price to the last 30 days of snapshots (same route+depart_date). Flag if latest is ≥10% below the 30-day median.



	latest_price bigint	median_30d numeric	recommendation text
1	153	154.00	WAIT

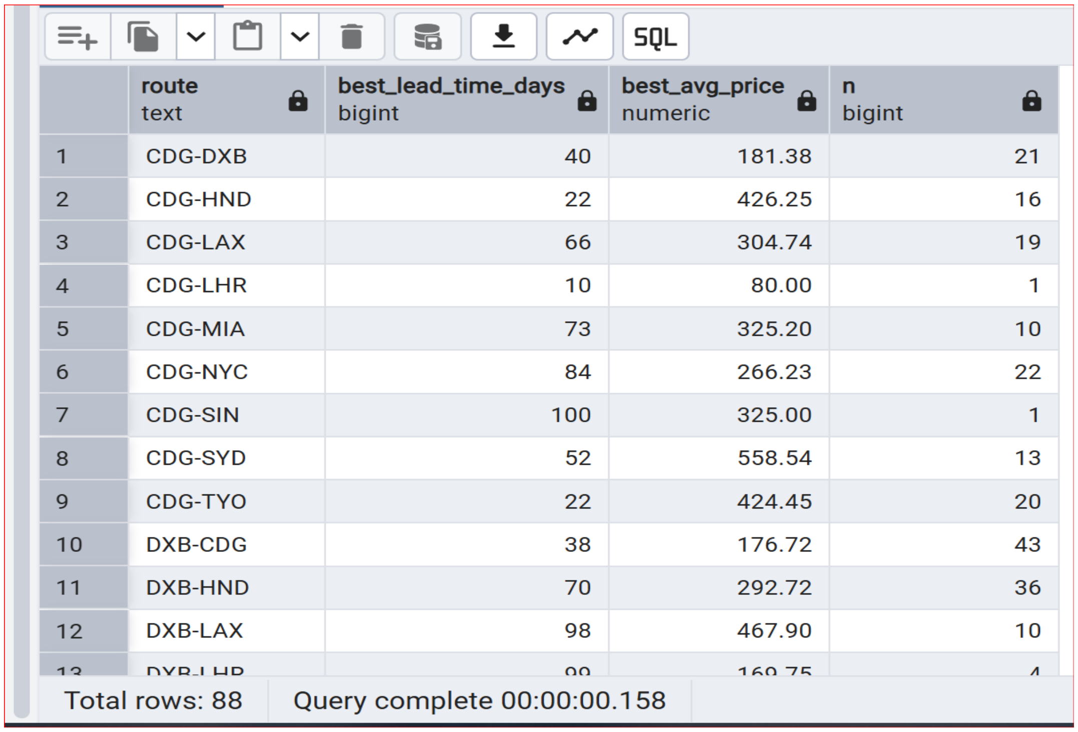
--4) What day of week is typically cheapest to depart (by route)?



	route text	depart_dow text	avg_price numeric	n bigint
1	CDG-DXB	Fri	192.14	404
2	CDG-DXB	Mon	202.28	393
3	CDG-DXB	Sat	198.72	249
4	CDG-DXB	Sun	197.26	383
5	CDG-DXB	Thu	197.53	113
6	CDG-DXB	Tue	196.54	119
7	CDG-DXB	Wed	188.22	374
8	CDG-HND	Fri	447.37	201
9	CDG-HND	Mon	447.27	345
10	CDG-HND	Sat	448.76	139
11	CDG-HND	Sun	452.38	110
12	CDG-HND	Thu	449.46	356
13	CDG-HND	Tue	447.16	312

Total rows: 614 Query complete 00:00:00.159

--5) What booking window (lead time) tends to have the lowest average price per route?



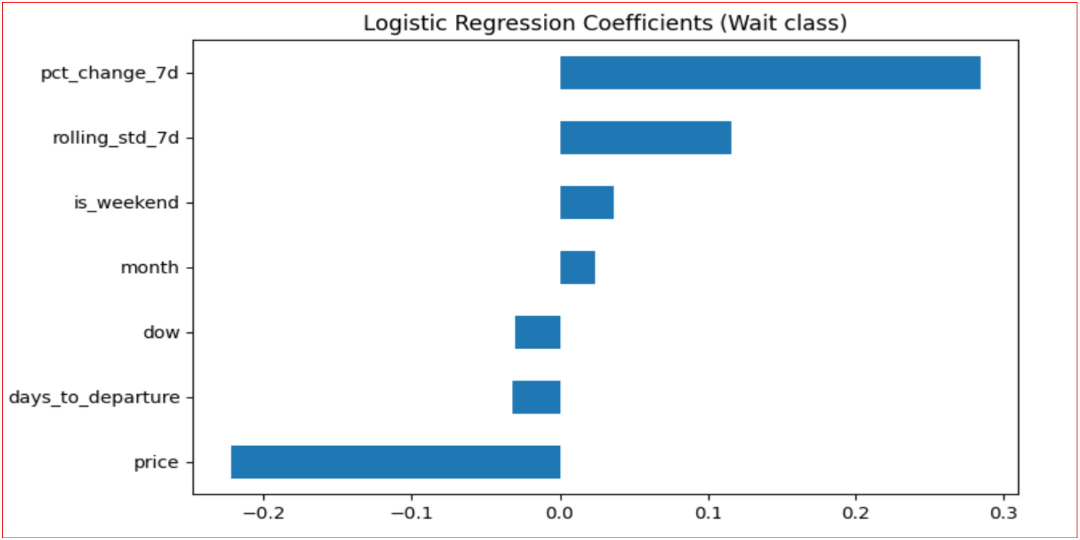
	route text	best_lead_time_days bigint	best_avg_price numeric	n bigint
1	CDG-DXB	40	181.38	21
2	CDG-HND	22	426.25	16
3	CDG-LAX	66	304.74	19
4	CDG-LHR	10	80.00	1
5	CDG-MIA	73	325.20	10
6	CDG-NYC	84	266.23	22
7	CDG-SIN	100	325.00	1
8	CDG-SYD	52	558.54	13
9	CDG-TYO	22	424.45	20
10	DXB-CDG	38	176.72	43
11	DXB-HND	70	292.72	36
12	DXB-LAX	98	467.90	10
13	DXB-LHR	99	169.75	4

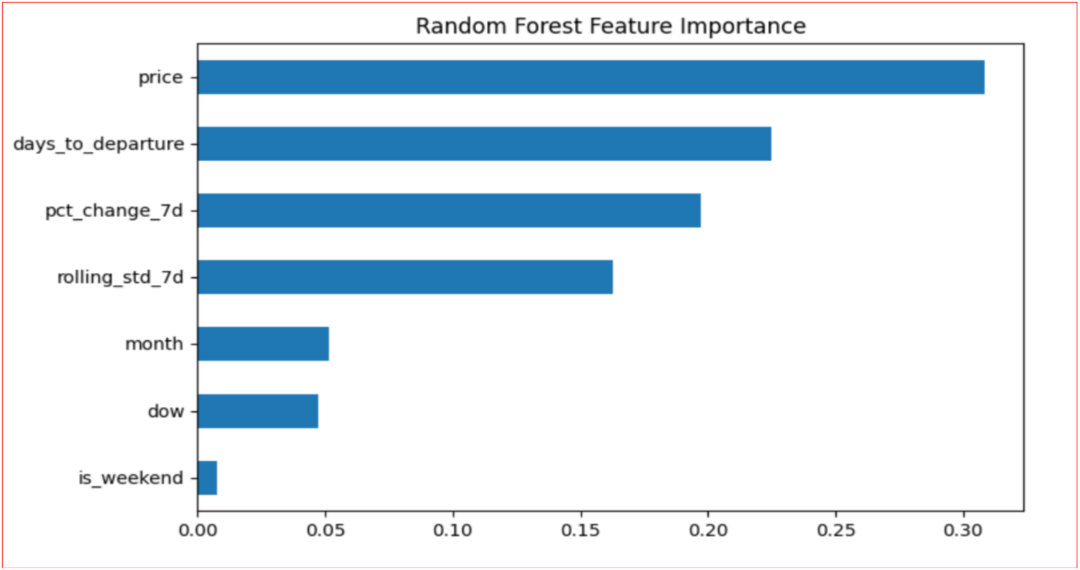
Total rows: 88 Query complete 00:00:00.158

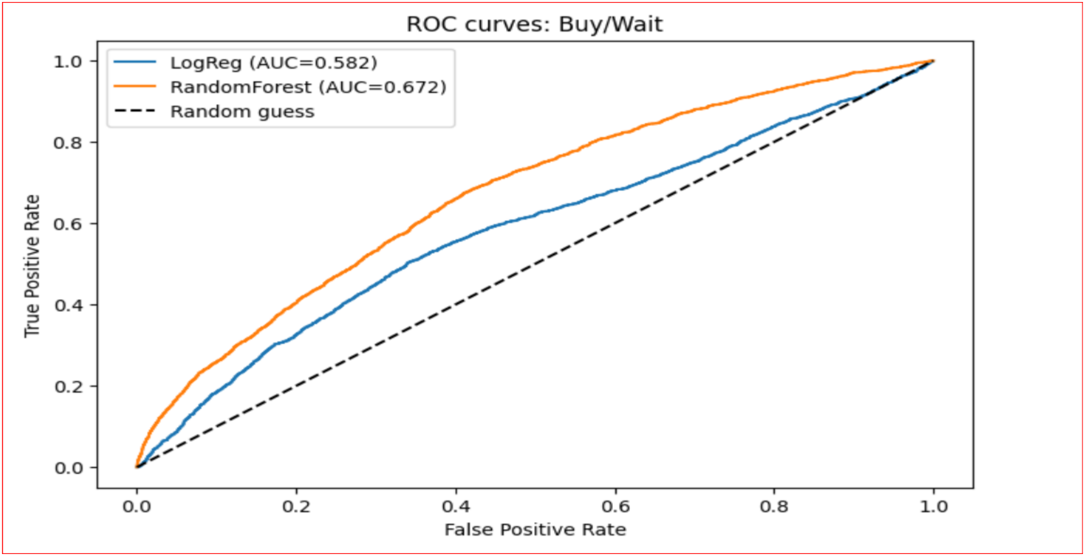
This modular, question-first query pattern follows the same “analysis menu” approach used in the floral project to support BI and ad-hoc analysis.

5) Buy/Wait ML Model (Python, scikit-learn)

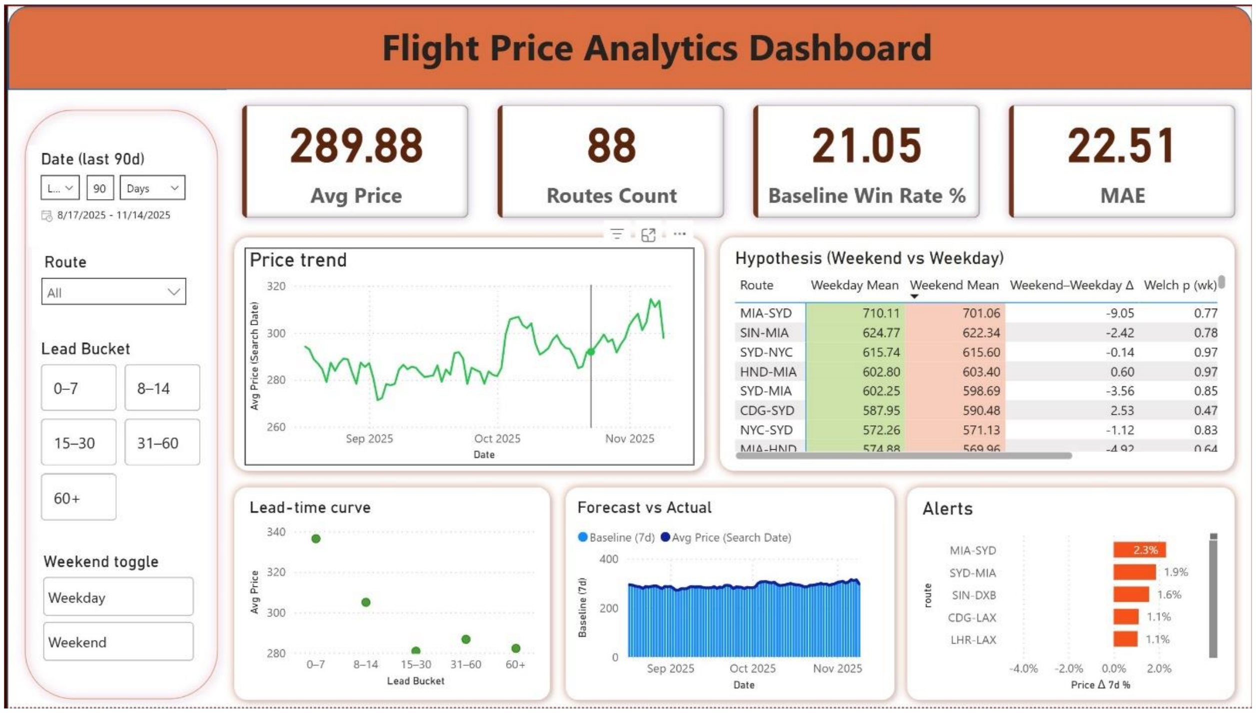
Framed “**Buy now vs Wait**” as a **supervised classification** problem using engineered features (current price vs rolling **7-day mean**, volatility, days-to-departure, DOW/month, weekend flag). Trained **logistic regression** and class-balanced **random forest** models with time-based **cross-validation**; evaluated with **ROC AUC**, precision/recall, and confusion matrices. Exposed a simple buy_or_wait() helper that returns both the decision and the probability of “Wait” for use in the app and alerts.


Logistic Regression Coefficients (Wait class)
pct_change_7d
rolling_std_7d
is_weekend
month
dow
days_to_departure
price
−0.2
−0.1
0.0
0.1
0.2
0.3


Random Forest Feature Importance
price
days_to_departure
pct_change_7d
rolling_std_7d
month
dow
is_weekend
0.00
0.05
0.10
0.15
0.20
0.25
0.30


ROC curves: Buy/Wait
LogReg (AUC=0.582)
RandomForest (AUC=0.672)
Random guess
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

5) Dashboard in Power BI



Flight Price Analytics Dashboard

Date (last 90d)

L... | 90 | Days

8/17/2025 - 11/14/2025

Route: All

Lead Bucket: 0–7, 8–14, 15–30, 31–60, 60+

Weekend toggle: Weekday, Weekend

289.88	Avg Price
88	Routes Count
21.05	Baseline Win Rate %
22.51	MAE

Hypothesis (Weekend vs Weekday)

Route	Weekday Mean	Weekend Mean	Weekend–Weekday Δ	Welch p (wk)
MIA-SYD	710.11	701.06	-9.05	0.77
SIN-MIA	624.77	622.34	-2.42	0.78
SYD-NYC	615.74	615.60	-0.14	0.97
HND-MIA	602.80	603.40	0.60	0.97
SYD-MIA	602.25	598.69	-3.56	0.85
CDG-SYD	587.95	590.48	2.53	0.47
NYC-SYD	572.26	571.13	-1.12	0.83
MIA-HND	574.88	569.96	-4.92	0.64

6) Business Recommendations

A. Daily "Buy" list

- Compare today's price to the **30-day median**.
- If today is **≥10% cheaper**, flag the route/departure as **BUY** and surface it in email/banner.

B. When to book (per route)

- Publish each route's **best lead-time window** (e.g., "15–30 days out").
- Refresh monthly so guidance stays current.

C. Weekend messaging

- If **weekend < weekday** prices, promote **"Save more on weekend departures."**
- Time ads and push notifications on **Thu–Fri**.

D. Volatility watchlist

- Track routes with the **highest price variance**.

- Add **price-drop badges** in search results and poll those routes more frequently.

E. Keep improving the signals

- Monitor **Baseline Win Rate** and **MAE**; switch to longer, seasonal baselines (e.g., **28-day**) where errors stay high.
- Retain the **question-driven SQL menu + one-page dashboard** so teams can self-serve and act fast.

F. ML-powered Buy/Wait guidance

- Use the random- forest Buy/Wait model alongside simple rules to score each route/departure.
- Surface high-confidence “Wait” and “Buy now” recommendations in the dashboard and app, and track model performance (ROC AUC, precision/recall) over time.