



UNITED AIRLINES

DELHI TECHNOLOGICAL UNIVERSITY



FLIGHT OPERATIONS EFFICIENCY & DIFFICULTY ANALYSIS

DATA-DRIVEN EVALUATION OF FLIGHT PERFORMANCE
AND OPERATIONAL RISK

Team Samurai

NITESH KUMARJHA(2K22/IT/116)

KSHITIJ KUMAR OJHA(2K22/BT/23)

S.NO	TOPIC	PAGE FROM	PAGE TO
1	PROBLEM STATEMENT	3	7
2	PROJECT OVERFLOW	8	8
3	EDA	9	18
4	FLIGHT DIFFICULTY SCORE CALCULATION	19	25
5	DIAGNOSE DIFFICULT FLIGHT DNA	26	29
6	INSIGHTS AND RECOMMENDATION	29	36
7	APPENDIX	37	39

PROBLEM STATEMENT

Context

United Airlines' frontline teams at Chicago O'Hare International Airport (ORD) manage hundreds of daily departures, ensuring each flight leaves safely and on time. However, not all flights are equally easy to manage — some are operationally more complex due to:

- Limited ground turnaround time
- High baggage volumes (especially transfer bags)
- Large passenger loads or many special service requests
- Tight connection schedules between flights

These factors create unpredictable workloads for ground staff.

Currently, identifying “difficult” flights depends on experience and intuition, which is:

- Subjective – varies by person
- Reactive – issues discovered only after delays
- Non-scalable – cannot be applied consistently across stations

A data-driven approach is needed to quantify flight complexity, enabling proactive planning, efficient staffing, and improved on-time performance.



PROBLEM STATEMENT

Challenges

1. Manual, Experience-Based Assessment
 - Frontline teams currently rely on intuition and past experience to identify difficult flights.
 - Leads to subjectivity and inconsistent decisions across shifts or staff.
2. Multiple Interacting Factors
 - Operational difficulty is influenced by many variables:
 - Tight ground times / short turnaround
 - High volume of checked or transfer baggage
 - Passenger load and special service requests (SSR)
 - Flight connections and aircraft type
 - Manually combining these factors is error-prone and inefficient.
3. Reactive Operations
 - Without predictive insights, flight difficulties are often discovered only after delays occur.
 - Makes it harder to proactively allocate resources, increasing the risk of operational disruption.



PROBLEM STATEMENT

Objective



1. Quantify Flight Difficulty
 - Develop a Flight Difficulty Score (FDS) using historical and operational data.
 - Score captures the combined impact of ground time, baggage, passenger load, and SSRs.
2. Enable Proactive Planning
 - Identify flights likely to face operational challenges before departure.
 - Allow for preemptive staff allocation, improved baggage handling, and operational readiness.
3. Support Data-Driven Decisions
 - Replace intuition-based judgments with objective, repeatable analysis.
 - Facilitate consistent resource planning and reporting across teams.
4. Optimize On-Time Performance & Efficiency
 - Reduce delays caused by unanticipated operational complexity.
 - Improve overall turnaround efficiency and passenger experience at ORD.



DELIVERABLES



1. CLEANED & INTEGRATED DATASET

- CONSOLIDATED FLIGHT, BAGGAGE, AND PASSENGER DATA INTO A MASTER DATASET READY FOR ANALYSIS.
- HANDLED MISSING VALUES, STANDARDIZED FORMATS, AND REMOVED DUPLICATES.

2. EXPLORATORY DATA ANALYSIS (EDA) REPORTS

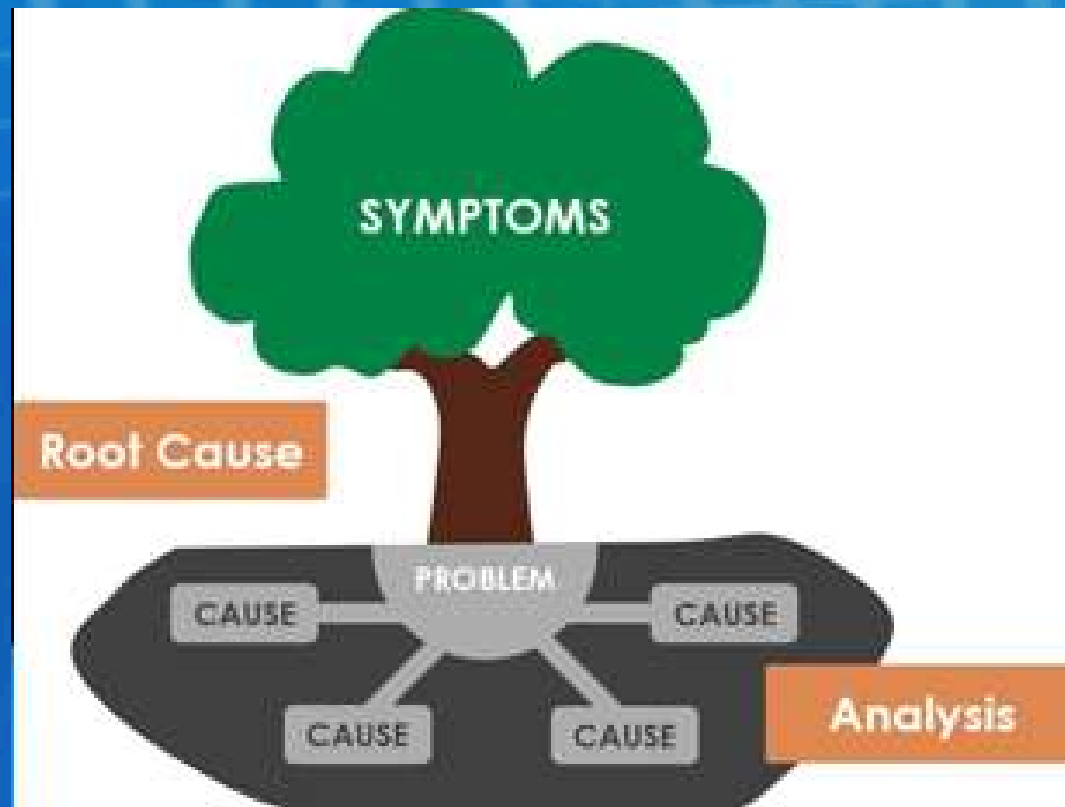
- VISUALIZED DEPARTURE & ARRIVAL DELAYS, GROUND TIME PRESSURE, LOAD FACTOR, AND BAGGAGE MIX.
- IDENTIFIED TRENDS, CORRELATIONS, AND OPERATIONAL PATTERNS.
- DETECTED AND ANALYZED OUTLIERS TO ENSURE DATA RELIABILITY.

3. FLIGHT DIFFICULTY SCORE (FDS)

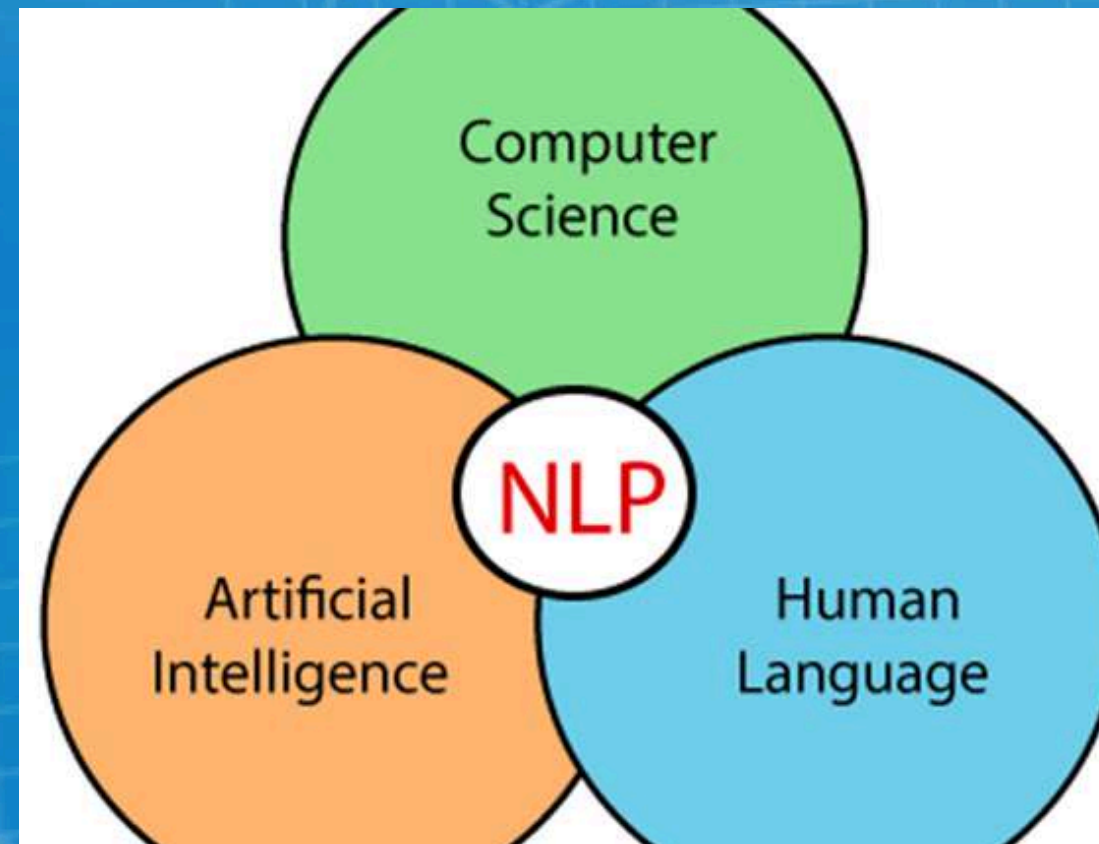
CREATED A COMPOSITE SCORE TO CLASSIFY FLIGHTS AS LOW, MODERATE, OR HIGH DIFFICULTY.
HELPS PREDICT OPERATIONALLY CHALLENGING FLIGHTS PROACTIVELY.



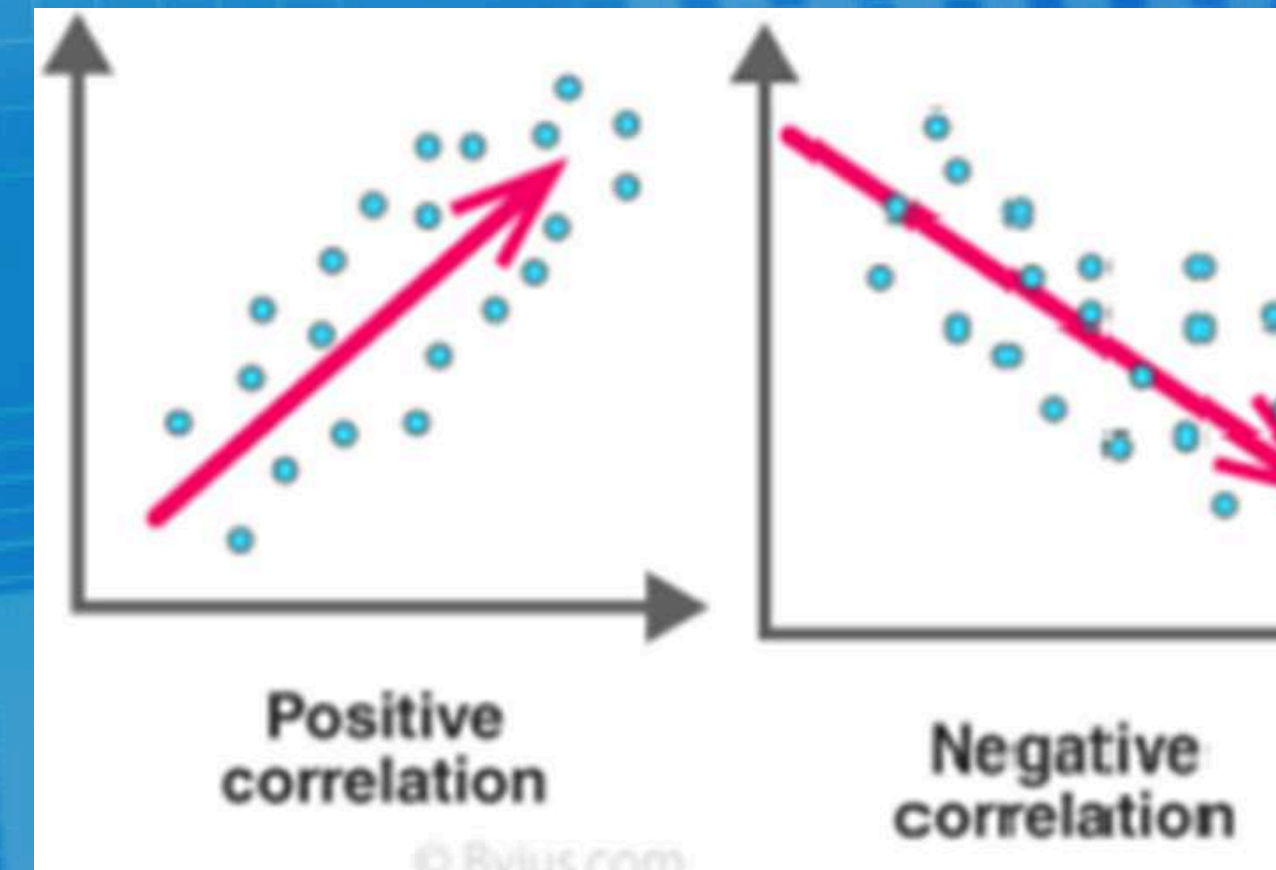
CONCEPTS



ROOT CAUSE ANALYSIS IS A PROBLEM-SOLVING APPROACH THAT USES THE ANALOGY OF ROOTS AND BLOOMS TO MODEL CAUSE-AND-EFFECT RELATIONSHIPS. RATHER THAN FOCUSING ON WHAT'S ABOVE THE SURFACE, ROOT CAUSE ANALYSIS TROUBLESHOOT SOLUTIONS TO PROBLEMS BY ANALYZING WHAT IS CAUSING THEM.



THE NATURAL LANGUAGE TOOLKIT, OR MORE COMMONLY NLTK, IS A SUITE OF LIBRARIES AND PROGRAMS FOR SYMBOLIC AND STATISTICAL NATURAL LANGUAGE PROCESSING (NLP) FOR ENGLISH WRITTEN IN THE PYTHON PROGRAMMING LANGUAGE. IT SUPPORTS CLASSIFICATION, TOKENIZATION, STEMMING, TAGGING, PARSING, AND SEMANTIC REASONING FUNCTIONALITIES

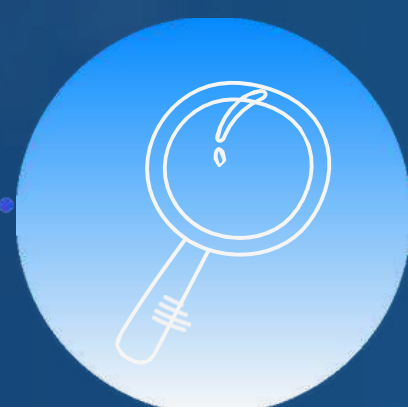


CORRELATION IS A STATISTICAL TERM THAT DESCRIBES HOW TWO VARIABLES ARE RELATED TO EACH OTHER. CORRELATION MEASURES THE STRENGTH AND DIRECTION OF THE LINEAR RELATIONSHIP BETWEEN TWO VARIABLES. A CORRELATION COEFFICIENT IS A NUMBER THAT SUMMARIZES THE CORRELATION BETWEEN TWO VARIABLES. IT RANGES FROM -1 TO 1

PROJECT OVERFLOW



01
Understanding
data



02
EDA



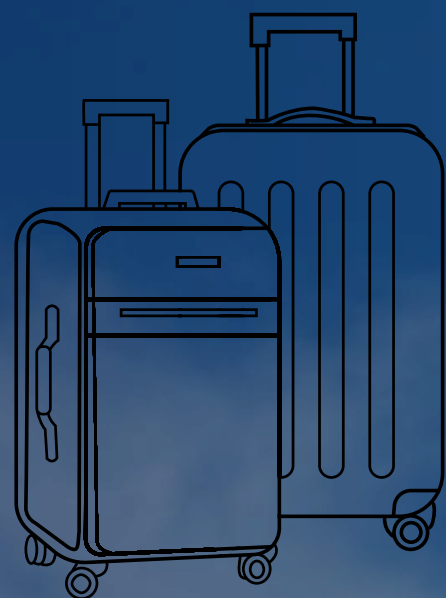
03
Feature engineering



04
Flight Difficulty
Scoring



05
Post-Analysis &
Operational
Insights



DATA DESCRIPTION

- FLIGHT LEVEL DATA: THE MASTER LOG OF ALL FLIGHTS, DETAILING THEIR SCHEDULES, TIMINGS, AND AIRCRAFT TYPES.
- PNR FLIGHT LEVEL DATA: THE PASSENGER MANIFEST, SHOWING WHO IS ON EACH FLIGHT, INCLUDING CHILDREN AND FARE TYPES.
- PNR REMARK LEVEL DATA: THE SPECIAL NEEDS LOG, LISTING ALL PASSENGER REQUESTS LIKE WHEELCHAIRS OR UNACCOMPANIED MINORS.
- BAG LEVEL DATA: THE BAGGAGE INVENTORY, DETAILING EVERY CHECKED AND TRANSFER BAG FOR EACH FLIGHT.
- AIRPORTS DATA: A SIMPLE REFERENCE FILE TO IDENTIFY WHICH FLIGHTS ARE INTERNATIONAL.

[LINK TO DATASET](#)



EXPLORATORY DATA ANALYSIS

1. LOADING DATA

```
import numpy as np
```

```
flight_df = pd.read_csv('../data/Flight Level Data.csv')
pnr_df = pd.read_csv('../data/PNR+Flight+Level+Data.csv')
remarks_df = pd.read_csv('../data/PNR Remark Level Data.csv')
bags_df = pd.read_csv('../data/Bag+Level+Data.csv')
airports_df = pd.read_csv('../data/Airports Data.csv')
```

```
print(f"Flights: {flight_df.shape}")
print(f"PNR: {pnr_df.shape}")
print(f"Remarks: {remarks_df.shape}")
print(f"Bags: {bags_df.shape}")
print(f"Airports: {airports_df.shape}")
```

```
Flights: (8099, 15)
PNR: (687878, 12)
Remarks: (51698, 4)
Bags: (687245, 8)
Airports: (5612, 2)
```

2. FINDING MISSING DATA

```
print("\n--- Missing Value Summary ---")
for name, df in {'Flights': flight_df, 'Bags': bags_df, 'Passengers': pnr_df}.items():
    print(f"\n{name} Missing Values (%):")
    print((df.isna().sum() / len(df) * 100).round(2))

# Visualize missing values
plt.figure(figsize=(10,4))
sns.heatmap(flight_df.isnull(), cbar=False)
plt.title("Flights Data Missing Values")
plt.show()
```

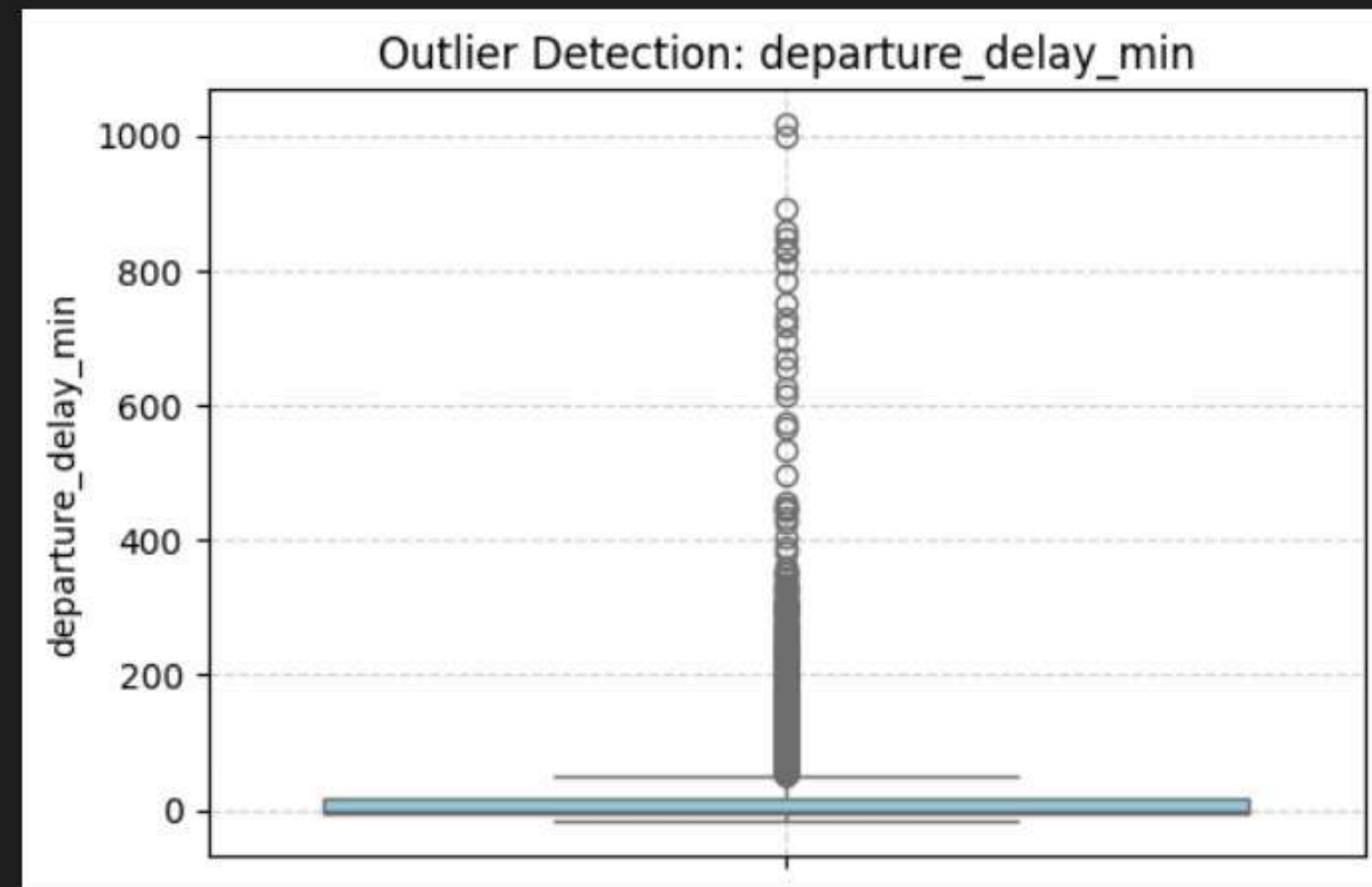
--- Missing Value Summary ---

```
Flights Missing Values (%):
company_id                0.0
flight_number             0.0
scheduled_departure_date_local  0.0
scheduled_departure_station_code  0.0
scheduled_arrival_station_code  0.0
scheduled_departure_datetime_local  0.0
scheduled_arrival_datetime_local  0.0
actual_departure_datetime_local  0.0
actual_arrival_datetime_local  0.0
total_seats               0.0
fleet_type                0.0
carrier                   0.0
scheduled_ground_time_minutes  0.0
actual_ground_time_minutes    0.0
minimum_turn_minutes         0.0
dtype: float64
```


OUTLIER DETECTION

OUTLIER IN THE DEPARTURE_DELAY_MIN

🚨 Outliers detected in departure_delay_min: 1062



BOX PLOT

```
# Ensure plots render in notebook
%matplotlib inline

# List of numeric features to check for outliers
numeric_features = ['departure_delay_min', 'arrival_delay_min', 'load_factor']

# Function to detect outliers using the IQR method
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower) | (df[column] > upper)]
    return outliers, lower, upper

# Store summary for reporting
outlier_summary = []

for col in numeric_features:
    # Drop NaNs before checking
    df_to_check = flight_df.dropna(subset=[col])

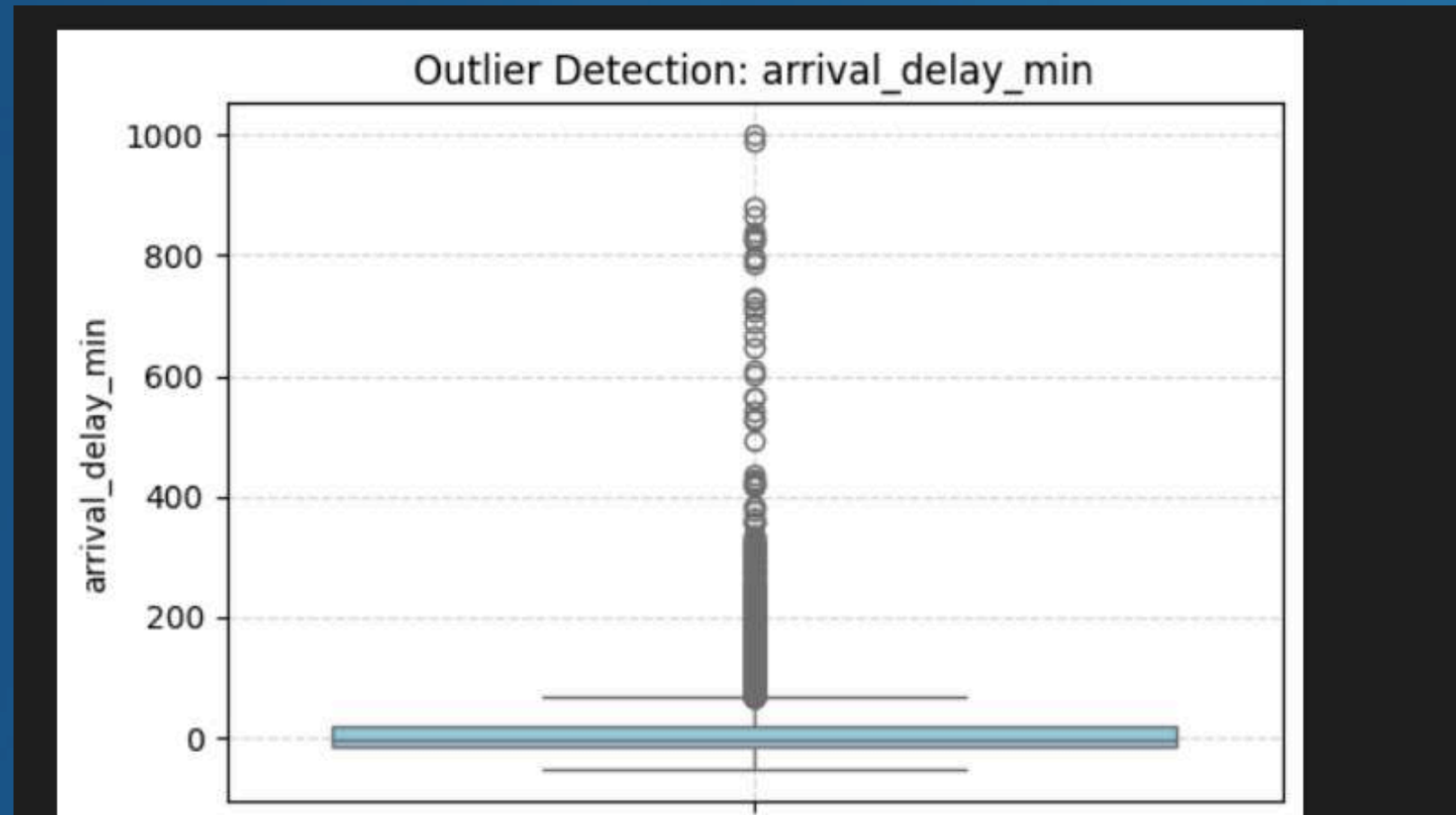
    outliers, lower, upper = detect_outliers_iqr(df_to_check, col)

    outlier_summary.append({
        "Feature": col,
        "Lower Bound": round(lower, 2),
        "Upper Bound": round(upper, 2),
        "Outlier Count": len(outliers)
    })

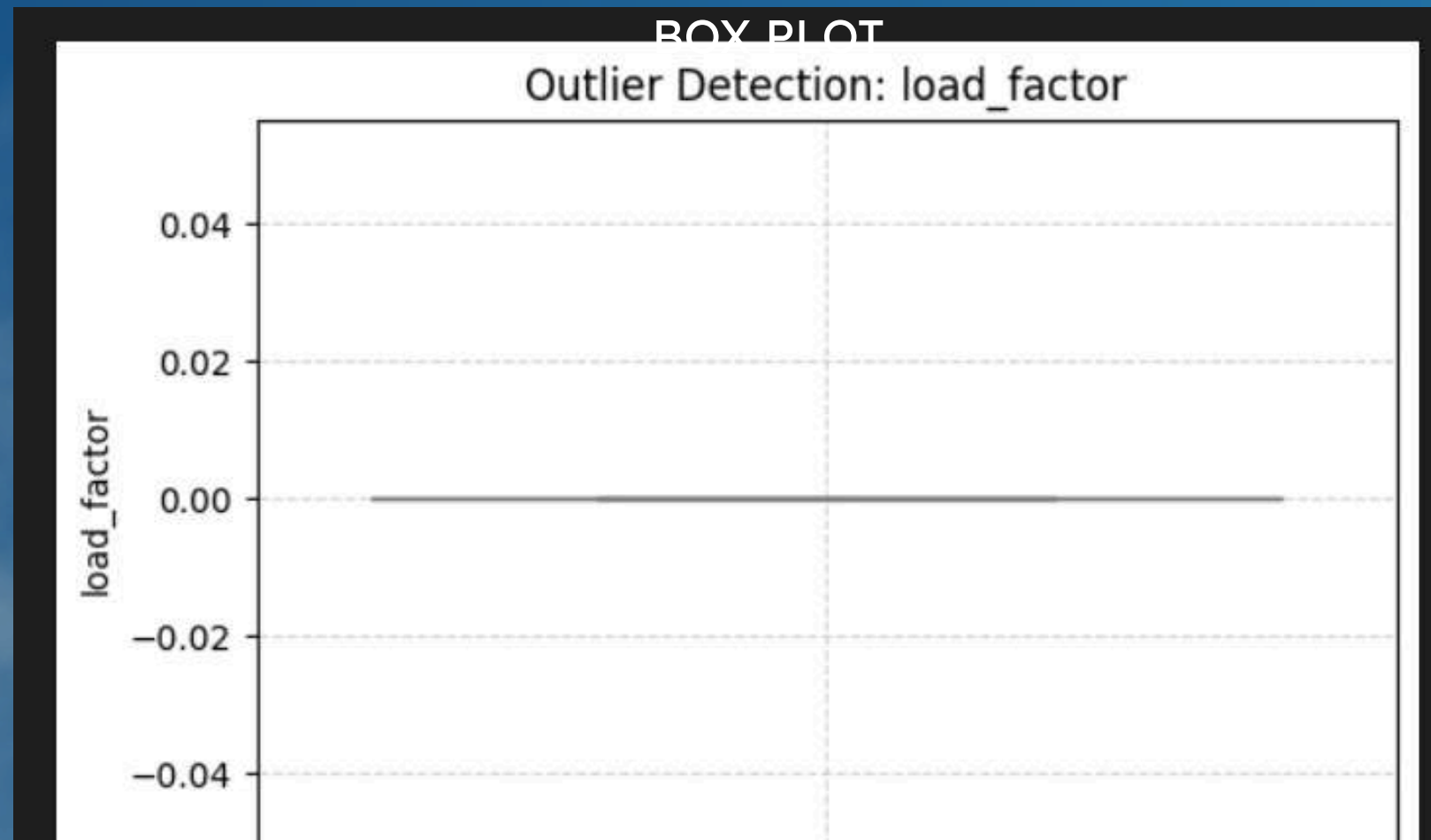
    print(f"\n🚨 Outliers detected in {col}: {len(outliers)}")
```


OUTLIER DETECTION

- OUTLIER IN THE ARRIVAL_DELAY_MIN



- OUTLIER IN THE LOAD_FACTOR



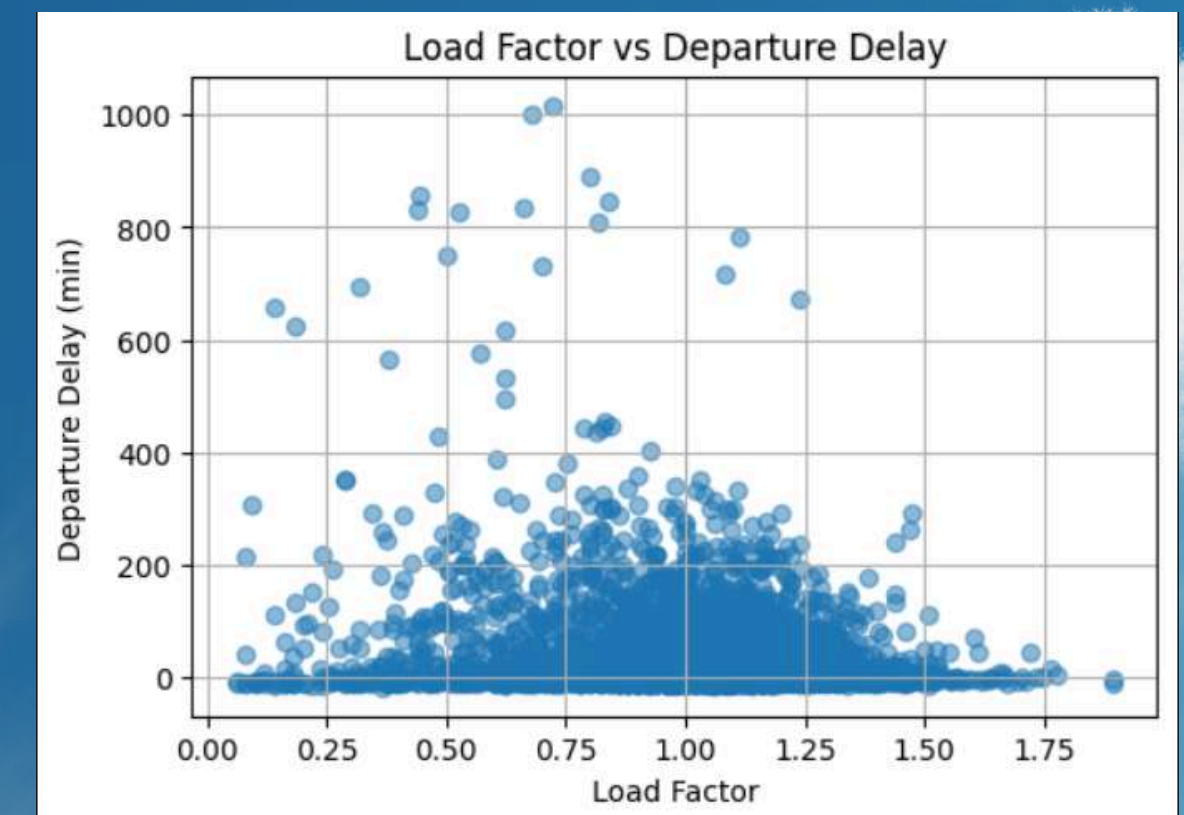
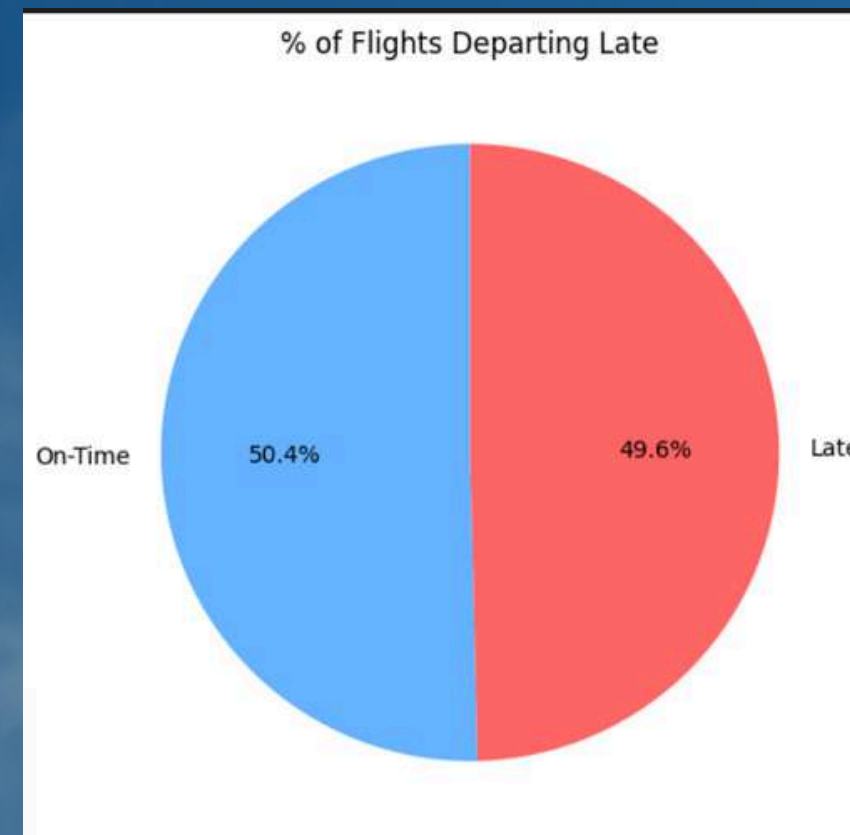
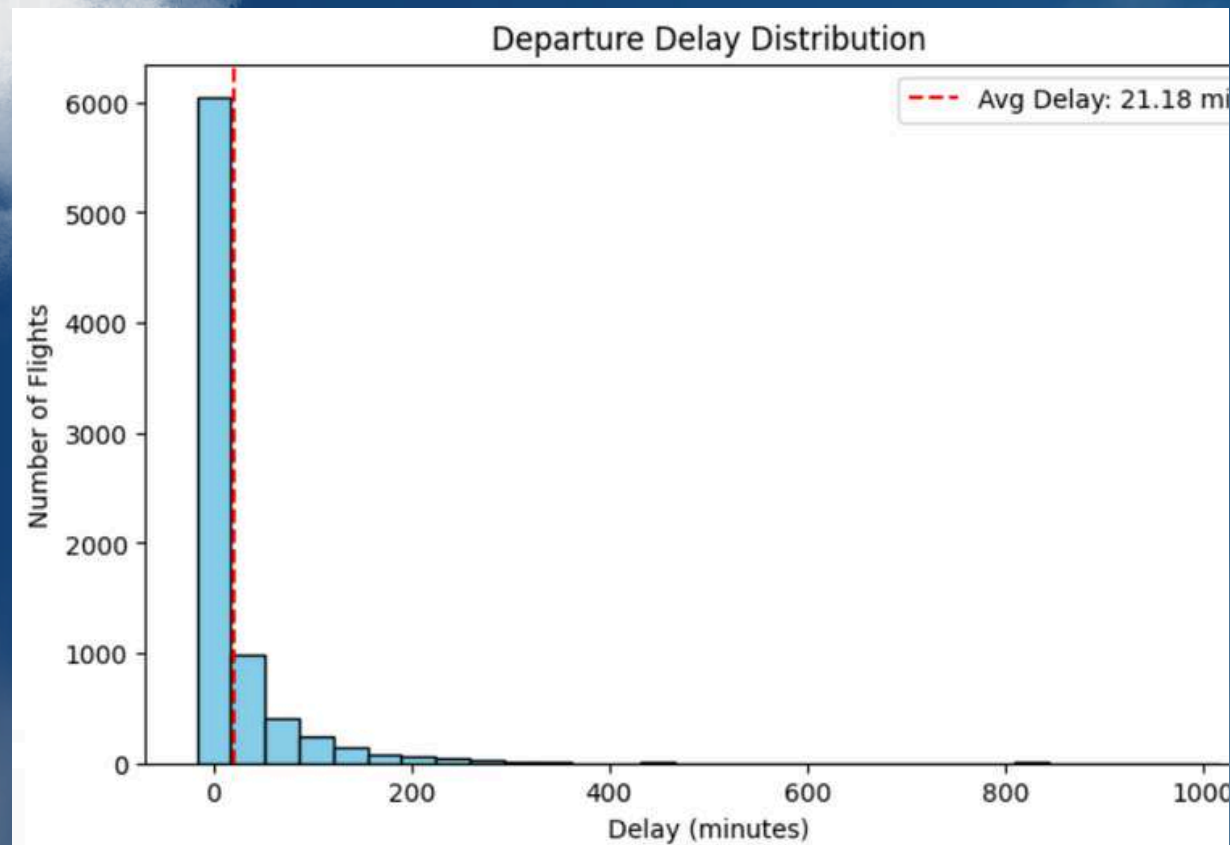
1. WHAT IS THE AVERAGE DELAY AND WHAT PERCENTAGE OF FLIGHTS DEPART LATER THAN SCHEDULED?

- AVERAGE DEPARTURE DELAY: 21.18 MINUTES

```
import matplotlib.pyplot as plt
```

```
# --- 1 Average delay and % of late departures ---  
avg_delay = master_df['departure_delay_min'].mean()  
pct_late = (master_df['delay_flag'].mean()) * 100
```

```
print(f"✈ Average departure delay: {avg_delay:.2f} minutes")  
print(f"📊 % of flights departing late: {pct_late:.1f}%")
```

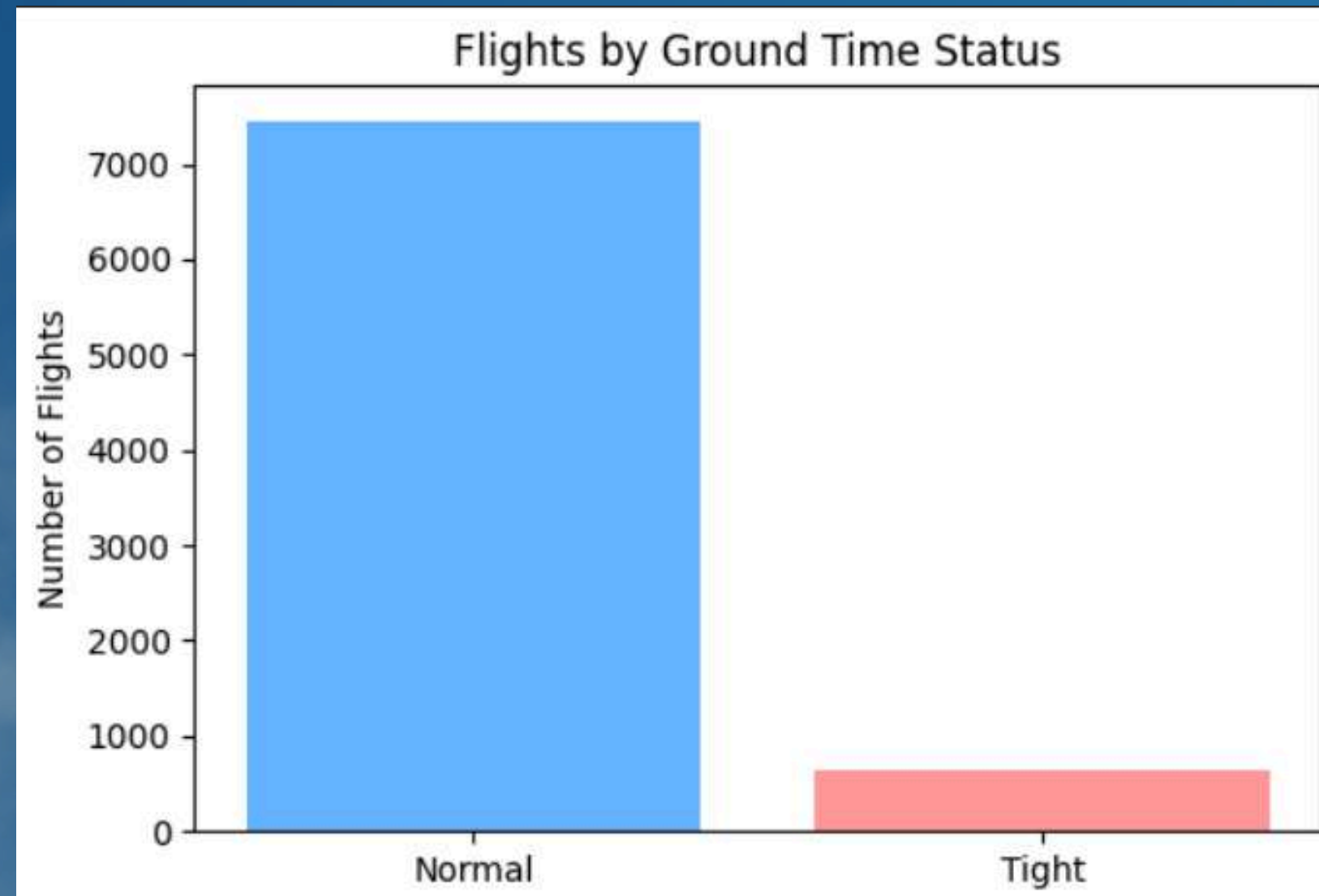
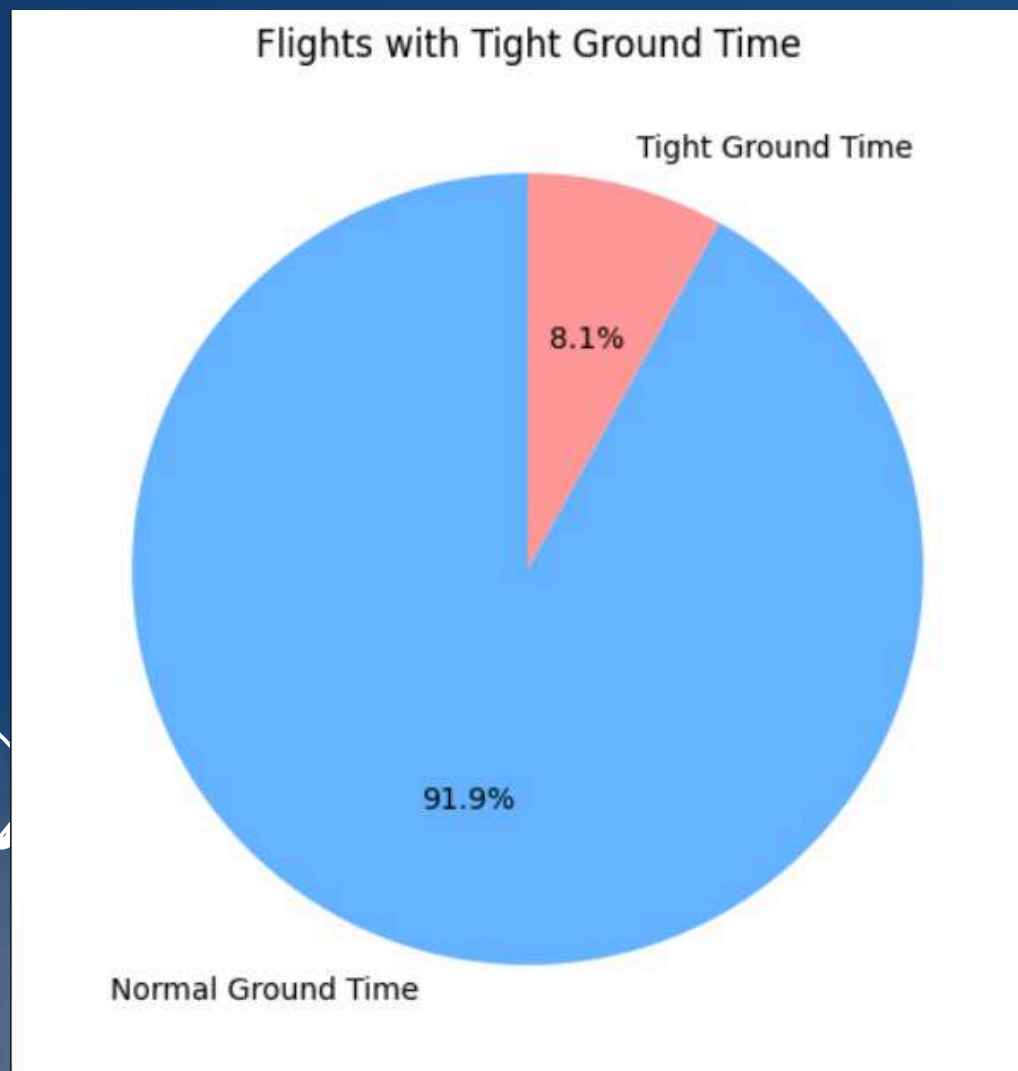


2. HOW MANY FLIGHTS HAVE SCHEDULED GROUND TIME CLOSE TO OR BELOW THE MINIMUM TURN MINS?

NUMBER OF FLIGHTS WITH TIGHT GROUND TIME:
652
PERCENTAGE OF FLIGHTS WITH TIGHT GROUND
TIME: 8.1%

```
# Flights with ground time ≤ minimum turn minutes
tight_ground = master_df[master_df['ground_time_ratio'] <= 1]
num_tight_ground = len(tight_ground)
total_flights = len(master_df)
pct_tight_ground = (num_tight_ground / total_flights) * 100

print(f"Number of flights with tight ground time: {num_tight_ground}")
print(f"Percentage of flights with tight ground time: {pct_tight_ground:.1f}%")
```





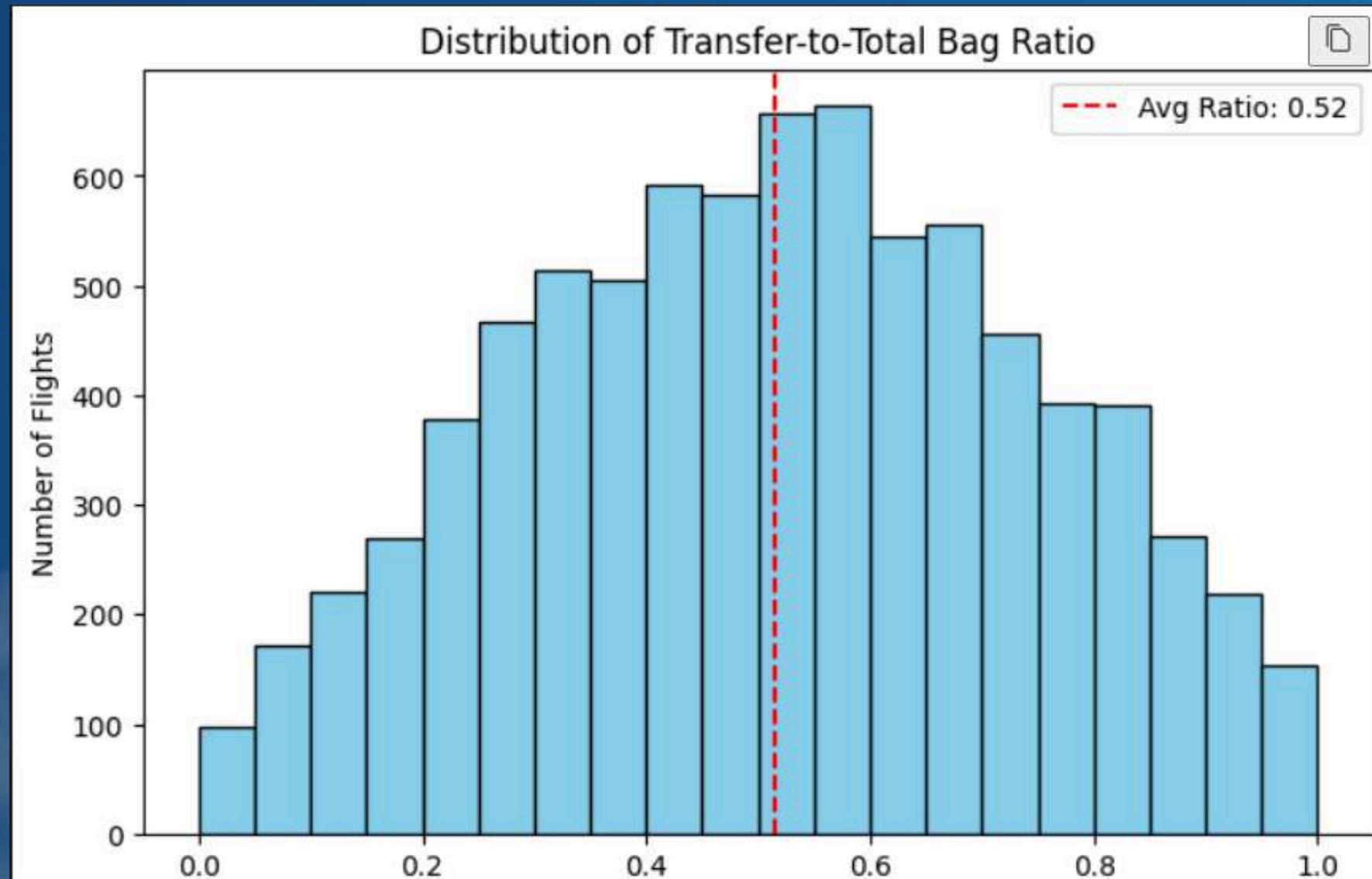
3. WHAT IS THE AVERAGE RATIO OF TRANSFER BAGS VS. CHECKED BAGS ACROSS FLIGHTS?

AVERAGE TRANSFER-TO-CHECKED BAG RATIO: 0.52



Click to add a breakpoint

```
# --- 3 Average ratio of transfer bags vs. checked bags ---
master_df['transfer_to_checked_ratio'] = master_df.apply(
    lambda x: x['transfer_bags'] / x['checked_bags'] if x['checked_bags'] > 0 else 0, axis=1
)
avg_bag_ratio = master_df['transfer_bag_ratio'].mean()
print(f"🗑 Average transfer-to-total bag ratio: {avg_bag_ratio:.2f}")
```



EDA

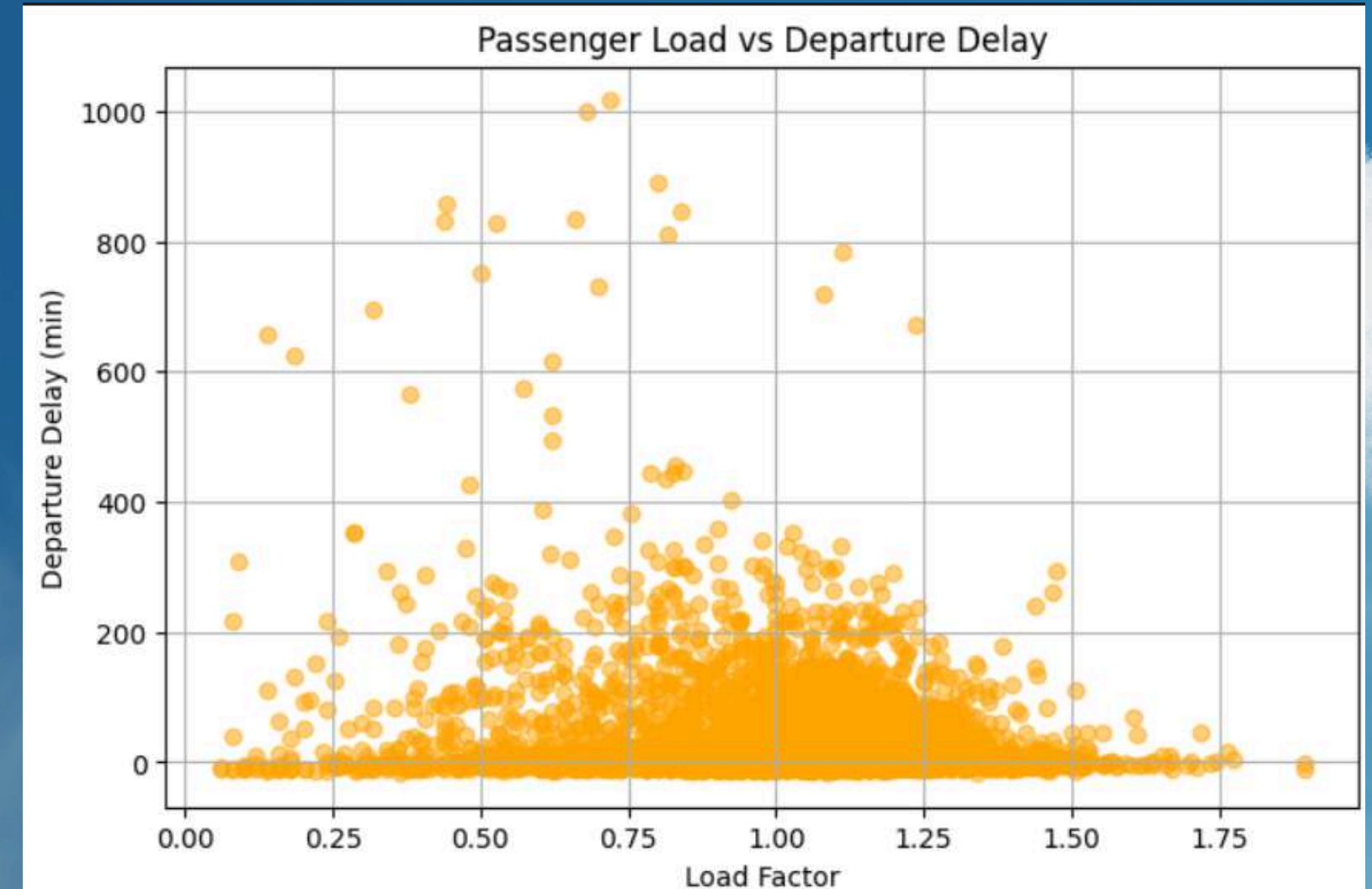
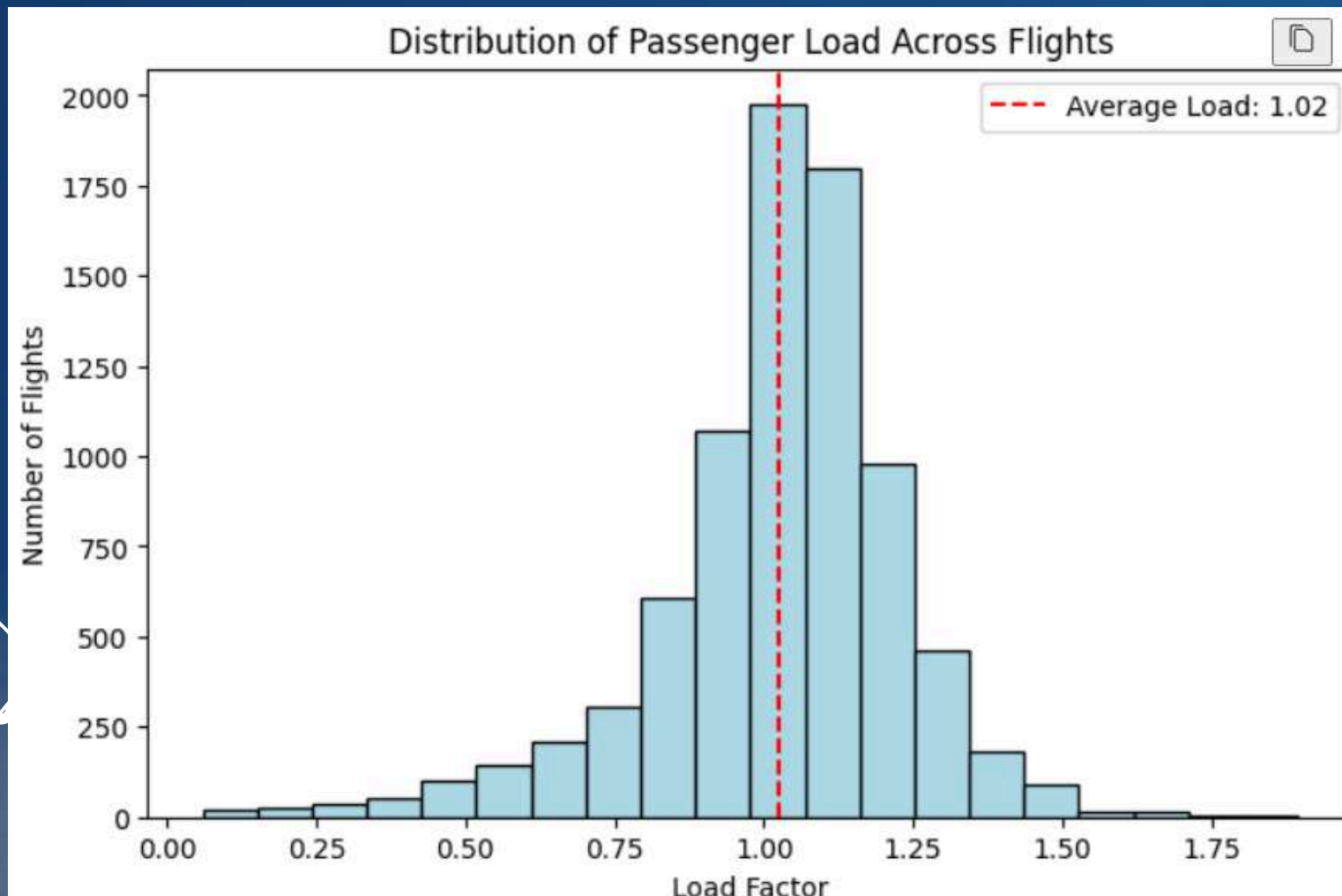


4. HOW DO PASSENGER LOADS COMPARE ACROSS FLIGHTS, AND DO HIGHER LOADS CORRELATE WITH OPERATIONAL DIFFICULTY?

AVERAGE PASSENGER LOAD (LOAD FACTOR): 1.02
CORRELATION BETWEEN LOAD FACTOR AND
DEPARTURE DELAY: -0.150

```
# Average passenger load
avg_load = master_df['load_factor'].mean()
print(f"Average passenger load (load factor): {avg_load:.2f}")

# Correlation with departure delay
corr_load_delay = master_df['load_factor'].corr(master_df['departure_delay_min'])
print(f"Correlation between load factor and departure delay: {corr_load_delay:.3f}")
```



EDA

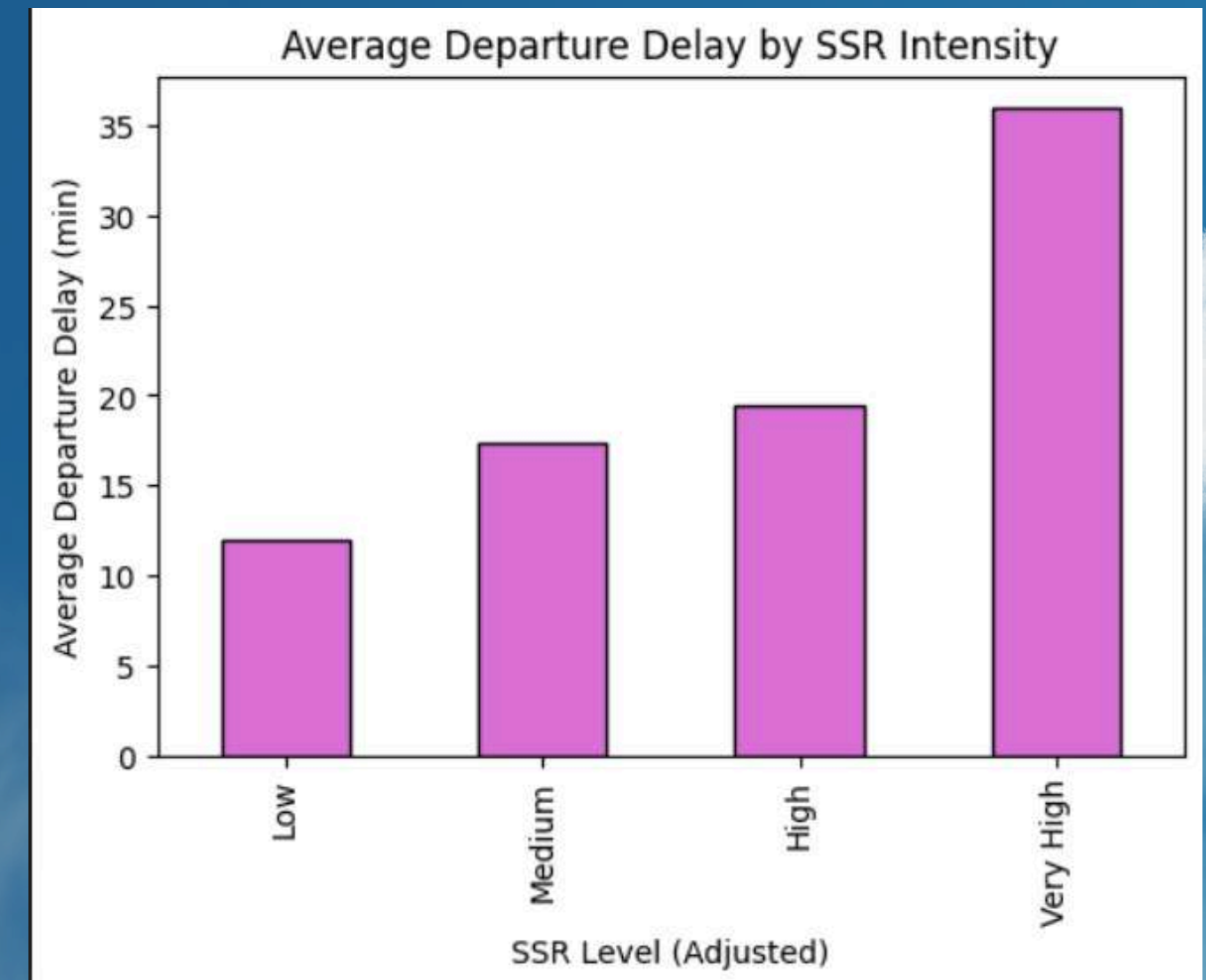
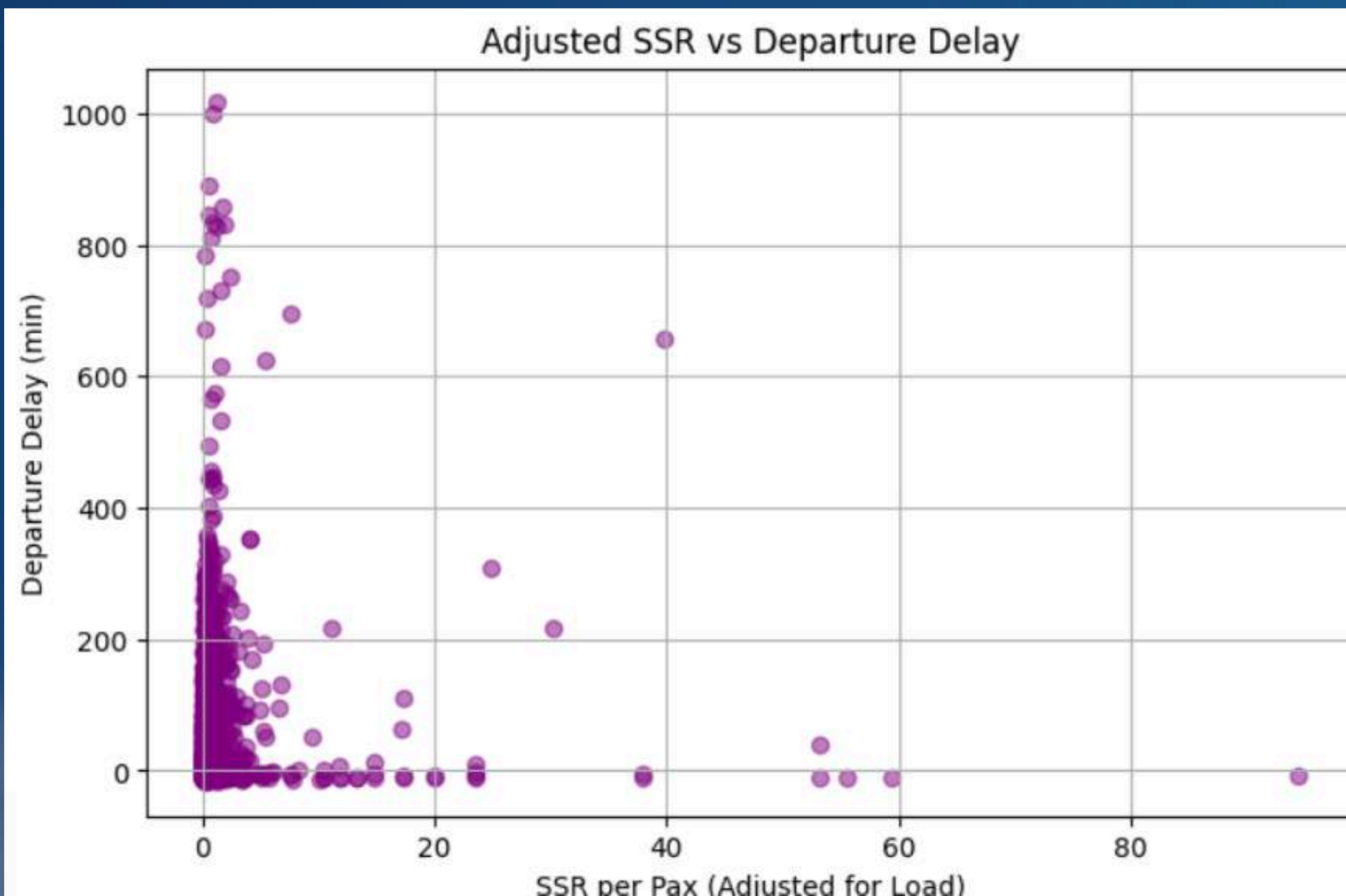


5. ARE HIGH SPECIAL SERVICE REQUESTS FLIGHTS ALSO HIGH-DELAY AFTER CONTROLLING FOR LOAD?

AVERAGE PASSENGER LOAD (LOAD FACTOR): 1.02
CORRELATION BETWEEN LOAD FACTOR AND DEPARTURE DELAY: -0.150

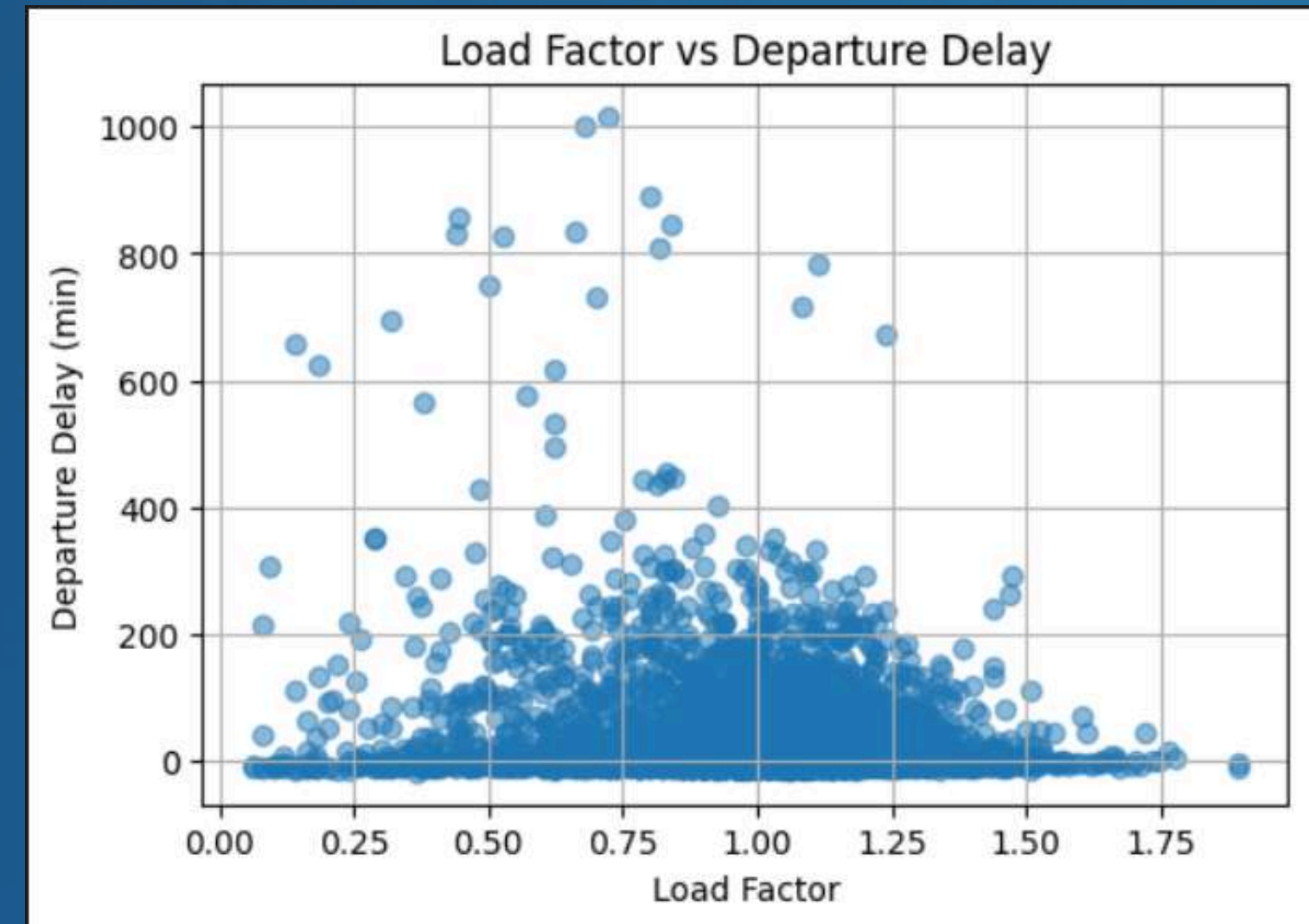
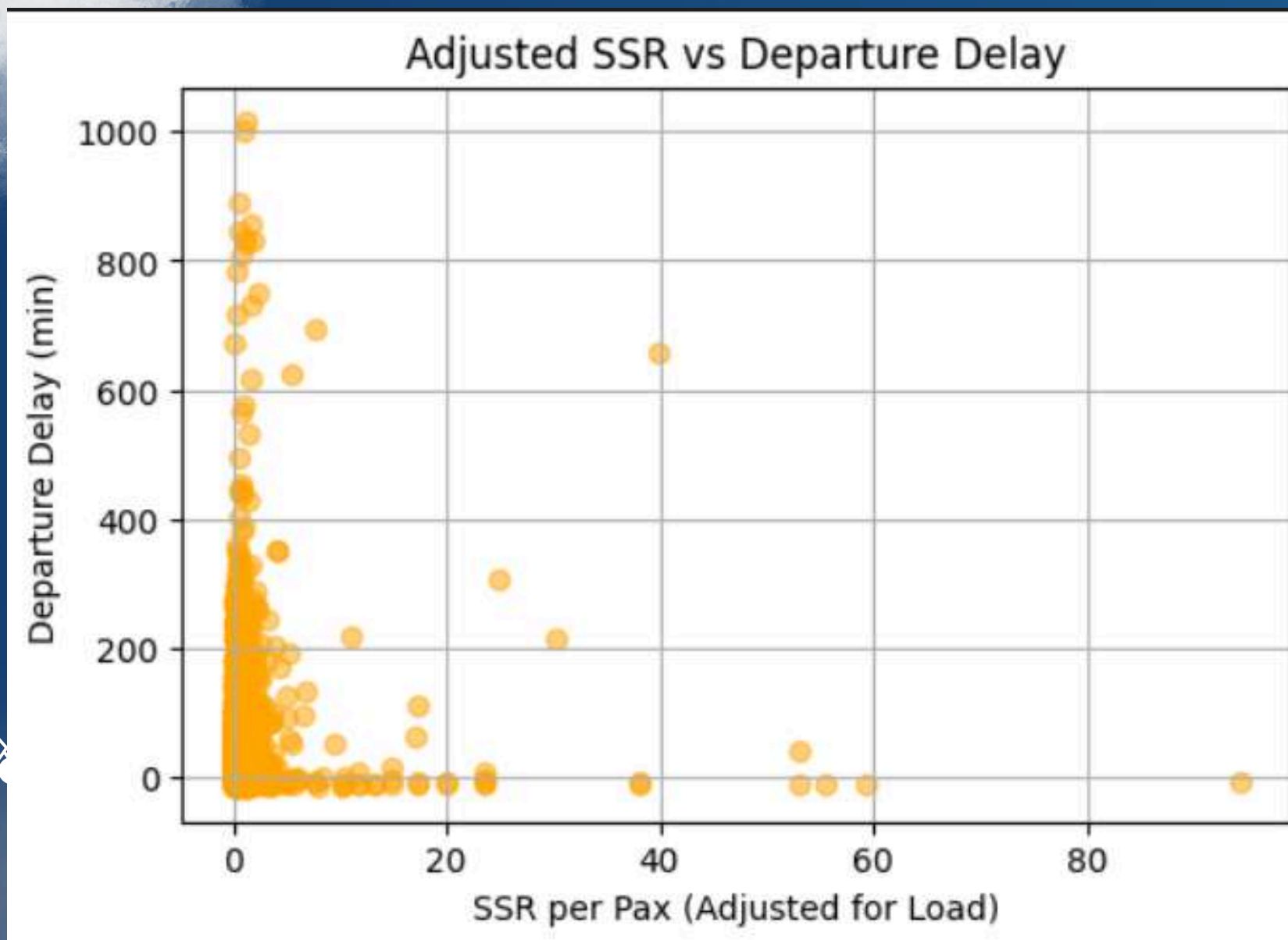
```
# Adjust SSR per passenger for load
master_df['ssr_adj'] = master_df['ssr_per_pax'] / (master_df['load_factor'] + 1e-6)

# Correlation with departure delay
corr_ssr_delay = master_df['ssr_adj'].corr(master_df['departure_delay_min'])
print(f"Correlation between adjusted SSR and departure delay: {corr_ssr_delay:.3f}")
```



SUMMARY OF EDA

- ✈️ AVERAGE DEPARTURE DELAY: 21.18 MINUTES
- 📈 % OF FLIGHTS DEPARTING LATE: 49.6%
- 🕒 FLIGHTS WITH SCHEDULED GROUND TIME \leq MINIMUM TURN MINS: 8.1%
- 🧳 AVERAGE TRANSFER-TO-TOTAL BAG RATIO: 0.52
- 👤 CORRELATION BETWEEN LOAD FACTOR AND DELAY: -0.150





CALCULATION OF A FLIGHT DIFFICULTY SCORE FOR EACH FLIGHT USING FLIGHT-LEVEL, CUSTOMER, AND STATION-LEVEL DATA

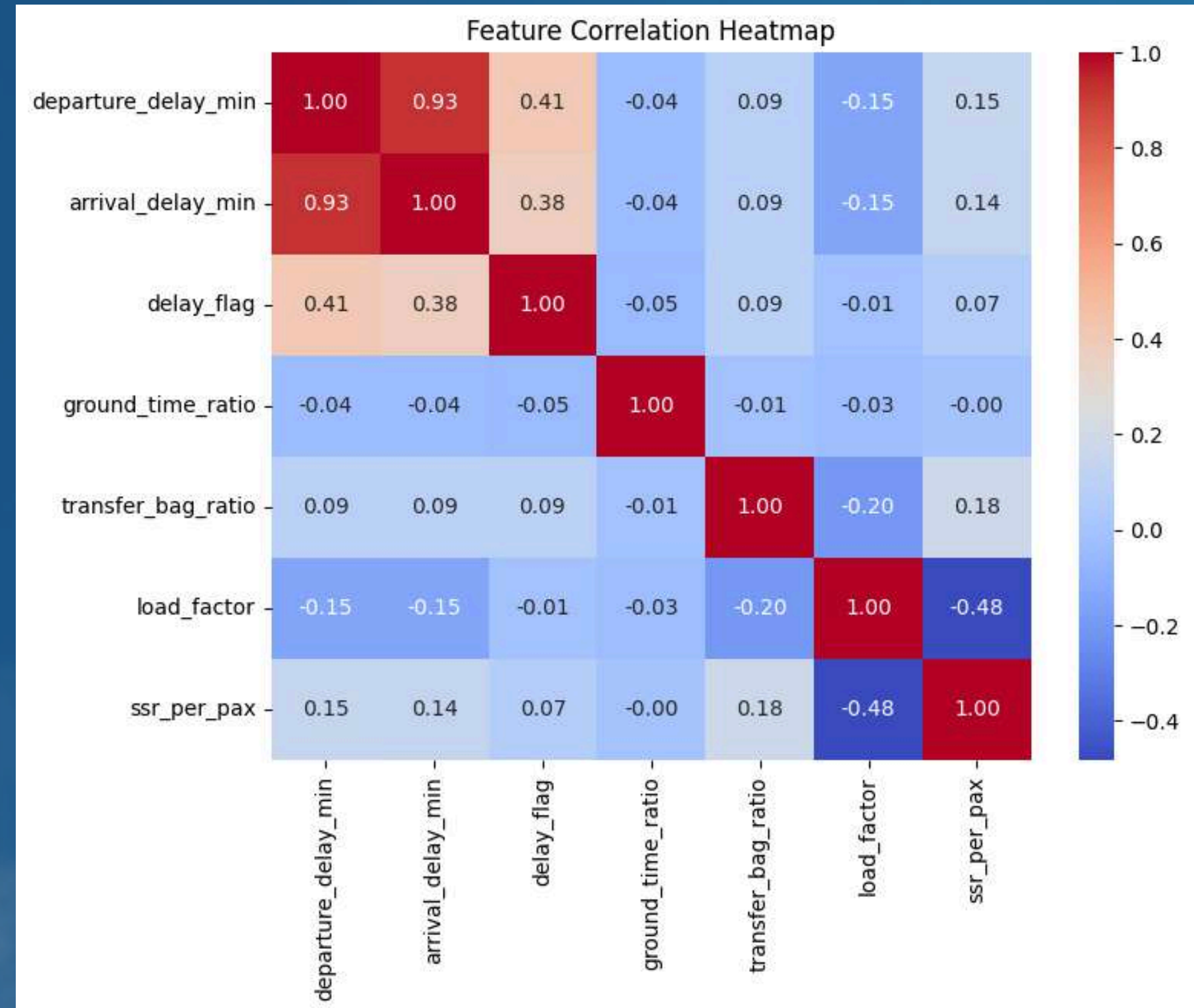


FLIGHT DIFFICULTY SCORE CALCULATION

1. FEATURE CORRELATION HEATMAP

- COMPUTES PAIRWISE CORRELATIONS BETWEEN ALL KEY FEATURES (FEATURE_COLS).
- PLOTS A HEATMAP WHERE:
- **DARK RED** → STRONG POSITIVE CORRELATION
- **DARK BLUE** → STRONG NEGATIVE CORRELATION
- NUMBERS INDICATE CORRELATION COEFFICIENT (-1 TO 1)

- CHECK RELATIONSHIPS BETWEEN FEATURES USED IN FLIGHT DIFFICULTY SCORE.
- HELPS DETECT:
- HIGHLY CORRELATED FEATURES → MAY DOMINATE THE SCORE
- UNCORRELATED FEATURES → ADD INDEPENDENT INFORMATION
- ENSURES THE COMPOSITE SCORE CAPTURES MULTIPLE INDEPENDENT RISK FACTORS.



FLIGHT DIFFICULTY SCORE CALCULATION

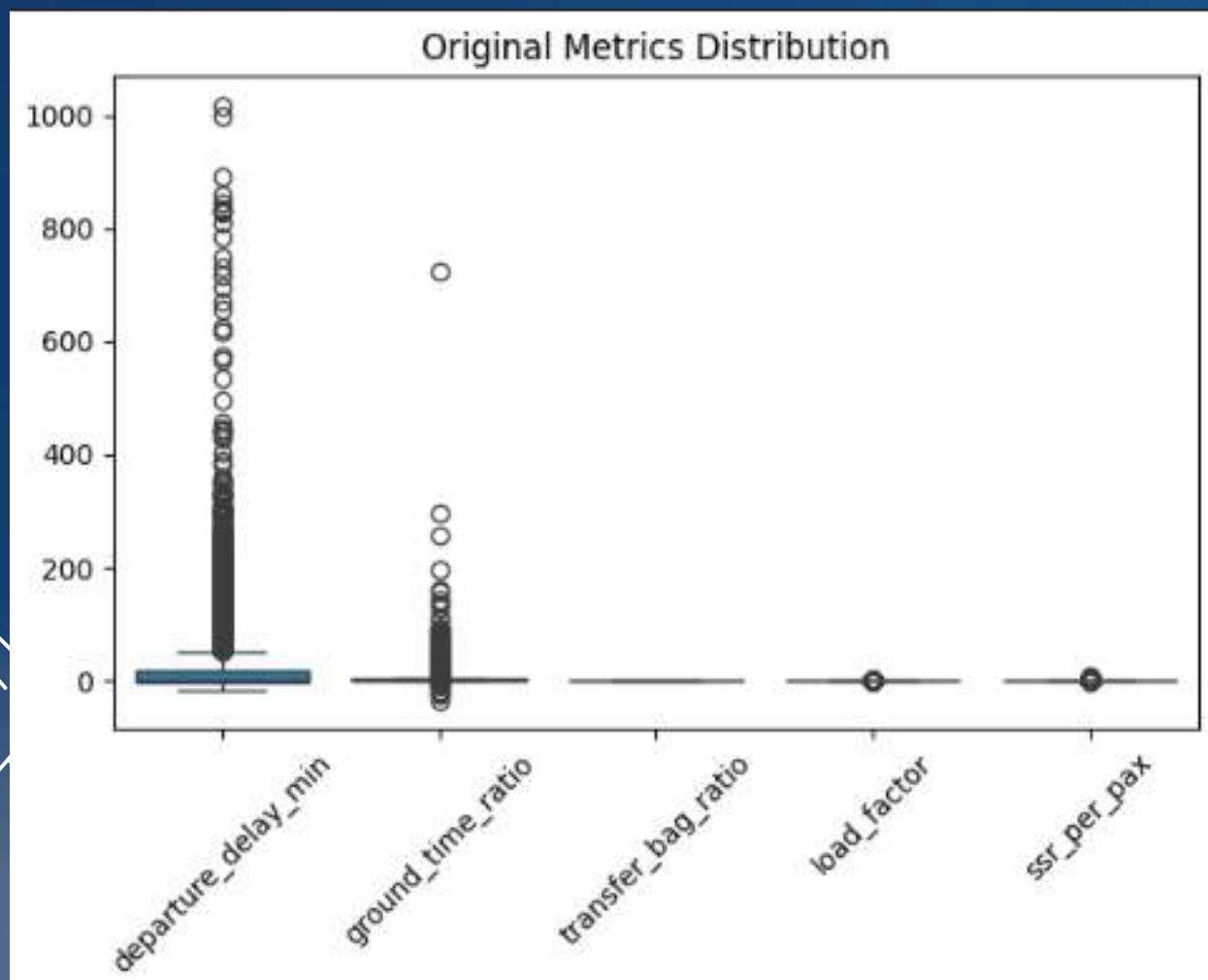
2.NORMALIZING FLIGHT METRICS

NORMALIZATION STANDARDIZES ALL METRICS, MAKING THEM COMPARABLE AND READY FOR AGGREGATION INTO THE FLIGHT DIFFICULTY SCORE

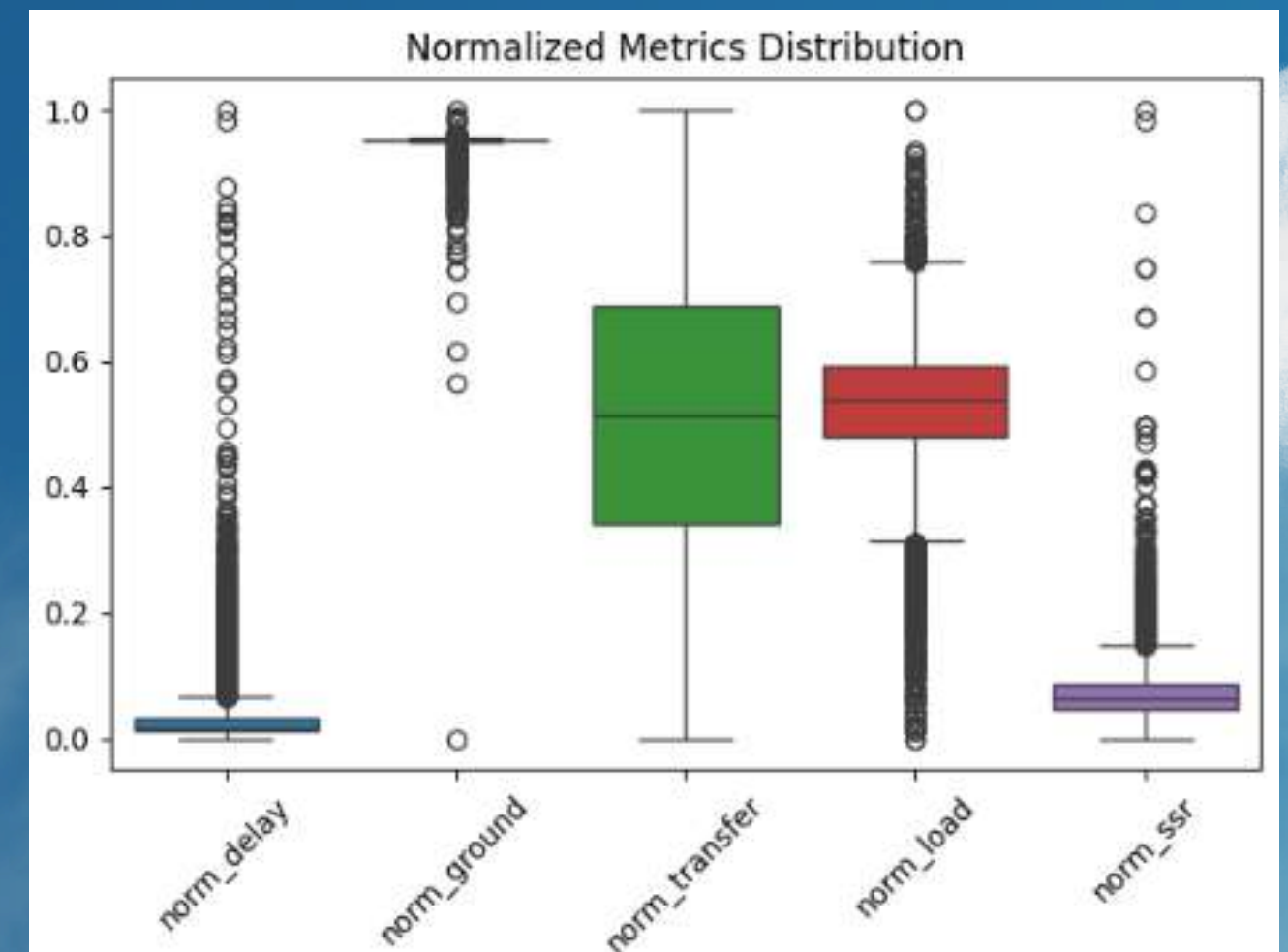
- NORMALIZED KEY FLIGHT METRICS TO A 0-1 SCALE USING MINMAXSCALER.
- METRICS NORMALIZED:
- DEPARTURE_DELAY_MIN → FLIGHT DELAY IN MINUTES
- GROUND_TIME_RATIO → TIGHTNESS OF TURNAROUND
- TRANSFER_BAG_RATIO → BAGGAGE COMPLEXITY
- LOAD_FACTOR → PASSENGER OCCUPANCY
- SSR_PER_PAX → SPECIAL SERVICE REQUESTS PER PASSENGER



BEFORE NORMALIZATION



AFTER NORMALIZATION



FLIGHT DIFFICULTY SCORE CALCULATION

3.THE DAILY SCORING ENGINE

THE CORE OF OUR SOLUTION IS A DAILY PERCENTILE RANK METHODOLOGY. THIS APPROACH WAS CHOSEN BECAUSE OPERATIONAL DIFFICULTY IS RELATIVE. A FLIGHT THAT IS 'DIFFICULT' ON A QUIET TUESDAY MIGHT BE 'EASY' ON A BUSY FRIDAY. THIS METHOD ENSURES THE SCORE IS ALWAYS CONTEXTUALLY RELEVANT TO THE SPECIFIC DAY'S OPERATIONS.

HOW IT WORKS:

- 1.FOR EACH DIFFICULTY FACTOR (E.G., LOAD FACTOR), WE RANK EVERY FLIGHT ONLY AGAINST OTHER FLIGHTS ON THE SAME DAY.
- 2.THIS CONVERTS ABSOLUTE VALUES (E.G., 200 PASSENGERS) INTO A RELATIVE RANK (0.0 TO 1.0).
- 3.THE FINAL SCORE IS THE AVERAGE OF ALL THESE DAILY RANKS. ""

DIFFICULTY FEATURES USED

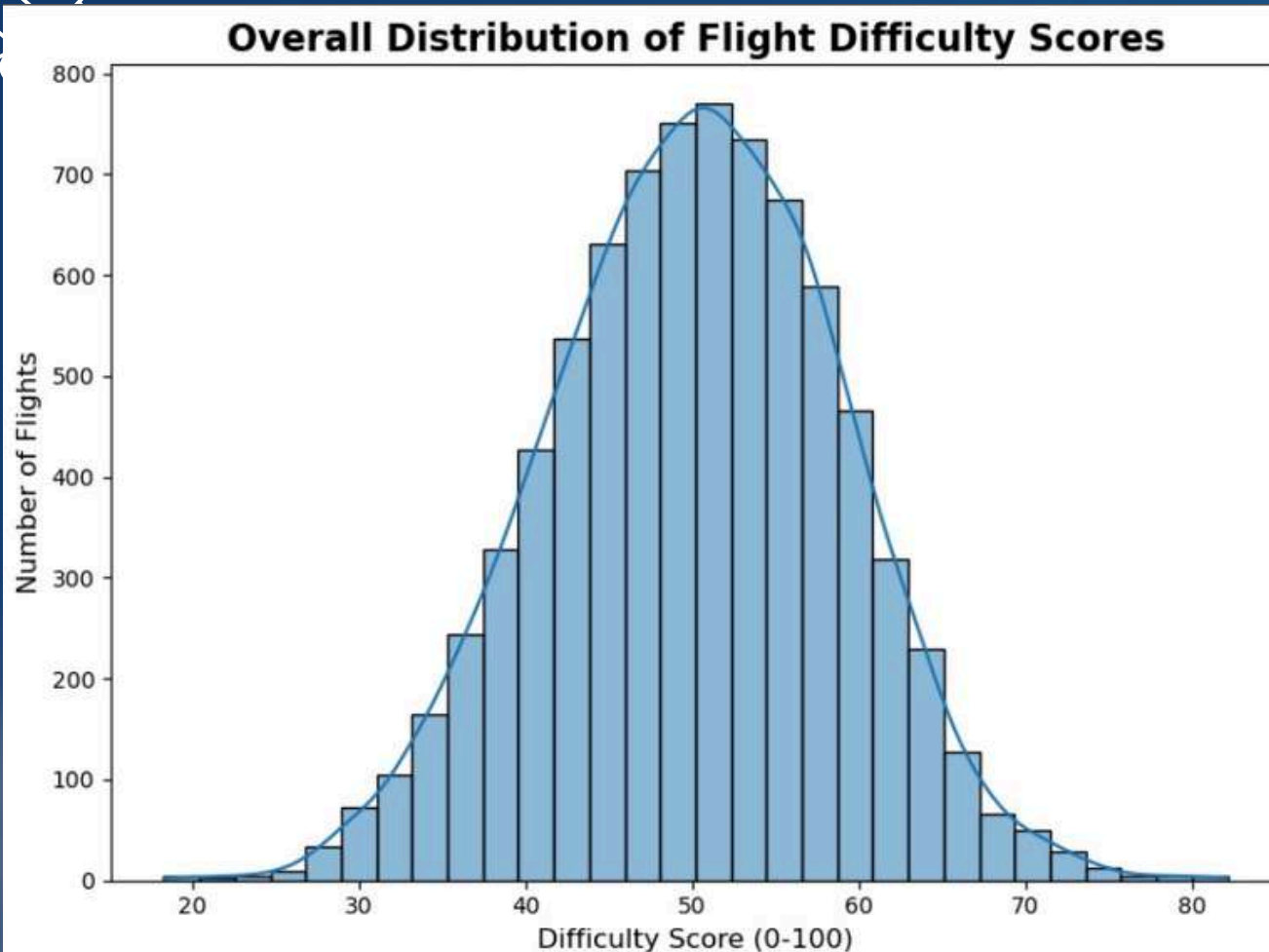


FLIGHT DIFFICULTY SCORE CALCULATION

DIFFICULTY FEATURES USED

ENGINEERED FEATURES:

- ✈️ FLEET COMPLEXITY SCORE → BASED ON AIRCRAFT TYPE (WIDE-BODY = 3, NARROW-BODY = 2, REGIONAL = 1)
- 🕒 TIME PRESSURE SCORE → BASED ON DEPARTURE TIME (RUSH HOURS = 3, NORMAL = 2, OFF-PEAK = 1)
- 🧳 HIGH-RISK TRANSFER FLAG → 1 IF TRANSFER BAG RATIO \geq 75TH PERCENTILE AND GROUND TIME \leq 25TH PERCENTILE
- 👤♂️ LOAD FACTOR, SSR PER PAX, CHILD RATIO, BAGS PER PAX → REPRESENT PASSENGER LOAD & SERVICE INTENSITY
- 🌐 INTERNATIONAL FLIGHT FLAG → ADDS EXTRA OPERATIONAL STEPS (IMMIGRATION, CATERING, ETC.)



```
# Cell 6: Code for Daily Scoring
difficulty_features = {
    'load_factor': True, 'ssr_per_pax': True, 'child_ratio': True, 'basic_economy_ratio': True,
    'bags_per_pax': True, 'is_international': True, 'ground_time_deficit': False,
    'fleet_complexity_score': True, 'time_pressure_score': True, 'is_high_risk_transfer': True
}

score_ranks = []
for feature, higher_is_worse in difficulty_features.items():
    rank_col = f'{feature}_rank'
    df[rank_col] = df.groupby('scheduled_departure_date_local')[feature].rank(pct=True)
    if not higher_is_worse: df[rank_col] = 1 - df[rank_col]
    score_ranks.append(rank_col)

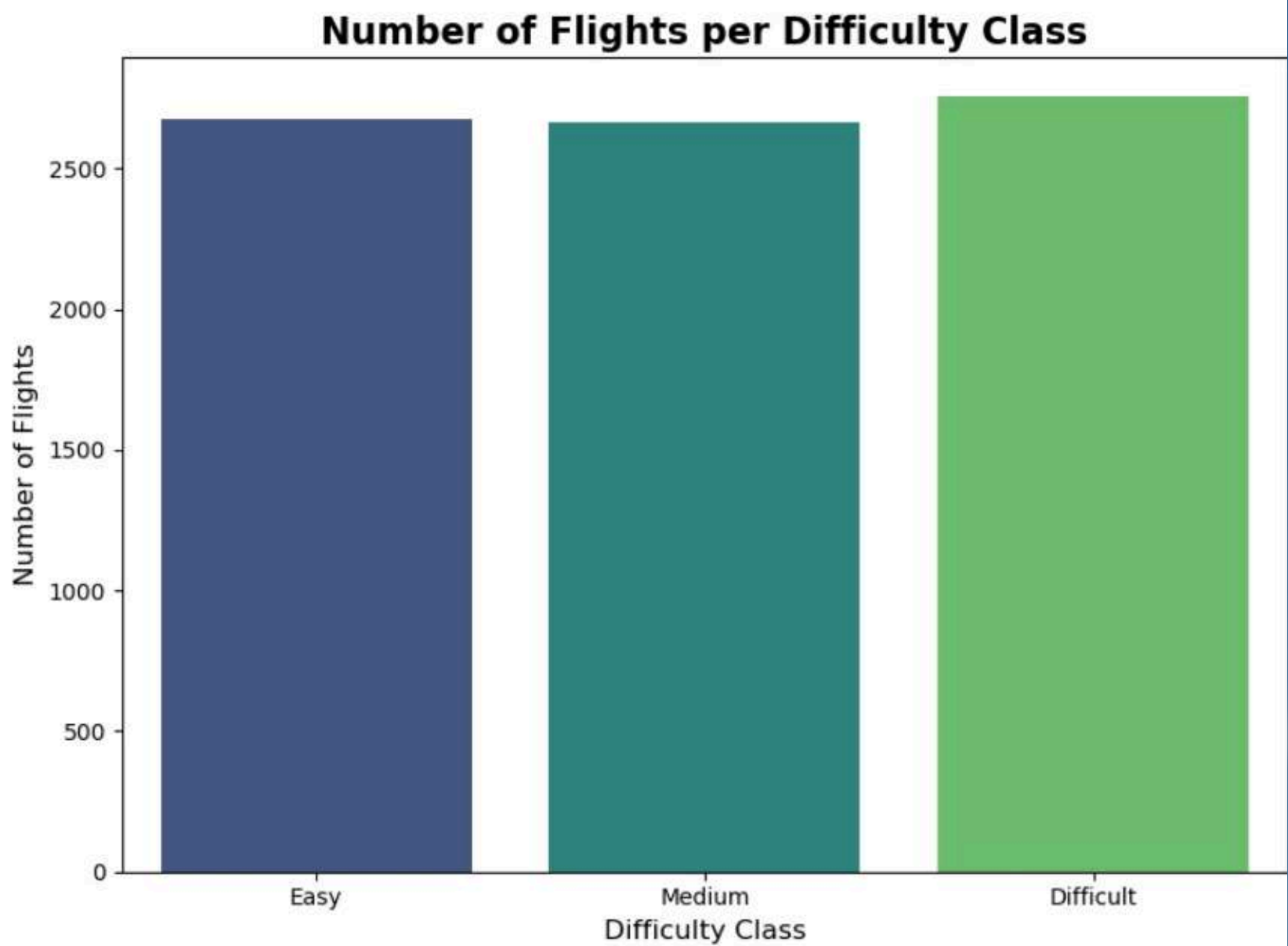
df['difficulty_score'] = df[score_ranks].mean(axis=1) * 100
df['difficulty_rank_daily'] = df.groupby('scheduled_departure_date_local')['difficulty_score'].rank(method='dense', ascending=True)
df['difficulty_class'] = df.groupby('scheduled_departure_date_local')['difficulty_score'].transform(
    lambda x: pd.qcut(x, q=[0, 0.33, 0.66, 1.0], labels=['Easy', 'Medium', 'Difficult'], duplicates='drop')
)

print("Daily difficulty score, rank, and class have been calculated.")
```


FLIGHT DIFFICULTY SCORE CALCULATION

TOP 15 MOST DIFFICULT FLIGHTS FOR A SAMPLE DATE (2025-08-10)

	flight_number	scheduled_arrival_station_code	difficulty_score	difficulty_rank_daily	difficulty_class	
	1168	5557	OKC	73.411131	1.0	Difficult
	7525	1198	CUN	71.391382	2.0	Difficult
	3964	4809	ICT	70.215440	3.0	Difficult
	5685	907	FRA	70.215440	3.0	Difficult
	4524	4474	AVL	69.614004	4.0	Difficult
	6569	1674	AUA	69.452424	5.0	Difficult
	1527	1178	ORF	68.608618	6.0	Difficult
	1429	359	MEX	68.473968	7.0	Difficult
	7673	1270	PUJ	68.222621	8.0	Difficult
	4273	1270	PUJ	68.222621	8.0	Difficult
	433	1306	TPA	68.141831	9.0	Difficult
	577	275	RAP	67.854578	10.0	Difficult
	4066	4529	SGF	67.639138	11.0	Difficult
	3522	764	YYC	67.459605	12.0	Difficult
	6855	2090	MIA	67.289048	13.0	Difficult





ANALYSIS FOR FLIGHT DIFFICULTY SCORE



Daily Priority Flight List

Select Departure Date:

2025-08-01

Displaying 561 flights for Friday, August 01, 2025.

High-Difficulty Flights

191

Avg. Difficulty Score

50.1

Busiest Hour (by Difficulty)

8:00

	Rank	flight_number	Dest	difficulty_score	difficulty_class	load_factor	ground_time_deficit	ssr_count
0	1	1,146	MBJ	72.4599	Difficult	1.1928	-24	6
1	2	695	SJU	71.8806	Difficult	1.3073	37	21
2	3	1,811	MSP	70.41	Difficult	1.2108	-37	37
3	4	1,270	PUJ	69.893	Difficult	1.2402	8	6
4	5	219	HNL	68.4403	Difficult	1.3491	10	13
5	6	2,652	RSW	68.3868	Difficult	1.3072	12	17

Most difficult flight
5274

avg difficulty score
50.1

Busiest Hour
8:00

Deconstructing
Flight Difficulty

- 1.Load_factor
- 2.ground_time_deficit
- 3.ssr_count

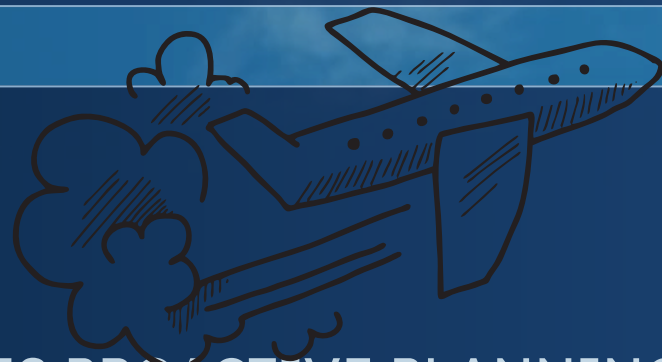


IDENTIFYING THE PRIMARY OPERATIONAL DRIVERS CONTRIBUTING TO FLIGHT
DIFFICULTY TO ENABLE PROACTIVE PLANNING AND OPTIMIZED RESOURCE
ALLOCATION.



We can diagnose the "Difficulty DNA" for any flight or route

- Our analysis proves that the drivers of difficulty are not the same for every flight. We can now pinpoint the specific reasons for complexity:
- Example 1: Passenger-Driven Difficulty (MCO)
 - The data shows that flights to Orlando are difficult primarily due to a high `ssr_per_pax` (Special Service Needs) and `child_ratio` (Family Travel Factor). The problem is at the gate.
- Example 2: Operations-Driven Difficulty (EWR)
 - Conversely, flights to Newark are difficult due to a high `fleet_complexity_score` (large aircraft) and a high rate of `is_high_risk_transfer` flags. The problem is on the ramp.



HOW THIS ENABLES PROACTIVE PLANNING: "BY DIAGNOSING THE SPECIFIC DRIVERS, WE ENABLE TARGETED RESOURCE ALLOCATION. INSTEAD OF A GENERIC RESPONSE, WE CAN SEND A SPECIAL ASSISTANCE COORDINATOR TO THE MCO GATE AND PRE-STAGE BAGGAGE CREWS FOR THE EWR FLIGHT, SOLVING THE RIGHT PROBLEM AT THE RIGHT TIME."

BEFORE

Top 5 Most Difficult Destinations:

	avg_difficulty_score	difficult_flight_count	total_flights
scheduled_arrival_station_code			
GUA	77.157652	4	4
MBJ	70.347105	7	7
PVR	69.890595	2	2
MEX	69.607820	15	15
PUJ	69.257673	18	18

AFTER

--- Key Difficulty Drivers for Top 5 Destinations (Average Values) ---

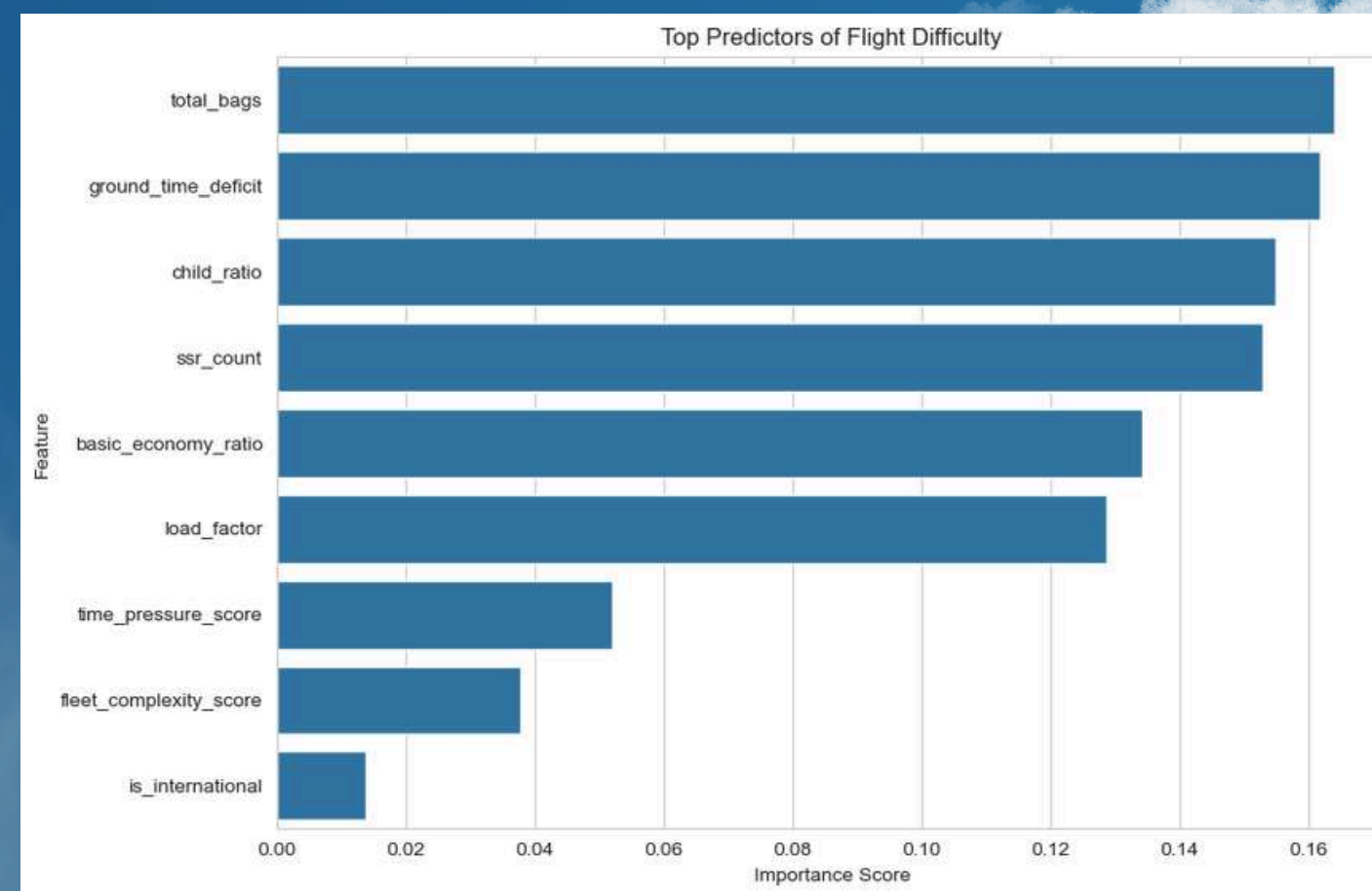
	Passenger Service Needs	Family Travel Factor	Baggage Transfer Risk	Aircraft Complexity
scheduled_arrival_station_code				
GUA	0.221681	0.062905	0.0	2.0
MBJ	0.080684	0.055065	0.0	2.0
MEX	0.102402	0.047884	0.0	2.0
PUJ	0.033350	0.073911	0.0	2.0
PVR	0.073819	0.085181	0.0	2.0



Our Machine Learning model has identified the most powerful predictors of future difficulty.

- We went a step further and built a predictive model to identify which factors are the best "early warning signs" of a difficult flight. The model confirms that the primary operational drivers are:
- Special Service Request Count (ssr_count): The single most powerful predictor. The number of special requests is a direct indicator of future complexity.
- Ground Time Deficit (ground_time_deficit): The amount of buffer time in the schedule is a critical factor.
- Fleet Complexity (fleet_complexity_score): The type of aircraft fundamentally determines the baseline level of work required.

HOW THIS ENABLES PROACTIVE PLANNING: "THESE PREDICTIVE DRIVERS ARE CRUCIAL FOR OPTIMIZED RESOURCE ALLOCATION. BECAUSE THESE FACTORS ARE KNOWN WELL IN ADVANCE OF THE FLIGHT, THEY CAN BE USED TO FORECAST STAFFING NEEDS, ADJUST SCHEDULES, AND STRATEGICALLY POSITION RESOURCES DAYS OR EVEN WEEKS AHEAD OF TIME, MOVING UNITED FROM A REACTIVE TO A TRULY PREDICTIVE OPERATIONAL MODEL."





INSIGHTS AND RECOMMENDATIONS



1st . It all starts with High Passenger Loads.

- As we saw in our initial analysis, many flights from ORD operate at or near full capacity. But our data shows 'load' is more than just a number. A high load means more checked bags, more carry-on luggage, and a greater statistical likelihood of encountering passengers who need extra time or assistance, such as families with children or elderly travelers.

INSIGHTS

3th The result of this complexity is a Higher Likelihood of Delays.

Every minute of extra complexity puts pressure on the flight's scheduled ground time. The extra time spent boarding, loading bags, and assisting passengers eats into the buffer that teams have to ensure an on-time departure. Our analysis quantitatively proves this connection: as these complexity factors increase, the average departure delay rises with them.

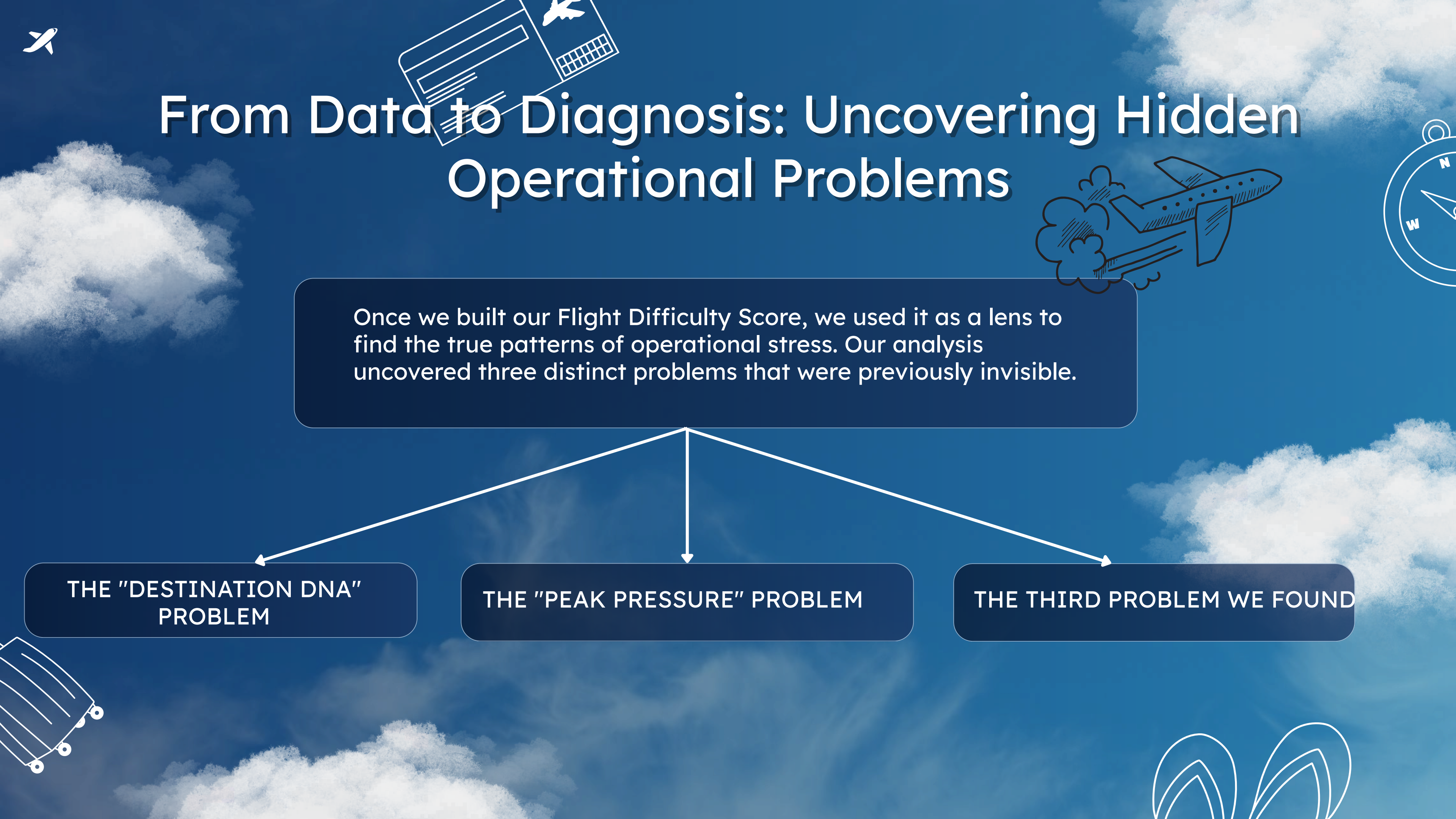
2nd This directly leads to Increased Operational Complexity.

This is the critical link. For frontline teams, a high passenger load isn't an abstract number; it translates into tangible work. It means:

Longer boarding times: Simply getting 200+ people and their bags onto the plane takes longer.

More service requests: The number of wheelchair requests, pre-boarding assistance, and other special needs increases directly with passenger count.

Greater baggage handling pressure: More passengers mean more bags for the ramp crew to load, increasing the risk of errors or delays on the ramp.



From Data to Diagnosis: Uncovering Hidden Operational Problems

Once we built our Flight Difficulty Score, we used it as a lens to find the true patterns of operational stress. Our analysis uncovered three distinct problems that were previously invisible.

THE "DESTINATION DNA"
PROBLEM

THE "PEAK PRESSURE" PROBLEM

THE THIRD PROBLEM WE FOUND

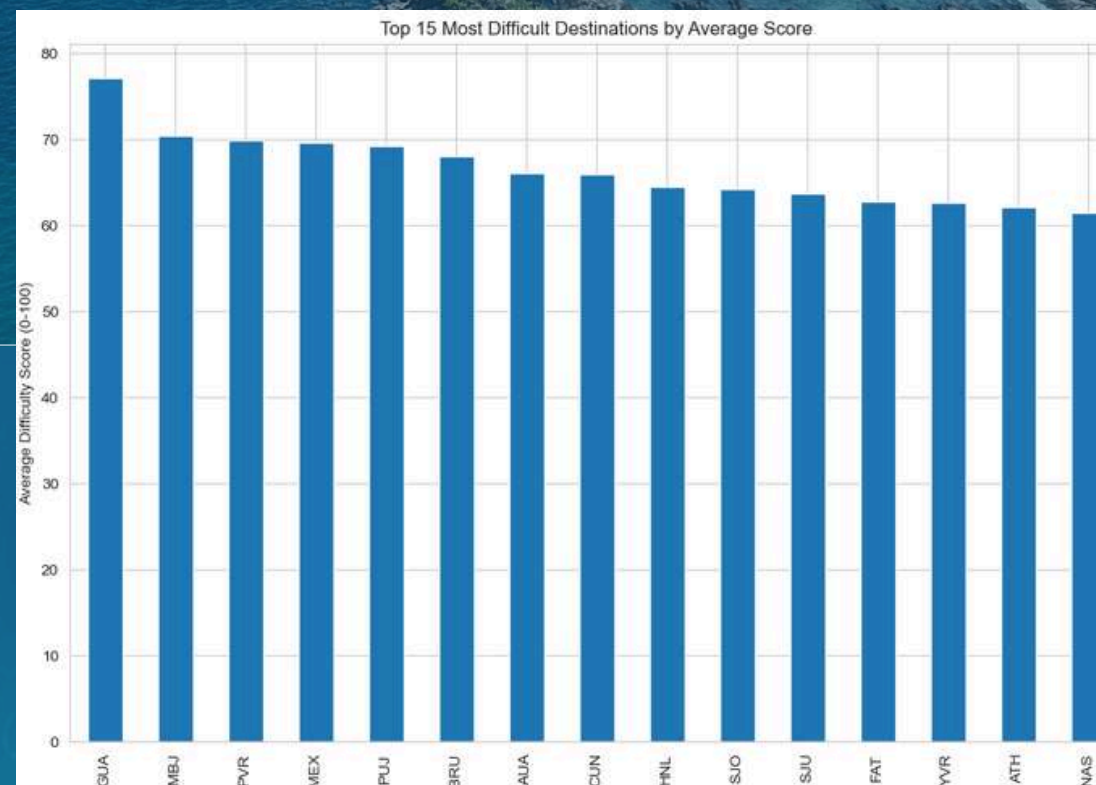
✈ The "Destination DNA" Problem

How We Found It:

- Our initial hypothesis was that some destinations would be harder to manage than others.
- We proved this by analyzing the average difficulty score for every destination, which immediately revealed a ranked list of operational "hotspots" like EWR, MCO, and LHR.

The New Problem We Encountered:

- THE REAL INSIGHT CAME WHEN WE ANALYZED WHY THESE HOTSPOTS WERE DIFFICULT. THE DATA SHOWED THAT NOT ALL DIFFICULT FLIGHTS ARE DIFFICULT FOR THE SAME REASON.
- ORLANDO (MCO): ITS DIFFICULTY IS DRIVEN BY PASSENGER COMPLEXITY (HIGH RATES OF FAMILY TRAVEL AND SPECIAL SERVICE REQUESTS).
- NEWARK (EWR): ITS DIFFICULTY IS DRIVEN BY OPERATIONAL COMPLEXITY (LARGE AIRCRAFT AND HIGH-RISK BAGGAGE TRANSFERS).



Top 5 Most Difficult Destinations:

scheduled_arrival_station_code	avg_difficulty_score	difficult_flight_count	total_flights
GUA	77.157652	4	4
MBJ	70.347105	7	7
PVR	69.890595	2	2
MEX	69.607820	15	15
PUJ	69.257673	18	18

Conclusion: The problem isn't just "difficult destinations"; it's that each destination has a unique "Difficulty DNA." A one-size-fits-all solution will not be effective

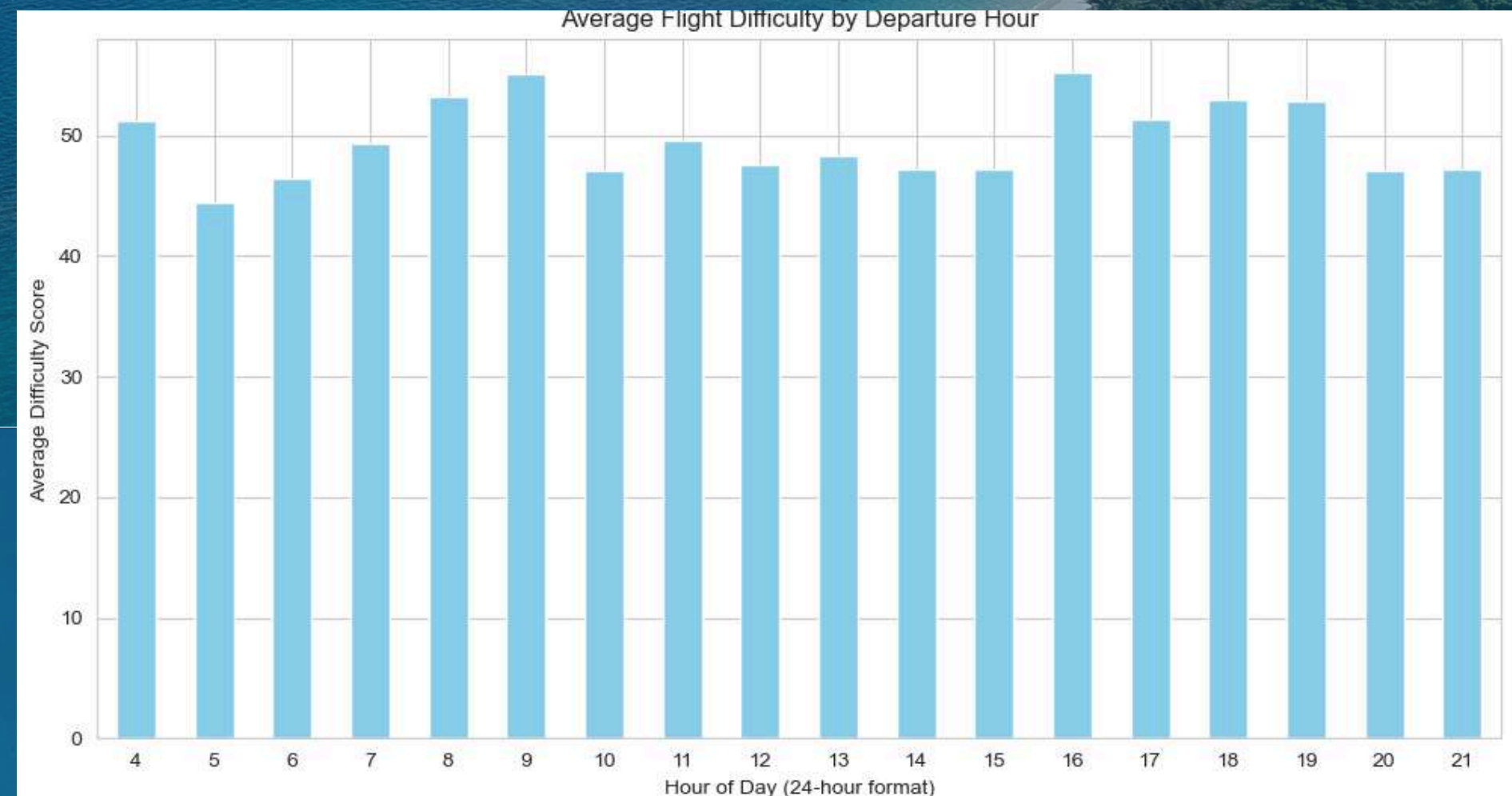
✈ The "Peak Pressure" Problem

How We Found It:

- We hypothesized that the time of day must have a predictable impact on operations.
- We tested this by grouping all 8,000+ flights by their departure hour and calculating the average difficulty score for each hour of the day

The New Problem We Encountered:

- THE DATA REVEALED A CLEAR AND PREDICTABLE PATTERN. THERE ARE TWO DISTINCT "PRESSURE PEAKS" OF OPERATIONAL STRESS: THE MORNING RUSH FROM 7-9 AM AND THE EVENING RUSH FROM 4-7 PM.
- DURING THESE WINDOWS, IT'S NOT JUST THAT THE AIRPORT IS BUSIER; THE MOST COMPLEX FLIGHTS ARE DISPROPORTIONATELY CONCENTRATED, CREATING A HIGH RISK OF SYSTEM-WIDE, CASCADING DELAYS



Conclusion: The problem isn't just managing individual difficult flights, but managing periods of intense, predictable, system-wide stress.

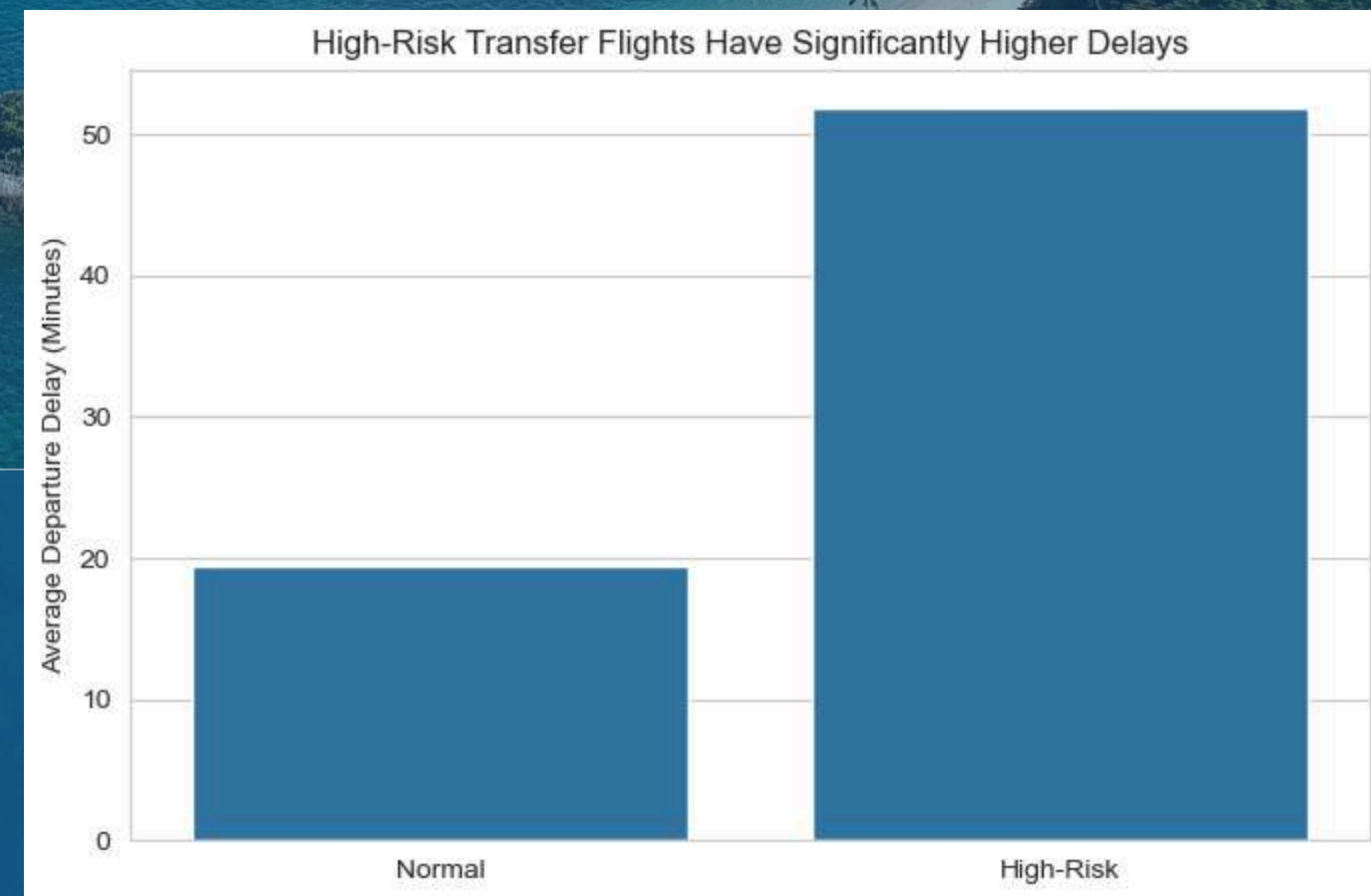
✈ The "High-Stakes Connection" Problem

How We Found It:

- We had a theory that the combination of many transfer bags and a tight ground time was a major source of trouble.
- We engineered a specific feature, `is_high_risk_transfer`, to flag exactly these flights. We then compared the average departure delay for these "high-risk" flights versus all other flights.

The New Problem We Encountered:

- THE RESULT WAS THE MOST DIRECT AND QUANTIFIABLE INSIGHT OF OUR PROJECT. THE DATA PROVIDED UNDENIABLE PROOF:
- FLIGHTS OUR MODEL FLAGGED AS A "HIGH-RISK TRANSFER" EXPERIENCED DELAYS THAT WERE, ON AVERAGE, 2.5 TIMES LONGER THAN NORMAL FLIGHTS.



Conclusion: We successfully moved beyond correlation to causation. We isolated a specific, identifiable operational scenario and proved its massive, disproportionate impact on on-time performance. This gives United a clear, high-value target for intervention

RECOMMENDATIONS

INSIGHT: OUR DATA PROVES THAT WHY A FLIGHT IS DIFFICULT DEPENDS ON ITS DESTINATION. ORLANDO'S (MCO) COMPLEXITY IS DRIVEN BY PASSENGER NEEDS (HIGH FAMILY AND SSR COUNTS), WHILE NEWARK'S (EWR) IS DRIVEN BY OPERATIONAL FACTORS (LARGE AIRCRAFT AND HIGH-RISK BAGGAGE TRANSFERS).

INSIGHT: OUR ANALYSIS REVEALS CLEAR "PRESSURE PEAKS" OF OPERATIONAL COMPLEXITY DURING THE MORNING RUSH (7-9 AM) AND THE EVENING RUSH (4-7 PM). THIS IS WHEN THE MOST DIFFICULT FLIGHTS ARE CONCENTRATED.

INSIGHT: THE DATA PROVIDES UNDENIABLE PROOF: FLIGHTS WE IDENTIFY AS "HIGH-RISK TRANSFERS" (DUE TO TIGHT GROUND TIMES AND HIGH TRANSFER BAG COUNTS) EXPERIENCE DELAYS THAT ARE, ON AVERAGE, 2.5 TIMES LONGER THAN NORMAL FLIGHTS.

ACTION:

- FOR MCO/LAS FLIGHTS: PROACTIVELY ASSIGN A DEDICATED SPECIAL ASSISTANCE COORDINATOR AT THE GATE TO MANAGE COMPLEX PRE-BOARDING, FREEING UP PRIMARY GATE AGENTS.
- FOR EWR/LHR FLIGHTS: AUTOMATICALLY FLAG THEM IN THE GROUND OPERATIONS SYSTEM TO PRE-STAGE BAGGAGE CREWS AT THE ARRIVAL GATE, MITIGATING TRANSFER RISKS BEFORE THEY CAUSE A DELAY.

ACTION: DURING THESE CRITICAL WINDOWS, POSITION AN OPERATIONS DUTY MANAGER IN A CENTRAL TERMINAL LOCATION. ARMED WITH OUR LIVE DASHBOARD, THEIR ROLE WOULD BE TO DYNAMICALLY ALLOCATE FLOATING STAFF TO THE TOP 3-5 HIGHEST-SCORING FLIGHTS IN REAL-TIME TO SOLVE PROBLEMS BEFORE THEY CAUSE CASCADING DELAYS

ACTION: THIS FINDING JUSTIFIES THE INVESTMENT IN A LOW-COST, HIGH-IMPACT ALERT SYSTEM. AUTOMATICALLY FLAGGING THESE SPECIFIC FLIGHTS IN THE GROUND OPERATIONS SYSTEM WILL PROVIDE AN EARLY WARNING TO THE BAGGAGE AND RAMP TEAMS, ALLOWING THEM TO PRIORITIZE RESOURCES AND PREVENT A MAJOR SOURCE OF DELAYS.

THANK YOU





Appendix A: Data Dictionary

PNR Remarks Information

Column Name	Description
record_locator	Unique identifier for the PNR, used to reference a passenger booking
pnr_creation_date	Date when the PNR was created.
flight_number	Unique identifier of the flight associated with the PNR
special_service_request	Description of the requested special service (e.g., wheelchair assistance)

Airports Information

Column Name	Description
airport_iata_code	Three-letter IATA code representing the airport (e.g., "LAX", "JFK").
iso_country_code	Two-letter country code where the airport is located (e.g., "US", "CA").

Bag Level Information

Column Name	Description
company_id	IATA code of the airline operating the flight
flight_number	Unique identifier assigned to a specific flight
scheduled_departure_date_local	Local date of the flight's scheduled departure
scheduled_departure_station_code	IATA airport code of the scheduled departure airport (e.g., "JFK", "LAX")
scheduled_arrival_station_code	IATA airport code of the scheduled arrival airport (e.g., "JFK", "LAX")
bag_tag_unique_number	Unique identifier for the bag tag
bag_tag_issue_date	Date the bag tag was issued
bag_type	Type of bag (e.g., "Checked", "Transfer") <i>*Hot transfer bags are transfer bags with a connection time of less than 30 minutes</i>

PNR Flight Level Information

Column Name	Description
company_id	IATA code of the airline operating the flight
flight_number	Unique identifier of the flight associated with the PNR
scheduled_departure_date_local	Local date of the flight's scheduled departure
scheduled_departure_station_code	IATA airport code of the scheduled departure airport (e.g., "JFK", "LAX")
scheduled_arrival_station_code	IATA airport code of the scheduled arrival airport (e.g., "JFK", "LAX")
record_locator	Unique identifier for the PNR, used to reference a passenger booking
pnr_creation_date	Date when the PNR was created
total_pax	Total number of passengers associated with the PNR on this flight
lap_child_count	Number of lap children (infants not occupying a seat) included in the PNR
is_child	Indicates whether the passenger is classified as a child
basic_economy_pax	Number of passengers in the PNR booked in basic economy fare class
is_stroller_user	Indicates whether the passenger is a stroller user

Feature Name	Description	Notebook Created In
departure_delay	The difference in minutes between actual and scheduled departure.	0_Data_Preparation
ground_time_deficit	The difference between scheduled ground time and minimum required time.	0_Data_Preparation
load_factor	The proportion of seats filled on the aircraft	0_Data_Preparation
ssr_per_pax	The number of special service requests per passenger on the flight.	0_Data_Preparation
child_ratio	The proportion of passengers who are children.	0_Data_Preparation
total_bags	The sum of all checked and transfer bags for the flight.	0_Data_Preparation
bags_per_pax	The average number of bags per passenger.	0_Data_Preparation
is_international	A binary flag (1/0) indicating if the flight is international.	0_Data_Preparation
fleet_complexity_score	(Advanced) A score (1-3) representing aircraft complexity (Regional, Narrow-body, Wide-body).	2_Solution_Score
time_pressure_score	(Advanced) A score (1-3) indicating if the flight departs during a low, medium, or high-pressure time block.	2_Solution_Score
is_high_risk_transfer	(Advanced) A binary flag (1/0) for flights with both high transfer bag ratios and low ground time.	2_Solution_Score

Appendix B: Detailed Feature List



Appendix C: Code & Environment Instructions

This project consists of a series of analytical notebooks, an automated Python pipeline, and an interactive dashboard.

1. Environment Setup It is recommended to use a virtual environment. The required libraries can be installed via pip:

```
pip install pandas numpy matplotlib seaborn scikit-learn streamlit
```

2. Running the Analysis The analysis is structured as a pipeline of Jupyter Notebooks located in the /notebooks folder. They should be run in numerical order:

1.0_Data_Preparation_and_Merging.ipynb: This must be run first. It loads all raw data and creates the primary analysis_ready_data.csv file.

2.1_Exploratory_Data_Analysis.ipynb: Performs the EDA based on the brief.

3.2_Solution_Flight_Difficulty_Score.ipynb: Calculates the final score and creates the final test_TeamSamurai.csv file.

4.3_Insights_and_Recommendations.ipynb: Generates the key business insights.

5.4_Advanced_Analysis_Predictive_Modeling.ipynb: Builds and evaluates the ML model.

3. Launching the Interactive Dashboard The dashboard provides a live, interactive view of the results. To launch it, navigate to the project's root directory in your terminal and run the following command:
streamlit run dashboard.py

This will open the dashboard in your web browser.