Lessons Learned from Aircraft Component Failure Prediction using Full Flight Sensor Data

Changzhou Wang, Darren Puigh, Audrey Lei, Wei Guo, Jun Yuan and Mark Mazarek

The Boeing Company, Seattle, Washington, 98124, USA

changzhou.wang@boeing.com
darren.puigh@boeing.com
audrey.z.lei@boeing.com
wei.guo7@boeing.com
jun.yuan@boeing.com
mark.a.mazarek@boeing.com

ABSTRACT

Successful aircraft predictive maintenance relies on the accurate prediction of major aircraft component failures for operators to schedule and carry out maintenance operations before failure actually happens. In this paper, we share important lessons learned from our development of prognostics alerts using full flight sensor data, including various challenges of using big data, data quality issues, failure identification for data labeling, engineering-driven vs. data-driven methods, and aggregating alerts into actionable alerts. We also provide recommendations based on our experience with prognostic alerts developed and deployed for many airline operators.

1. INTRODUCTION

Aircraft predictive maintenance improves safety compliance, increases component life, reduces repair cost, and avoids schedule interruption. Modern aircraft are often equipped with thousands of sensors, and full flight sensor data with thousands of parameters are recorded for each flight at an average rate of 1Hz. Recent improvements on the accessibility to this sensor data and the affordability of big-data machine learning platforms have enabled data scientists to build prognostic models for predicting component failures at the right time with reasonable accuracy so that operators can carry out maintenance actions without excessive burdens.

However, aircraft are designed to be reliable and major components shall not and do not fail frequently. Therefore, it is often difficult, and sometimes infeasible, to directly employ state-of-the-art big-data machine learning approaches, as these methods require a large number of training cases. To address this challenge, data scientists often explore and incorporate deep physical and engineering knowledge when building prognostic models.

In this paper, we describe our experience in developing prognostic models from large volume of flight sensor data, discuss challenges and provide recommendations for best practices. Figure 1 shows a general flow for our prognostic model development process. Descriptive and diagnostic analytics, though not a focus in this paper, also benefit from the data acquisition and processing steps. In addition, we discuss data quality issues whenever appropriate [Lukens, Rousis, Thomas, Baer, Lujan, Smith, (2022)].

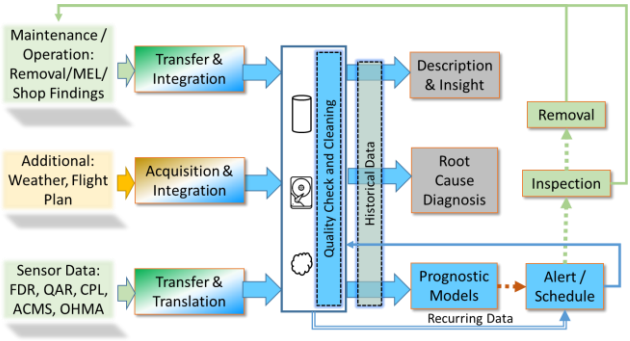

Figure 1 A prognostic model development process

2. FLIGHT SENSOR DATA

Aircraft are equipped with many sensors for monitoring operating conditions, such as temperature, pressure, valve position, power, rotation speed, and airflow rate. These sensed conditions, together with some control commands and calculated aircraft states, called "**parameters**" below, are recorded as flight sensor data during the whole flight or maintenance, and downloaded from the aircraft in real time or post-flight. Raw sensor data in compact binary format need to be translated into engineering unit data (e.g., numbers in units like foot or ampere) before analysis. Different devices

and systems are used in the data sensing, transmission, recording, and translation processes. Each step in this pipeline may introduce different types of data quality issues. We recommend creating and updating a checklist for all known issues, developing reusable code to check each known issue upon obtaining new data, and combining multiple independent sources to improve confidence on the quality check results. In addition, some quality issues only occur in a few flights, so it is important to check every flight instead of a random sample of flights.

The following is a list of common quality issues we have encountered:

- Data coverage. All flights of some airplanes might be missing (from a dataset that intends to include these airplanes); some flights might be missing for an airplane; some flights might be duplicated; and some parameters might be missing in some flights. For recurring data feeds, there are often delays for data arrival, and flights might come out of temporal order.
- Flight metadata. Departure and arrival airports are often important, but may be missing or incorrect due to manual input. Units for the same parameter can vary from across flights due to configuration changes. Parameter names often change in a similar way. Dual channel parameters may be merged before recording. Parameters may be added or removed over time for different flights in the dataset.

DeltaTime	Year	Month	Day	Hour	Minute	Second
0.1				23	59	19
0.2	2020	3	5			
...						
40.1				23	59	59
41.1				0	0	0
42.1				0	0	1
...						
60.1				0	0	19
60.2	2020	3	6			

Figure 2 Different time fields are not synchronized.

- Timestamp. Different time fields in the raw recording may have different sampling rate and starting time offsets. For example, in Figure 2, the year, month and day fields are recorded once every minute while the other three fields are recorded at 1Hz, and the recording time (DeltaTime) is off by at least 0.1 second. Simple carryforward or nearest neighbor time filling and interpolation will cause time jump back by almost 24 hours (from 2020/3/5 23:59:59 to 2020/3/5 00:00:00). In addition, glitches in recording can cause time jumping, freezing (i.e., not changing), or rewinding. Translation errors may shift multiple time segments to overlap each other. Multiple sets of timestamps (e.g., relative time and absolute time) might be available but inconsistent to each other. Timestamps from different sources may use different time zones and text formats (if data is not provided in binary format).
- Value. Parameter values can be out of range. Special values (e.g., 999) may be used to indicate invalid values. Parameter values might change too quickly or slowly (e.g., altitude change during Climb). Parameter values might be stale despite continuous operation condition changes. Glitches in recording might cause value dropout at the same time for a set of related parameters. Health flags might exist for every value in the raw data but become lost during translation. Parameter values might be valid only when a related mode/control parameter has a special value.

3. FAILURE IDENTIFICATION

To build an accurate failure prediction model and validate the model from historical data, we need complete and accurate failure information. Unfortunately, 100% coverage and accuracy are often very hard to achieve. In practice, we need to collect information from many diverse data sources with different levels of data qualities [Hodkiewicz, Ho (2016)], often incomplete, optional or in free text, and use multiple corroborative evidences to confirm failures and determine failure position and time.

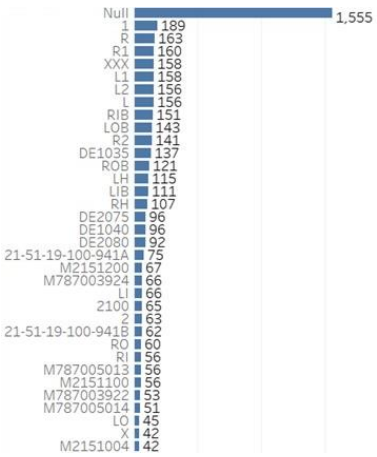

Figure 3 Occurrences of 35 (out of 166 total) different position text. Only L1, L2, R1 and R2 are expected.

- Determine component life on the aircraft: aircraft (tail), installation and removal time, part number, serial number, and position (when there are multiple installations of a component in different positions on the same aircraft). Part and serial numbers are often mandatory fields with occasional typos. Position is often optional and free-text with great variations (see Figure 3 as an example). Flight deck effect and maintenance messages (generated by on-board condition monitoring systems), complaints, MEL[1]

[1] MEL, Minimum Equipment List, is intended to permit aircraft operation with some inoperative items for a short period before repair can be done. Once an item in the MEL list is failed, operator is required to "put it on MEL". Hence, the MEL record indicates the failure time, and associate comment provides additional information such as position of the component.

comments, removal reason, maintenance action and shop findings might often include indication of position; but it is non-trivial to link related records for the same component, extract position info and resolve conflicts.

- Determine failure time. If the first MEL record can be found for the removal, it likely provides the failure time. Maintenance messages and complaints might provide alternative or additional evidences. When the failing flight is available, it can be more reliable to detect failure from the sensor data than relying on human input. More importantly, not all removals are due to failures. Scheduled removals, upgrades, cleaning, part swapping or robbery are examples. Removed parts might be shown as no fault found in the shop, and may be excluded from failing parts.

- Determine failure mode, likely from shop findings and teardown reports. However, it might be very difficult to determine the sequence of failures and hence the main failure mode, despite what is written in the report. Failures or maintenance messages of related components might provide hint.

- Find behavior changing events. Changes in on-board control software may change the characteristics of relevant sensor data. Failure and replacement of related components (e.g., a temperature sensor providing feedback loop for the control of the target component) may also change the behavior of the component. Extended grounding periods, such as those due to COVID19, can also modify the behavior (while the sensor data is often not available during ground sustainment).

4. MODEL DEVELOPMENT

Once the sensor data is cleaned and failure time is accurately determined, it would appear straightforward to apply a cutting-edge machine learning algorithm to build a prediction model [Darrah, Lovberg, Frank, Quinones-Gruiero, Biswas, (2022); Mitici, de Pater, Barros, Zeng, (2023)]. Unfortunately, current machine learning methods, such as deep recurrent neural network or transformer-based attention networks, cannot be directly applied in many practical cases. The history of a single component often spans over hundreds to thousands of flights, each containing tens of thousands of time steps. Successful prediction of failures often depends on signals in past flights, or trends over many historic flights or baseline conditions when the component is first installed.

On the other hand, today's sequence machine learning methods cannot handle very long sequences. Indeed even the best Transformer based algorithm cannot handle more than a few thousands time steps [West (2023)]. As a result, a practical approach often takes two steps: (1) extract summarized features from each flight, and (2) predict failure from the history of flights.

Engineering knowledge often provides a good starting point for identifying interesting events within each flight and extracting features according to these events. This will greatly reduce the number of parameters and the volume of data. Usually, most initial hypotheses hold up only on sample flights, but do not survive after being tested on all flights. As a result, it pays off quickly to have a high-level language (see Figure 4 as an example) or API to capture, test and refine engineering hypotheses on many flights. In addition, it is often difficult for engineers to provide exact thresholds or window sizes, especially when there are a large number of combinations, while a machine learning method may be used to select the best combination that best differentiates normal vs. near failure flights.

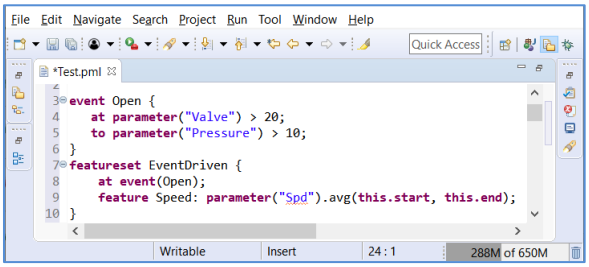

Figure 4. A high-level language for engineering-driven feature extraction

Due to personal experience, engineering knowledge is usually biased and limited to a small scope, and may prevent discovery of reliable prediction logic. A data driven approach instead starts with all available data. For example, we can build a normal behavior model of a component from historical data, and use deviations or anomalies in each flight to predict failures [Yuan (2022)]. Here, statistical analysis and machine learning methods can be used to select relevant parameters, interesting events, and appropriate aggregation features in building the normal behavior model.

Without engineering guidance, the data-driven methods might become infeasible due to the required computing resources. Indeed, engineering input is key to select the target parameters to model the normal behavior.

Once summarized features are extracted from each flight, an appropriate machine learning method can be applied directly as there are usually only a few hundreds of sequences, each with up to a few thousands of flights (time steps). Since the number of failure cases are often small, deep learning methods might not work well. Instead, we can again leverage engineering knowledge to augment the data with temporal aggregation across flights, and then apply tree-based machine learning methods (e.g., Random Forest or XGBoost).

Finally, due to the small number of failure cases, it is important to simplify final models to reduce over-fitting, and improve interpretability. When the prognostic models are aligned with engineering knowledge, it is more robust to

handle unseen failure paths since the underlying physics are not changed.

5. Alert Generation and Feedback Loop

Prognostic models usually predict for each flight about how likely the component will fail soon, e.g., whether it will fail within 30 days from each given flight. However, the per-flight prediction result is often not stable, and can flip between positive (will fail) and negative (will not fail) in consecutive flights.

Based on the per-flight predictions, airline customers may be alerted to take inspection and replace the component when inspection results are positive (e.g., damage found). When the number of positive predictions is large, it is impractical for airlines to take actions on each positive flight. Instead, per-flight predictions shall be aggregated into actionable alerts.

Ideally, for each and every failure, a single alert is issued, and the inspections confirms the damage before the actual failure. In practice, some alerts are false alarms, inspection results might be negative, and failure may happen despite of negative inspection result.

From the customer point of view, the performance of a prognostic model is not the per-flight metrics, but rather on the actionable alerts and inspection actions: how many failures have been captured? How many failures are missed without alert? How many failures are missed with negative inspection results?

Ideally, the end-to-end prognostic model shall use these metrics as the optimization objective. This might be done using a hyper-parameter tuning framework, e.g., Liaw, Liang, Nishihara, Moritz, Gonzalez, Stoica (2018). During model development, there is no inspection due to generated alerts on historical data. This might be simulated with a probability model on inspection actions and inspection results. As a result, the hyper-parameter tuning framework shall be flexible enough to take the simulation function as the tuning objective.

6. Conclusion

In this paper, we discussed key elements and lessons learned in our research and deployment experience on component failure prediction. Challenges like very large data volume, very long time sequence, and limited number of failures make it difficult to take a holistic approach to build a single prognostic model using the actionable alert performance as the optimization objective. Encouraged by the recent success of ChatGPT, we are looking forward for technology advancements that allow us to build Generative AI models for flight sensor data and fine-tune them for different components' failure prediction.

Acknowledgement

We would like to thank our data team, engineering team, and airline collaborators for their support in collecting data, providing engineering input and operation feedback.

References

Darrah, T., Lovberg, A., Frank, J., Quinones-Gruiero, M., Biswas, G. (2022). Developing Deep Learning Models for System Remaining Useful Life Predictions: Application to Aircraft Engines. *Annual Conference of the PHM Society*, 2022.

Hodkiewicz, M., Ho, M. (2016). Cleaning historical maintenance work order data for reliability analysis. *Journal of Quality in Maintenance Engineering*. Volume 22, Issue 2, 2016.

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J., Stoica, I. (2018). Tune: A Research Platform for Distributed Model Selection and Training. July 13, 2018. https://arxiv.org/abs/1807.05118.

Lukens, S., Rousis, D., Thomas, D., Baer, T., Lujan, M., Smith, M. (2022). A Data Quality Scorecard for Assessing the Suitability of Asset Condition Data for Prognostics Modeling. *Annual Conference of the PHM Society*, 2022.

Mitici, M., de Pater, I., Barros, A., Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Journal of Reliability Engineering and System Safety 234 (2023)*.

West, C. (2023). AI and the FCI: Can ChatGPT project an understanding of introductory physics? March 3, 2023. https://arxiv.org/pdf/2303.01067.pdf.

Yuan, J. (2022). Boeing Operationalized Aircraft Predictive Maintenance. *AAAI Fall Symposia Series on Artificial Intelligence for Predictive Maintenance*. Nov 19, 2022.

Changzhou Wang received his Ph.D. in Information Technology from George Mason University, Fairfax, Virginia, USA in 2000. He joined Boeing in Sept 2000 and is now an Associate Technical Fellow and data scientist in Boeing Global Services. His research interests include temporal data analytics tool development, adapting advanced machine learning methods and combining large-scale flight sensor data and deep engineering knowledge for prognostic modeling. He is a member of ACM and AGIFORS.

Darren Puigh received his Ph.D. in Particle Physics in 2011 from Cornell University, Ithaca, NY. After that, he was a postdoctoral researcher at the Ohio State University in collaboration with the research center CERN, near Geneva, Switzerland, and the Large Hadron Collider experiment. He joined Boeing as a data scientist in 2016, and he is now a senior data scientist in Boeing Global Services. His research interests include solving problems with data and data analysis. He enjoys working with unique and varied data sets, typically multi-variate time series data, to understand and leverage solutions for complex systems.

Audrey Lei is a data scientist at The Boeing Company, where she develops data-driven prognostics to support airline predictive maintenance. She earned a Master of Information & Data Science from UC Berkeley in 2022 and an MS in Electrical Engineering from the University of Southern California in 2017. Before joining Boeing in 2018, she was a technology consultant for a Big 4 firm. Audrey is a member of the Society of Women Engineers, and her technical interests include computer vision, NLP, data privacy and ethics.

Wei Guo received her Ph.D. in Industrial and Systems Engineering from University of Washington, Seattle, WA in 2020. She joined Boeing Global Services as a data scientist in January 2022. Her research include developing explainable prognostic models with multivariate time series data and revamping manufacturing processes with state-of-the-art machine learning solutions for increased efficiency.

Jun Yuan received his Ph.D. in Computer Science from the Southeast University, China in 1995. He is currently a Technical Fellow with the Boeing Global Services, working in the area of predictive maintenance. Throughout his 20+ year career with Boeing, Jun has been leading various R&D projects in the areas of data management, data integration, and data analytics. Jun has published 30+ technical papers and has been awarded 12 US Patents. Prior to joining The Boeing Company, Jun worked as a faculty member in the School of Computer Science of the Florida International University.

Mark Mazarek grew up in Champion, Ohio, USA, and earned B.S. in Aerospace Engineering from Embry Riddle Aeronautical University (Daytona Beach, Florida, USA) and M.S. in Systems Engineering from Missouri Science and Technology (Rolla, Missouri, USA). A senior manager in Boeing Global Services (BGS) Engineering focused on Commercial Airplane Fleet Performance & Lifecycle data, he leads a team of managers, individual contributors and integrated project teams in the development of digital, data and analytics engineering capabilities that improve customer operations, optimize lifecycle cost of airplanes and platforms, drive operational excellence, and further strengthen product and services quality and safety. He also leads the development team focused on operational digital twin and operational digital thread initiatives, as BGS focuses on scaling digital engineering capabilities. Most recently, he led the development and execution of the commercial predictive maintenance capability within BGS Engineering collaborating with Airplane Health Management, Boeing Commercial Airplanes Customer Support, BCA Design Engineering and airline customers. Prior to this role, he led the 777X Reliability and Maintainability team, overseeing certification deliverables and reliability artifacts.