ATD-2 Machine Learning Model Validation Framework Design

Case Study

MOSAIC ATM™

ATD-2 Machine Learning Model Validation Framework Design

Why Model Validation is important?

With the rise of machine learning integration into everyday systems and processes, monitoring the reliability and accuracy of the projections generated becomes greater. More organizations have deployed machine learning, but few are implementing practices to monitor the model's performance. Machine learning models produce less than reliable results without watching for bias, drift, and retraining on new data, leaving organizations at risk. These firms would have been wise to implement validation into their ML development but instead are linked to examples of what not to do in this piece.[1]

Model validation and the practice of MLOps[2] aim to test whether the trained model is trustworthy, unbiased, uncovers errors in scoring, identifies scalability challenges, and enhances the overall model quality. Mosaic, a leading aviation systems development company, is a proponent of explainable, actionable, and reliable machine learning solutions.

In the following case study, Mosaic examines how they built validation into an ML scoring process for Aviation Trajectory predictions. What good are ML-driven projections if they aren't trustworthy?

ATD-2 Background

Mosaic ATM has been supporting the Airspace Technology Demonstration 2 (ATD-2)[3] for several years. NASA's mission is to provide solutions to several problems in the complex, multi-airport environment. At most airports today, departures are managed to push back from the gate, which can overload runways and cause excessive taxi and hold times. Increasing operational efficiency has several downstream benefits, including significant reduction of CO2 emissions.

Additionally, significant uncertainty in the duration of the taxi-out, takeoff, and climb phases of flight leads to inaccurate demand predictions, decreased situational awareness, and overly conservative airspace restrictions that traffic managers are compelled to apply to compensate for this uncertainty.

Mosaic ATM is well suited to support NASA's effort as machine learning, artificial intelligence, and predictive analytics have been critical to solving these challenges. More information on the ATD-2 effort can be found here.[4]

Machine Learning Validation Background

The last few months of ATD-2 were spent preparing to engage with the Digital Information Platform (DIP). In this project, Mosaic was tasked with supporting a demo in the D10 TRACON that utilizes the TOS program developed in ATD-2. The difference here is that the system driving the backend is based on machine learning models instead of legacy STBO. There will be some components from STBO integrated in, but the core system will be running in the cloud while supporting TOS operations.

In parallel to the development and integration of the machine learning models into the system, a validation framework was developed in python for verification and validation purposes. This framework evolved into system health alerting using Airflow.

This task validates the following:

- Model results are correctly stored in the fuser data and correspond to the correct gufi, timestamp (nothing is lost in the flow from java, flask, etc.)
- Models are producing all the expected results for each flight

Machine Learning Models

Mosaic has designed and deployed several machine learning algorithms to predict different phenomena in and around busy airports. A few examples are.

- Taxi Time In
- Taxi Time Out
- Arrival Runway
- Airport Configuration
- Estimated On-Time Arrival
- Departure Runway

System Setup

There are multiple systems set up and deployed on NTX equipment. Each one is configured for different airports but leverages the same validation framework. For example, one system runs for the D10 TRACON and includes DFW, DAL, and other small airports within that terminal airspace. Another system is configured to run the Northeast airports, including EWR, LGA, and JFK.

Deployed Models

Models are trained and stored on NTX equipment using ML Flow. The selected version of each model, along with environment specs and training results are stored in a database and can be accessed using a ML Flow Web Interface.

Once models are ready for deployment, they register within ML Flow. The validation framework uses a python module called paramiko to ssh into the live machines and return the model specifications. The validation then pulls the required information from the ml flow server to validate the modeling results.

Validation Tests

The following validation tests are performed on each deployed model:

- Completeness Test
- Feature Change Test
- Identical Features Test
- Matching Predictions Test
- Model Response Test
- Completeness Test

A valid prediction populates in an output table for all flights that have non-null core features and are present in the request data. If a flight has a null core feature, an error code will be associated with the model response and logged in a quality table. This test also ensures that any flights will null predictions have an error code associated.

Sample output for failing tests: These failures occurred before predictions were being written to the data table, and the predictions were being subject to mediation rules with TFM, TMA data.

2021-01-14 11:01:34 INFO	_completeness_test() There are 4928/75346 flights missing predictions
2021-01-14 11:01:34 INFO	_completeness_test() gufi=AAL2779.MIA.DFW.210112.1810.0051.TFM, timestamp=2021-01-13 21:07:11, prediction=nan
2021-01-14 11:01:34 INFO	_completeness_test() gufi=AAL1973.SJD.DFW.210112.1935.0030.TFM, timestamp=2021-01-13 21:38:11, prediction=nan
2021-01-14 11:01:34 INFO	_completeness_test() gufi=AAL1973.SJD.DFW.210112.1935.0030.TFM, timestamp=2021-01-13 21:39:41, prediction=nan
2021-01-14 11:01:34 INFO	_completeness_test() gufi=AAL1973.SJD.DFW.210112.1935.0030.TFM, timestamp=2021-01-13 21:40:11, prediction=nan

Feature Change Test

This test confirms that if a feature change results in a different predicted value, it is recorded in the output data and associated with the correct timestamp of the feature change. Not all feature changes result in different predictions.

Identical Features Test

This test ensures that flights with identical features have the same predicted results populated in the database. For example, if multiple flights have the same carrier, predicted arrival runway, and predicted arrival stand, the test ensures that the unimpeded ama and unimpeded ramp taxi times are identical.

Matching Predictions Test

This test extends the completeness test and ensures that non-null predictions in the data output table match the values parsed from the ingested data in the data quality table. The test branches into three separate categories:

- Tolerance (eon service)
- Exact numeric match (taxi in, taxi out)
- String match (arrival runway, departure runway)

Sample output for failing tests (eon-service):

2021-01-14 11:03:15 INFO	_matching_predictions_test() Total flights for test: 66881
2021-01-14 11:03:15 INFO	_matching_predictions_test() Count predictions Pass 1 minute tolerance: 18417 [27.5%]
2021-01-14 11:03:15 INFO	_matching_predictions_test() Count predictions Fail 1 minute tolerance: 48464 [72.5%]

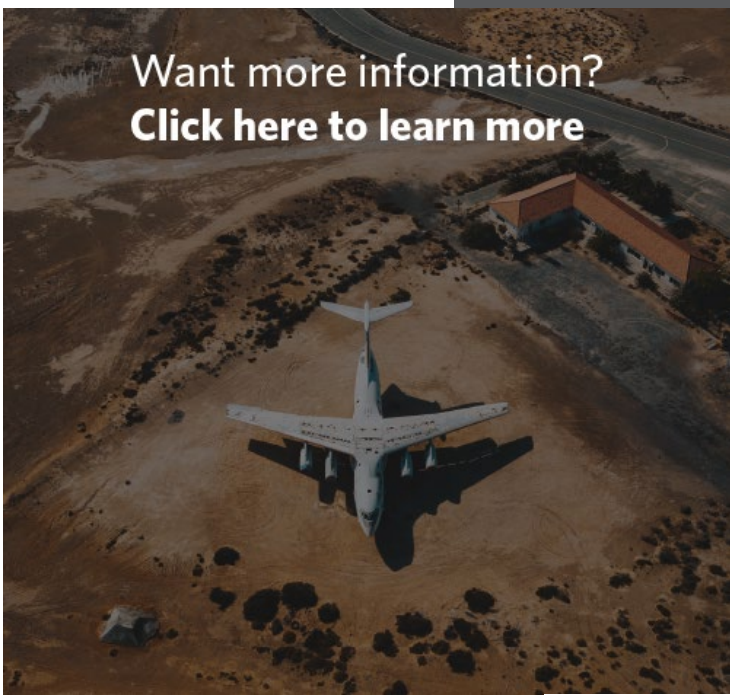
Model Response Test

This test injects the data from the quality table directly through the machine learning models to ensure the predictions and error responses line up. It is an extra check to ensure the flask service doesn't drop anything when writing to the output table. It also helps ensure the live system is running with the correct production environment, producing slightly different results if the scikit-learn module is not identical to what the model was trained on.

Conclusion

This case study demonstrates how intensive MLOPs and model validation can be. Hopefully, the necessity of these processes comes through, especially in an operationally complex environment, the busy airport. To realize the full power of machine learning, organizations need to focus on operations and delivery of the predictions as much as the development of the algorithm itself. In the case of NASA and ATD-2, not being able to trust the projections generated could be a life-or-death situation. Air traffic controllers must trust the recommendations presented by them, and validation is essential towards building trust.

Mosaic's holistic approach differentiates us from others in the market because we focus on developing custom applications that fit your ecosystem rather than trying to shoehorn prebuilt solutions that ultimately don't integrate into your workflow.



Endnotes

1. https://towardsdatascience.com/real-life-examples-of-discriminating-artificial-intelligence-cae395a90070

2. https://ml-ops.org/

3. https://aviationsystems.arc.nasa.gov/research/atd2/index.shtml

4. https://mosaicatm.com/2021/04/09/atd-2/