Aviation Data Analytics in the Cloud

Esteban Ayala Arroyo
Computer Science
Advisor: Jeffrey Duffany, Ph.D.
Polytechnic University of Puerto Rico
Graduate Project EXPO, May 2025

Abstract. *Maintenance, Repair, and Overhaul (MRO) in commercial aviation refers to the practices and systems airlines use to ensure aircraft remain airworthy, efficient, and compliant with regulations over time. It's a broad term, covering everything from routine inspections to unscheduled repairs and full component overhauls. In daily operations, MRO shows up in ways that might seem invisible to passengers—like a component swap during a turnaround or the tracking of wear patterns through maintenance logs. Airlines rely heavily on reliability data, both historical and real-time, to anticipate failures before they happen and to schedule work without disrupting flight schedules. This data—collected from flight hours, shop findings, onboard sensors, even crew reports—feeds into reports that are often requested by customers or suppliers. Sometimes it's messy, incomplete, or needs stitching together from different sources. But when it works, it helps close feedback loops, improve parts sourcing, and meet contractual obligations.*

Key Term — *Commercial aviation, Airlines, Maintenance*

INTRODUCTION

Maintenance, Repair, and Overhaul (MRO) operations are shifting with the times, blending traditional technical work with advanced digital tools [1]. These changes promise a lot, more reliability, faster turnaround, fewer unexpected failures. And yes, ideally, lower costs. But it's not just about installing a new platform and calling it progress. There's more to it.

Take data mining, for example. It's become one of those quiet game-changers. The more I read and explored, the clearer it became that MRO—Maintenance, Repair, and Overhaul—is one of the most promising areas for applying data mining in aerospace industry [2]. By sorting through massive sets of maintenance logs, sensor outputs, and inspection records, data mining helps spot patterns that the human eye might miss [3]. A tiny anomaly in engine vibration today could point to a costly issue six weeks from now. That kind of foresight? It matters. But it also raises a new challenge: people is needed for interpret what that data means and not just in theory.

That's where the workforce comes in. The skill set is evolving. It's no longer enough to understand how a hydraulic system works. You also need to navigate digital maintenance platforms, interpret analytics dashboards, maybe even write a basic query. It's not necessarily harder. Just different. And maybe a bit intimidating if your training didn't include those tools.

Which is part of the problem. A lot of current technicians are learning on the flight. Ideally, though, data literacy should start early. In tech schools, in apprenticeships, even in the first few months on the job. And it shouldn't stop there. These tools keep changing what works this year might be outdated next. So, ongoing training isn't optional anymore; it's the baseline.

And yet, all this talk of tools and platforms still depends on something deceptively simple: data. Reliable, complete, clean data. Without it, even the best algorithms produce noise. And let's be honest MRO records aren't always perfect. There are gaps, inconsistencies, even flat-out errors. Data mining can help fill in the blanks, sure. But only to an extent. At some point, someone still needs to double-check that what's in the system actually reflects reality.

Maybe the most valuable thing to do is create more opportunities for hands-on learning—real-world scenarios where techs can work through

complex problems using both their tools and their judgment. That's where things really start to click not just learning the system but understanding when and why to trust it. Or question it.

BACKGROUND

I didn't originally set out to focus my project on aviation maintenance, but looking back, it almost feels inevitable. After spending a fair amount of time in the aerospace industry particularly in areas tied to aircraft maintenance and supplier coordination. I started noticing a pattern that was, honestly, hard to ignore. Time and time again, we'd receive component data from suppliers that was incomplete, inconsistent, or just flat-out unreliable. Sometimes, critical maintenance records were missing timestamps, or there were discrepancies. And while everyone more or less learned to work around these gaps, it always struck me as a fragile way to operate, especially in an industry where precision isn't optional. That constant friction with fragmented data wasn't just frustrating; it started to feel like a puzzle I couldn't stop thinking about.

That was really the spark. I began wondering whether the tools and methods I'd been learning, especially in data mining, could be used to improve this situation. Not in a sweeping, industry-changing way (at least not immediately), but in small, practical ways. Could train systems spot patterns in these gaps? Predict where missing entries might exist? Even just flagging inconsistencies before they escalate would already be a step forward. The idea stuck with me. And when it came time to choose a topic for this project, it just made sense to go back to that recurring problem and see what could be done.

It's a space with enormous potential but also significant complexity. Maintenance logs, sensor outputs, compliance documents, they're all part of a vast and often messy ecosystem. And yet, there's so much to gain from cleaning it up: predictive fault detection, optimized repair schedules, fewer delays, improved safety. It's the kind of problem that feels worth solving—not just because it's challenging, but because the payoff matters.

Another reason this topic appealed to me was the chance to test classroom knowledge in a domain that demands more than theoretical understanding. Concepts like classification, clustering, and anomaly detection are powerful on paper, but how do they hold up when applied to a noisy, high-stakes environment like aircraft maintenance? I was curious, maybe even a little skeptical, about how well these tools would translate. That curiosity became part of my motivation: not just to apply what I've learned, but to test it in the real world, where things rarely behave like the textbook examples.

And lastly, there's the broader context that can't be ignored. The aerospace industry is undergoing a major digital transformation, and maintenance is right at the center of it. Companies are investing more and more in data infrastructure, AI, and predictive analytics. So, choosing this topic also felt like a way to align myself with where the field is headed. There's growing demand for professionals who can straddle both worlds—who understand the technical side of data science and also grasp the operational realities of aerospace. I'm hoping this project helps me build toward that kind of role.

PROBLEM

In the world of commercial aviation, maintaining aircraft safely and efficiently depends on far more than just skilled technicians and good equipment. Increasingly, it depends on data—specifically, the ability to gather, interpret, and act on it. That sounds straightforward, but in practice, it rarely is. MRO operations generate huge amounts of data: logs, sensor outputs, inspection notes, supplier documentation. Yet too often, this data arrives incomplete, inconsistent, or poorly formatted, especially from third-party sources. I've seen it myself, more than once. A component record shows up with missing install dates or unclear service history, and suddenly you're digging through data to piece together something that should've been obvious.

This isn't just frustrating, it's risky. MRO teams need reliable data to make informed decisions,

whether it's scheduling inspections or identifying a recurring fault pattern. And as the industry moves toward more predictive, data-driven maintenance strategies, the gap between what we *need* from our data and what we *actually* get is becoming harder to ignore.

That's where data mining comes in. The techniques I've learned—classification, pattern detection, outlier analysis, they aren't just academic tools. They can be applied directly to these messy, real-world datasets to extract meaning, find gaps, and flag potential issues before they become serious. But using them effectively in aviation also requires an understanding of how maintenance works, how aircraft systems are documented, and how MRO decisions are made in practice.

What makes this especially challenging, though, is that the complexity of aerospace data isn't just technical, it's also human. Technicians, engineers, and suppliers all interact with these systems differently. Mistakes happen. Information gets lost. So, working with this kind of data requires not only technical skills but also a level of patience, curiosity, and critical thinking that, frankly, you can't really teach in a single classroom.

This project is an attempt to bring those threads together. To take the data mining concepts I've studied and apply them in a way that respects both the technical rigor of aerospace and the imperfect, often unpredictable nature of real-world MRO environments.

Equipment & Materials

When I first started thinking about building out an end-to-end data platform, especially something that could handle raw, semi-structured data from external systems, one thing became immediately clear: no single tool could do it all. And honestly, that's not a bad thing. Each tool has its strengths, and the way they fit together is kind of the whole point. So, let's walk through the setup in Figure 3 & 4. You can make reference to Figure 1 to full solution diagram.

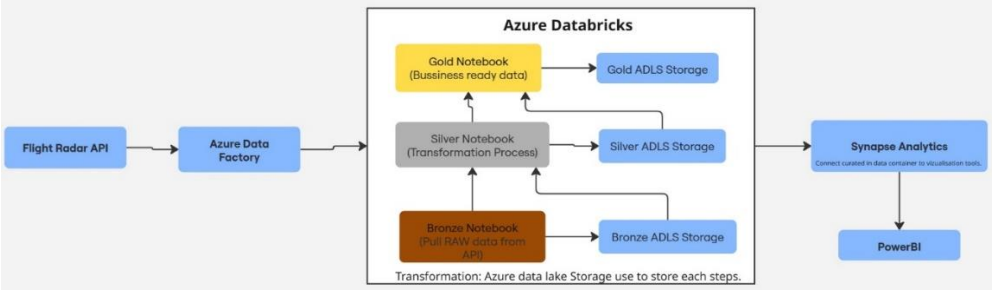

Figure 1
Medallion Architecture

Everything begins at the *source*, in this case an API endpoint. Nothing fancy—just data coming from a system somewhere, likely messy, definitely not analytics-ready. This is where Azure Data Factory steps in Figure 3. Data Factory isn't analyzing anything, and it's not doing complex transformations either. What it is doing is extracting that data and pushing it into your data lake, specifically, into Azure Data Lake Storage (ADLS).

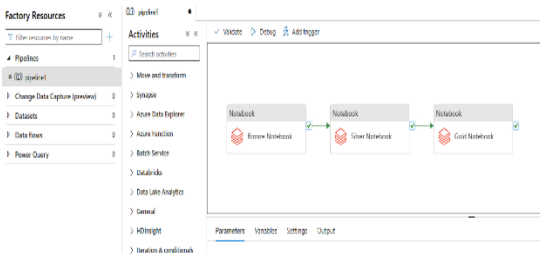

Figure 2
Azure Data Factory Orchestration

Now, here's where it starts to get more interesting. This is where Azure Databricks comes in.

The data lands in what's labeled as the bronze layer Figure 4. That's the raw stuff—pretty much untouched. It's often messy, maybe duplicated, sometimes inconsistent. But that's okay, because we're not trying to perfect it here. We're just capturing everything. Think of it as a digital shoebox where all your receipts go before you organize them.

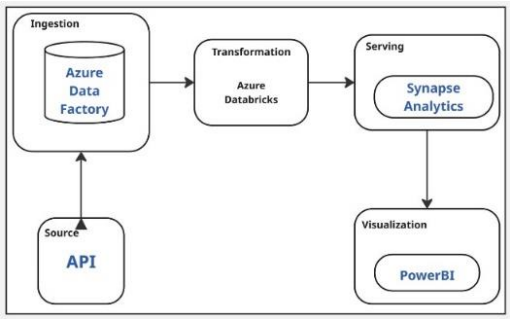

Figure 3
High Level ETL Diagram

Then to the silver layer. Databricks handles the heavy lifting: cleansing, filtering, joining datasets, enriching fields, all of it. It's a powerful platform that supports Spark-based processing, which makes it great for big data transformations. I like to think of it as the brain of this architecture, it's where most of the logic lives. This silver layer is structured, consistent, and usable. It's data that's finally starting to make sense.

The gold layer comes next. It's refined, business-ready data—typically aggregated or modeled in a way that's tailored for specific analytics needs. You don't want to clutter this layer with noise. This is what gets exposed to consumers.

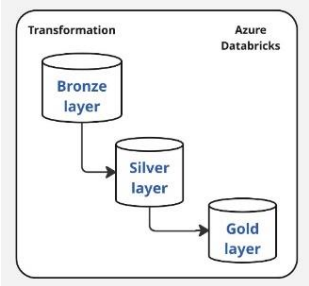

Figure 4
Databricks Transformation Layers

Which brings us to Synapse Analytics. Once the data is transformed and curated, it's passed to Synapse Figure 5.

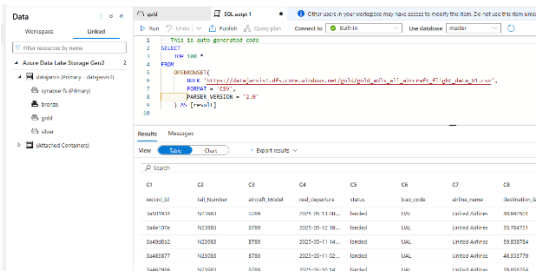

Figure 5
Synapse Data Loaded

Synapse acts as the serving layer. It's where queries happen, dashboards get built, and reports are powered. It's well-suited for large-scale analytics—whether it's running SQL queries over lake data or driving Power BI dashboards. And what's nice is that you don't have to move the data again. Synapse can read directly from the lake if you set it up that way.

Finally, visualization tools like PowerBI to connect to Synapse. That's the last mile. It's what the decision-makers see. But none of it would be possible without the pieces that came before—each one complementing the others.

And maybe the best part is that you don't have to write everything from scratch. The integration between these services is pretty smooth, at least most of the time. Things still go wrong. Pipelines fail, APIs break, or someone forgets to update schema mappings. But overall, this modular approach gives you control. You're not locked into one way of doing things.

Each tool here plays its role: Data Factory moves the data Figure 2, Databricks transforms it Figure 4, Synapse makes it accessible Figure 5. And taken together, they form a platform that can scale, adapt, and (ideally) make your data actually useful. PowerBI dashboard with business ready data. Figure 6.

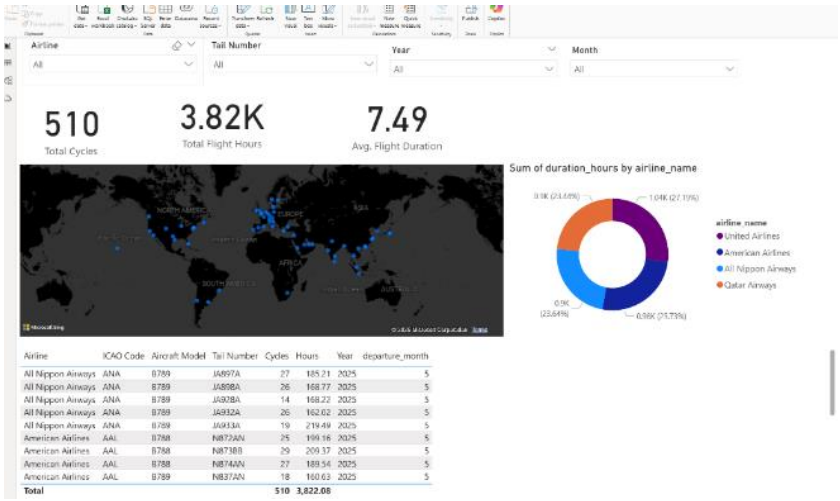

Figure 6
Power BI Dashboard

METHODOLOGY

When working with aerospace data, especially for reporting purposes, there's often this gap—details like the tail number, engine model, or even simple things like flight cycles tend to go missing or just not provided but needed to make reliability calculations. It's not that they aren't out there. They are. But they're scattered, or perhaps not prioritized by the main data sources. That's where something like the Flight Radar API becomes really useful.

So, here's what the project does, roughly speaking. It starts with ingestion. In this case, the source isn't a big internal database or a real-time sensor feed—it's an external API from *Rapid API*. I use that to pull enriched flight information: things like engine type, aircraft operator, ICAO airline code, flight duration, and the number of cycles. Especially important since they directly affect mechanical stress and cabin pressurization wear. A pressurization cycle is defined as the sequence in which the aircraft begins on the ground in an unpressurized state, becomes fully pressurized upon reaching cruise altitude, and then fully depressurizes upon descent and landing.

From there, I treat the data kind of like a pipeline. The first stage is just gathering and staging it—think of this as our version of Azure Data Factory.

Then begin the process. This is where the logic comes in. Using Databricks, do some transformations. For instance, the number of cycles is calculated in based the status of the flight if "Landed" count as a Cycle. Or cross-reference tail numbers to enrich the record with engine specs that weren't part of the initial API call. This step is where the raw feed starts becoming something structured, something that actually means something.

After transformation, it needs a place to live—a layer where others (or systems) can query it. That's where Synapse Analytics would come in. In other cases, maybe it's a centralized database or warehouse optimized for flight reporting, audit logs, or performance summaries. It doesn't have to be perfect. Sometimes even a CSV output works if it lands where it's needed.

Finally, visualization. Whether it's Power BI or Tableau, this step is all about surfacing that information. And not just for the sake of charts, it's about making the invisible visible. A missing engine model might be overlooked until you need to correlate it with maintenance delays. Tail numbers, when tracked properly, might reveal patterns in operator reliability. One insight leads to another.

So, these services, data ingestion, transformation, serving, and visualization—they're distinct, but deeply complementary. Kind of like different phases of the same conversation: you

gather, think, store, then speak. And even if it feels like overkill at first, having that structure makes a difference. Especially when the small stuff—the tail numbers, the cycles—starts to matter.

RESULTS AND DISCUSSION

After integrating the Flight Radar API with a structured ingestion and transformation pipeline, the result is a solution that can improve how MRO data is processed—and, perhaps more importantly, how it is used. The most immediate result was speed. Maintenance records that previously took weeks to complete, due to fragmented or missing flight data, were now processed in near real time. Or close enough that it made a noticeable difference in scheduling and dispatch.

One of the more persistent pain points had been reconciling incomplete records. Tail numbers with no engine model. Flight durations without departure or arrival context. The automated solution, while not flawless, filled those blanks with a degree of consistency that had simply not been practical before.

Now, I didn't expect perfection. And frankly, I didn't get it. Some records still came through patchy, especially when the upstream API failed to return all values. But the architecture allowed for fallback logic and periodic refreshes, essentially giving the system a second chance to fill in what it missed. That was enough, in most cases.

From a process perspective, what really changed wasn't just the completeness of the data, it was the confidence in it. MRO teams no longer had to pause or wait for weeks to confirm all the necessary data is there or double-check cycle counts before issuing a work order or shop visit. The data was prepped, consolidated, and delivered into their workflow in a timely manner. There were less back-and-forth. Fewer "Hey, can you check this?" emails. In a way, it just... flowed.

Interestingly, the greatest improvement didn't show up in the dashboards at first. It showed up in the speed of deployment. Reliability calculations were cleared and returned faster, sometimes by hours, occasionally by a full day. That might not sound dramatic, but over a month, across a fleet, that's a tangible operational gain. One that mattered more than initially expected result.

There's still room for nuance here. Automation sometimes creates new kinds of gaps, like over-relying on API freshness, or assuming field mappings won't change. I ran into both. And yet, the overall effect was a net positive. Manual data entry gave way to intelligent automation. Repetitive reconciliation was replaced by system-driven enrichment.

The system didn't solve everything. But it did change the rhythm of how things got done. And for MRO, where timing is everything, that alone felt like progress.

CONCLUSION

This project wasn't about building the most advanced model or deploying cutting-edge AI. It was about solving a problem that shows up in the day-to-day, missing data, fractured records, and the ripple effects that follow when you can't fully trust your inputs. By building a pipeline that ingests, enriches, and surfaces MRO-relevant data in a way that's structured and timely, I not chasing perfection—just usability. And that made all the difference.

The biggest takeaway? Clean data isn't just a technical goal. It's an operational enabler. When tail numbers line up, when engine types are populated automatically, when cycles don't need to be manually counted—it frees people up to focus on what actually matters: making the right maintenance decisions at the right time.

Here's still work to do. APIs evolve, data schemas drift, and no pipeline is ever "done." But the structure is there now. It's flexible enough to adapt, and lightweight enough to not get in its own way. And maybe more importantly, it proves a point: that small, practical applications of data mining can have a real, measurable impact—even in an industry as complex and regulated as the aerospace industry.

In the end, this wasn't just a data project. It was a maintenance project, a reliability project, a

workflow project. And if it helped close even a small part of the gap between data science and day-to-day MRO operations, then it did what it was supposed to do.

REFERENCES

[1.] Anant Sahay (2012). Leveraging Information Technology for Optimal Aircraft Maintenance, Repair and Overhaul (MRO) retrieved March 15. From Book ISBN: 9781845699826.

[2.] Asteris Apostolidis (2020) Aviation Data Analytics in MRO Operations: Prospects and pitfalls. Retrieved April 5, 2025. From: IEEE Xplore. [Online] Available: https://ieeexplore.ieee.org/document/9153694

[3.] Maurice Pelt et al. (2019) Data analytics case studies in the maintenance repair and overhaul (MRO) industry. Retrieved March 26,2025. From Matec Conference. [Online] Available: https://www.matec-conferences.org/articles/matecconf/abs/2019/53/matecconf_easn2019_04005/matecconf_easn2019_04005.html