Machine Learning Airport Surface Model

William J. Coupe
NASA Ames Research Center
Moffett Field, CA, USA
william.j.coupe@nasa.gov

Alexandre Amblard, Sarah Youlton
Universities Space Research Association
Moffett Field, CA, USA
alexandre.amblard@nasa.gov,
sarah.a.youlton@nasa.gov

Mathew Kistler
Mosaic ATM
Leesburg, VA, USA
mkistler@mosaicatm.com

***Abstract*—Future needs of the National Airspace System require decision support tools to adopt a service-oriented architecture in alignment with the FAA's vision for an Info-Centric NAS. To achieve this, many existing systems will need to undergo a digital transformation from a monolithic decision support tool to a service-oriented architecture where individual services are exposed through well defined Application Programming Interfaces (APIs). To enable this transformation, NASA has developed the Digital Information Platform as a cloud based foundation for development of aviation services with a special focus towards Artificial Intelligence and Machine Learning (ML) services. This paper describes the work required for the transformation of NASA's legacy surface management system to a real-time ML based decision support system deployed in the cloud. Details of the Machine Learning Operations (MLOps) infrastructure and best practices are described which enabled the end-to-end lifecycle management of ML within an integrated software system. Validation results are provided from an operational field evaluation where performance was benchmarked against the legacy approach.**

***Index Terms*—digital transformation, MLOps, airport surface model, operational field evaluation**

I. Introduction

Concepts and technologies to manage arrivals, departures, and surface operations using Trajectory Based Operations (TBO) have been under development by NASA, the Federal Aviation Administration (FAA), and industry to improve the flow of traffic into and out of the nation's busiest airports. NASA technologies for specific phases of flight were integrated [1] across surface [2], [3] and airspace domains [4], [5] and deployed as the Integrated Arrival Departure and Surface (IADS) traffic management system [6], [7]. The IADS system was developed in alignment with FAA's Surface Collaborative Decision Making (S-CDM) Concept of Operations [8] and refined over time [9].

Future needs of the National Airspace System (NAS) require decision support tools such as the IADS system to adopt an architecture as described by the FAA's vision for an Info-Centric NAS [10]. To align with the vision, decision support tools need to migrate toward learning, adaptable, and lightweight interacting systems. To achieve this, many existing systems will need to undergo a digital transformation from a monolithic decision support tool to a service-oriented architecture where individual services are exposed through well defined APIs. This enables industry services to be deployed alongside FAA infrastructure and services to supplement existing investments in TBO capabilities.

NASA has developed the Digital Information Platform (DIP) [11] to enable the transformation towards an Info-Centric NAS. The overall goal of DIP is to support transformation of the NAS through the development of a cloud-based platform for advanced, data-driven, digital services for both traditional and emergent operations. DIP provides access to real-time and historical data upon which data-driven services for NAS users are developed and operated. A key focus area of DIP is to accelerate the development of Artificial Intelligence (AI) and Machine Learning (ML) techniques applied to aviation and to provide an opportunity for AI/ML experts from outside traditional aviation to make an immediate impact through the development of novel services.

ML techniques have been applied to aviation problems for many years [12] to develop prediction models. However, the real challenge isn't building a ML model, the challenge is building an integrated ML system and to continuously operate it in production [13]. Without the proper approach, it is easy to incur massive ongoing maintenance costs at the system level when applying machine learning [14]. To address this challenge, in recent years there has been focused work on Machine Learning Operations (MLOps) to develop infrastructures and platforms for end-to-end lifecycle management of ML [13], [15], [16].

This paper describes the work required for the transformation of NASA's legacy IADS system to a real-time ML based decision support system deployed on DIP in alignment with FAA's Info-Centric NAS. The deployed ML system was validated during an operational field evaluation and performance was benchmarked against the legacy IADS system. Section II provides background information on IADS capabilities and the legacy approach. Section III describes the architecture of the new integrated ML system and how it was deployed as a service-oriented architecture. Section IV describes the MLOps best practices that were adopted and the software tools used in the development of the MLOps infrastructure. Section V shows a side by side comparison of the ML system vs. legacy system and Section VI provides concluding remarks.

II. Background

At the core of the IADS system was an airport surface model that generated predictions including but not limited to airport

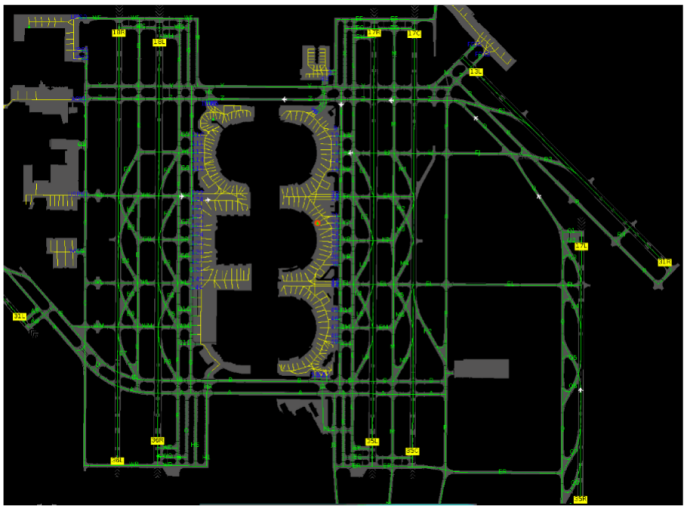

Fig. 1. KDFW airport surface adaptation.

configuration, runway assignment, unimpeded taxi times, and arrival ON times [9]. The airport surface model predictions were used as input by the IADS Terminal Scheduler which applied all known constraints across each airport surface and the terminal boundary [17] to generate predictions for the Estimated Take Off Time (ETOT) for each departure flight. To generate the airport surface model predictions the IADS system relied upon detailed adaptation developed for each individual airport and the terminal airspace.

Adaptation for each airport requires creating a link-node network for each airport surface which involves significant manual effort. Fig. 1 shows an example of adaptation created for KDFW airport which illustrates the detailed structure of the link-node network defining the gate locations, ramp and taxiway structure, and runway locations. The adaptation goes beyond defining the physical structure of the airport and also requires detailed knowledge from Air Traffic Control (ATC) encoded in decision trees including departure fix to runway mappings and other local knowledge that might be applied to the airport or airspace.

Consider a departure flight to illustrate how the legacy IADS system used the adaptation to generate predictions. The departure fix in the filed flight plan would be referenced against the decision trees defined in the adaptation to determine the assigned runway. Next, a path along the surface would be constructed from the parking gate to the runway as the shortest path within the link-node network. A pushback duration and taxi speed would be determined for the departure by referencing decision trees based on historical data and the flight would be propagated along the path at the given velocity to create a 4-D trajectory. Throughout the taxi duration from gate to runway, surface surveillance would detect the current location of the flight and generate an updated path to the runway and continuously propagate the flight along the updated path.

The adaptation based approach leveraging physics based 4-D trajectory predictions generated attractive results for KDFW and KDAL [7], [9]. The challenge with the adaptation approach is it took considerable time and effort to build and maintain for each individual airport and the terminal airspace. With the goal to scale the IADS capability across the NAS, the adaptation created a bottleneck to deployment.

III. Machine Learning Airport Surface Model

The Machine Learning Airport Surface Model was developed to address the deployment bottleneck introduced by legacy adaptation. The machine learning approach has the advantage that instead of requiring detailed adaptation and knowledge encoded in decision trees the machine learning can learn directly from the underlying data. The machine learning deployment can then be more easily scaled to multiple locations across the NAS.

A. Machine Learning Replacement of Legacy Adaptation

For departure aircraft, the Terminal Scheduler generates the ETOT on the filed route and the Trajectory Option Set (TOS) alternative routes [9], [18]. For arrival aircraft, the Terminal Scheduler takes the estimated ON time and uses that as a constraint to schedule departures within the remaining runway capacity. Consider Fig. 2 which illustrates the Terminal Scheduler and the different components required to schedule aircraft at both the runway and the terminal boundary. The Terminal Scheduler is composed of an Orchestrator, Trajectory Modeler, Departure Fix Scheduler, and Airport Scheduler for each airport.

The Terminal Scheduler runs every 30 seconds and has 3 sub-routines. The loop k1 is responsible for modeling the unimpeded trajectory to the runway for both departures and arrivals. The unimpeded trajectory is then used as input to the loop k2 which is the main scheduling loop. The main scheduling loop applies all known constraints along the terminal boundary and each airport surface and iterates until a schedule is found that satisfies all constraints. The loop k3 applies the main scheduling loop under a what-if scenario where a particular flight is assumed to use a TOS alternative route as opposed to the original filed route. By comparing the ETOT on the TOS route to the ETOT on the original filed route, the system recommends reroutes when the delay savings exceeds the flight operator provided Relative Trajectory Cost (RTC) [9], [18].

As shown in Fig. 2, the ML Airport Surface Model is used as the Trajectory Modeler component within the Terminal Scheduler in the ML approach. The ML Airport Surface Model replaces the legacy adaptation physics based 4-D trajectory modeler responsible for modeling unimpeded trajectories to the runway. The scalable ML solution generates the same critical data elements needed by the Terminal Scheduler without requiring adaptation. The critical data elements required include the airport configuration [19], departure runway [20], departure unimpeded take off time [21], arrival runway, and arrival estimated ON time [22]. The unimpeded take off time is a combination of the flight operator provided Earliest Off

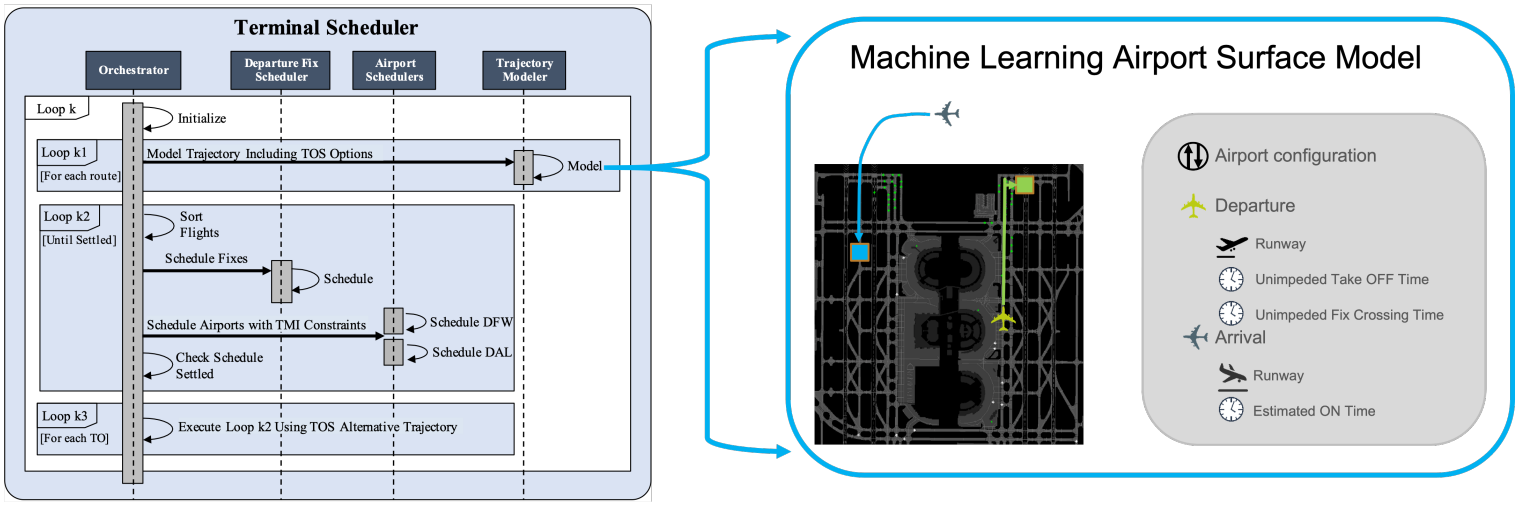

Fig. 2. ML airport surface model used within the scheduling algorithm.

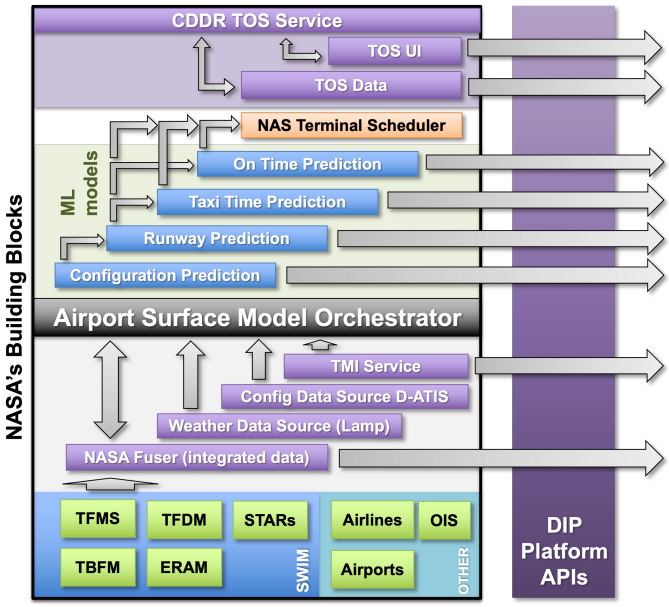

Fig. 3. Service-oriented architecture.

Block Time (EOBT) plus a prediction of unimpeded taxi time from gate to runway.

For departures, one of the distinctions between the ML Airport Surface Model and the legacy adaptation based Airport Surface Model is the use of surface surveillance. The legacy adaptation based Airport Surface Model used surface surveillance as input to detect the current location of every aircraft on the airport surface and update the trajectory to use the current location as the starting point. This provided a 4-D trajectory that was constantly updated at 30 second frequencies. In contrast, the ML Airport Surface model only updates the trajectory once when the flight crosses the taxiway spot. At spot crossing, the trajectory is updated to current time plus the prediction of unimpeded Airport Movement Area (AMA) taxi time.

B. Deployment as Service-Oriented Architecture

Fig. 3 shows a detailed view of the ML Airport Surface Model architecture. The ML Airport Surface Model is deployed in a service-oriented-architecture in which each logical service is distinct with well-defined inputs and outputs. The ML Airport Surface Model starts at the bottom of the figure from a foundation of raw data feeds including FAA System Wide Information Management (SWIM) data feeds and other available airline or airport data feeds. The raw feeds contain valuable data but can provide inconsistent information on the same flight that is difficult to reconcile in a real-time environment.

To address this challenge, NASA developed logic that could resolve data processing and mediation complexities. Much of this work is embodied in the Fuser service. The Fuser framework mediates between the disparate sources of data, pulling in the right data, at the right time. The Fuser leverages heuristics and analysis on which data source is best to use for a specific need and provides access to the information in a common well defined data model.

The Fuser data is used as input to the Airport Surface Model Orchestrator. The Orchestrator also pulls in weather data, current airport configuration data in the form of Digital Automatics Terminal Information Service (D-ATIS), and restriction data from NASA's Traffic Management Initiative (TMI) service. The TMI service combines restriction data from FAA SWIM data feeds in addition to local restrictions only available on the Operational Information System (OIS) page. The restriction data is parsed to identify individual restrictions correlated and assigned at the flight level by the TMI service prior to being passed as input to the Orchestrator.

The Orchestrator is responsible for collecting the inputs required by each ML prediction service and also responsible for calling the ML services in the proper order. Even though each service is distinct with well defined inputs and outputs

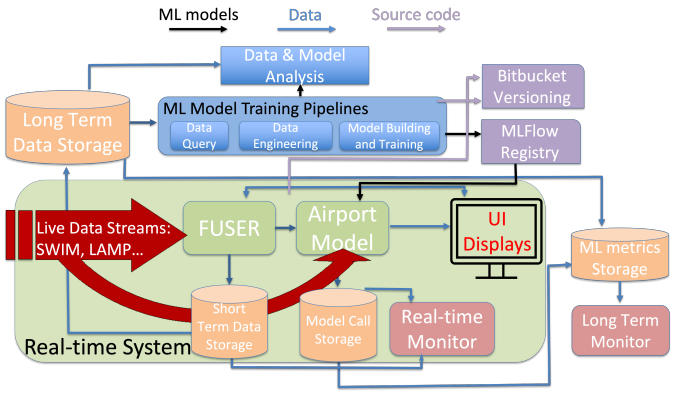

Fig. 4. MLOps infrastructure.

	Count	Accuracy
Offline, overall	-	88.8%
Online, overall	96	86.5%
Online, inputs correct	66 (68.8%)	87.9%
Online, any inputs incorrect	27 (28.1%)	81.5%
Online, default response	0 (0.0%)	-
Online, rwy replaced (out of config)	3 (3.1%)	100.0%

Fig. 5. Real-time KDFW departure runway metrics for debugging.

there are dependencies between the different ML prediction services that need to be accounted for. Fig. 3 shows the dependencies between the services as the output of the airport configuration service is used as input to the runway service. Similarly, the output of the runway service is used as input by the taxi time service and the arrival ON time service. Outputs of the runway, taxi-time, and arrival ON service are used as input by the Terminal Scheduler as described in Section III-A.

Outputs of the Terminal Scheduler are passed to the Collaborative Digital Departure Reroute (CDDR) TOS service which computes the delay savings for each TOS route as the difference in ETOT on the TOS route compared to the original filed route [23]. If the delay savings exceeds the flight operator provided RTC, the TOS route is then shown to the flight operator as a Candidate for reroute. Output of the CDDR TOS service also includes critical data elements required to drive the front end UI which the users interact with.

Outputs of each individual service shown in Fig. 3 are made available through well defined APIs deployed on NASA's Digital Information Platform (DIP). Each individual service can be consumed and used as building blocks for downstream service developers. By making the services available through API, others can benefit from the ML Airport Surface Model accelerating the development cycle for new capabilities that require these foundational data elements.

IV. MLOps

For deployment of the ML Airport Surface Model, we adopt MLOps best practices across the real-time system and the off-line training infrastructure. Fig. 4 illustrates the real-time system and the off-line training infrastructure and the relationship between the two. The adoption of MLOps best practices described in this Section helps reduce risk in deployment to ensure both the models and the pipelines feeding the models are consistent between the off-line training infrastructure and the real-time deployment.

A. MLOps Infrastructure

The off-line training infrastructure is shown in the top of Fig. 4 and begins with a historical archive of the data used as input to the Orchestrator shown in Fig. 3. These data are ingested by the ML model training pipelines that are composed of three phases: data query, data engineering, and model building and training. The trained models are then validated against out of sample test sets using k-fold cross validation and other validation analysis. The trained models are then stored in MLFlow and version control of the model training pipelines is maintained via Bitbucket.

The real-time system is shown in the bottom of Fig. 4 and begins with streaming data feeds that are fused and mediated in real-time by the Fuser prior to being used as input to the Orchestrator shown in Fig. 3. It is important to ensure that any data engineering techniques applied to the historical data are replicated in the real-time feeds or else model performance will degrade as the distribution of features in the training data will not match the features provided to the model in real-time. The real-time fused data feeds and the features delivered to the on-line model are stored in databases and used as reference in a real-time performance monitor. Nightly queries are performed to summarize model performance and stored in a ML metric storage database allowing for long-term monitoring of model performance.

The real-time performance monitor has been a valuable tool during deployment of the system to provide a tight feedback loop between the system and engineers. Often times it is easier to identify and debug issues while tracking the real-time performance as opposed to aggregate performance. To track real-time performance, we adopted a strategy to measure performance of models conditional on the input received. As shown in Fig. 3, the ML models have dependencies with upstream models and it is important to understand these dependencies if the performance of a particular model is degrading or if the inputs fed to the model are degrading.

Fig. 5 shows an example of the real-time metrics computed and tracked in the real-time performance monitor for the departure runway model at KDFW. We keep track of the off-line validation accuracy measured in training and compare it to the overall real-time performance accuracy. Each model

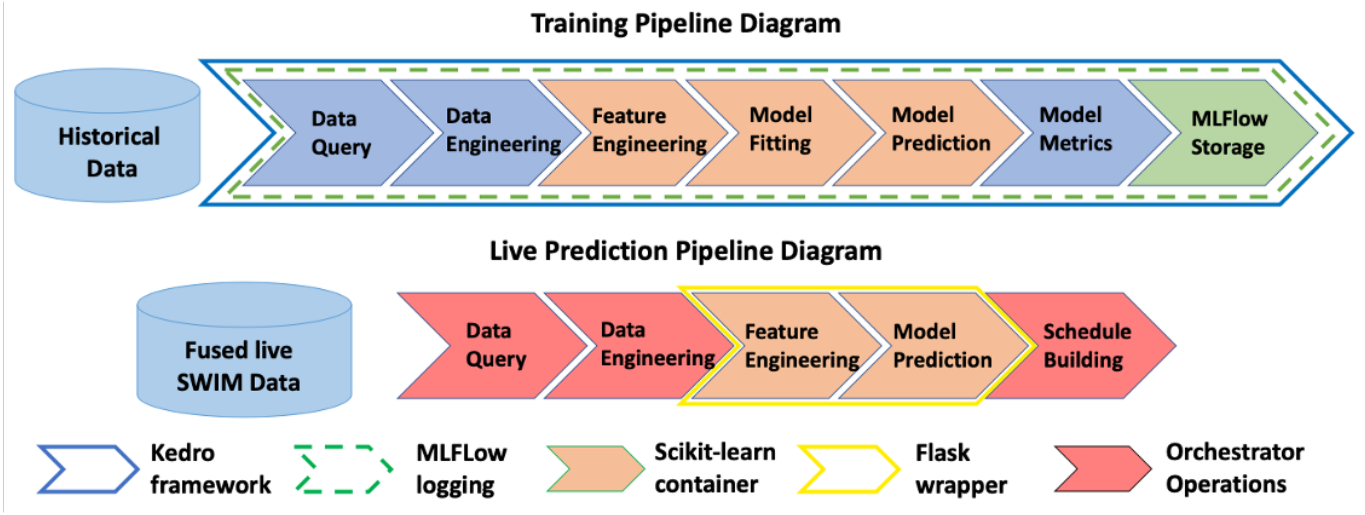

Fig. 6. ML pipelines and software tools.

is also scored within four distinct categories where a single prediction will fall into one of: 1) all input features correct, 2) any input feature was incorrect, 3) model generates a default response, and 4) the model prediction was replaced. The departure runway ML model generates a default response if any core feature is missing and the model prediction is replaced if it violates obvious constraints such as a South flow prediction when D-ATIS detects the airport is currently operating in North flow.

For each category, we keep track of the number of predictions that fall into the category and the associated accuracy. For example, for KDFW you can see that in total we observed 96 predictions of which 66 (68.8%) had all the correct input features with associated accuracy 87.9% and 27 (28.1%) had at least one input feature incorrect with associated accuracy 81.5%. For the departure runway prediction model, an example of a prediction with incorrect feature input is a departure flight that is predicted to use departure fix A but was observed to actually use departure fix B. In this scenario, we would expect for departure runway prediction model performance to degrade as shown in Fig. 5.

Whereas the real-time performance monitor evaluates data within hours of current time and provides a tight feedback loop for deployment and debugging, the long-term performance monitor evaluates data over months and is a valuable tool to measure the drift in model performance and to compare aggregated model performance against benchmark methods. Since the long-term monitor measures the performance over much larger data sets, it is not practical to measure performance from the raw data collected from the system. Enabling visualizations over longer time periods requires the raw data to be queried and processed in a nightly manner using ML monitor pipelines and the automation tool Apache Airflow. Each night, Apache Airflow Directed Acyclic Graphs (DAGs) are kicked off to process daily flight data and produce standard performance metrics that are then stored in a database. The processed metrics allow performance visualization over longer time periods while maintaining a quick and responsive retrieval of the metrics.

B. MLOps Software Tools

The MLOps infrastructure shown in Fig. 4 is enabled by adoption of key software tools including Kedro [24], Scikit-learn [25], MLFlow [26], XGBoost [27], Apache Airflow [28], and Plotly/DASH [29]. Fig. 6 shows the off-line training pipelines and the real-time pipelines and the different software tools adopted for each. The top of Fig. 6 shows the off-line training pipeline structured by Kedro which provides the ML model training framework.

Kedro enforces software engineering best practices applied to ML development. Kedro provides well-defined project and directory structures centered around data query and save, data engineering, and data science pipelines. Each of these three parts is described by a series of processes, called nodes, that are linked to each other through their inputs/outputs and form a DAG. Adoption of the Kedro framework enables scalable, maintainable, and reproducible development of ML models in the off-line training environment.

In addition to providing overall structure to the off-line training environment, Kedro enables abstraction of critical parameter values and ML model hyper-parameters into high level configuration files. This abstraction of parameter values results in highly scalable pipelines where the vast majority of code remains unchanged when training a wide variety of different airports, time ranges, or even machine learning model type.

The Kedro structure also allows for reusability of code between different types of problems, for example using the same code in development of a runway prediction model and taxi-time prediction model. Each pipeline is structured as a collection of atomic operations where each operation is a node. Because different problems often times require similar

Fig. 7. MLOps validation

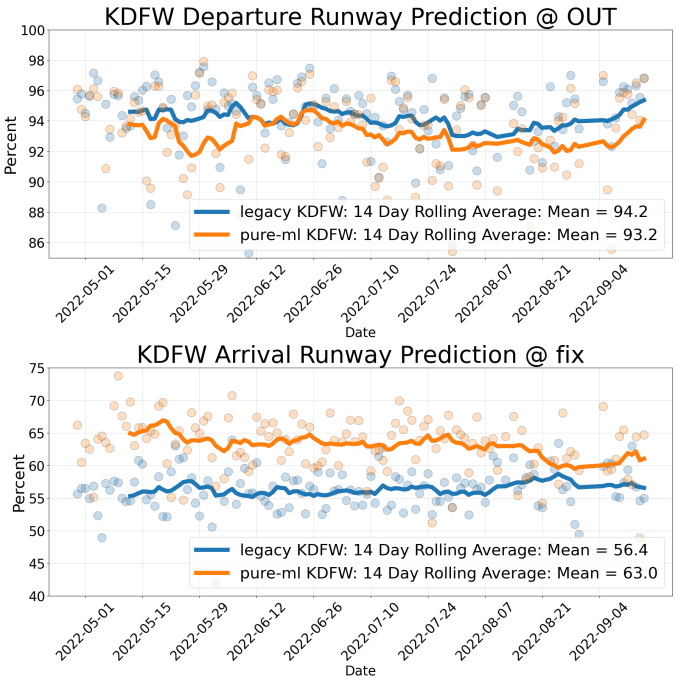

Fig. 8. Top: KDFW departure runway. Bottom: KDFW Arrival runway.

operations involved in the data query, data engineering, or data science pipelines a node that was originally developed for the runway model can be reused for the taxi-time model. The ability to reuse code across different types of models reduces development effort as additional prediction problems are considered.

Scikit-learn provides the Pipeline class which we use to store the feature engineering in both the off-line training pipeline and the real-time pipeline. The dividing line between data engineering functions and feature engineering functions is often arbitrary. We adopt the convention that data engineering is a process applicable to many different models whereas feature engineering is a process specific to a particular model.

Using the Scikit-learn Pipeline class, we combine the feature engineering with the model into a single object. At the end of the training process, we Pickle the combined object thus preserving the exact feature engineering steps applied in the off-line training pipeline. The combined object can then be deployed to the real-time system ensuring the exact code is used in deployment as was used in training.

We use XGBoost for the underlying ML models for all prediction problems. The pickled combination of ML model and feature engineering pipeline are registered to MLFlow along with artifacts calculated during the training process. Fig. 7 shows an example of artifacts recorded via MLFlow from an experiment run of the departure runway model.

On our MLFlow server, an experiment is defined to contain all the training performed for a type of ML model (e.g. departure runway model). Each time a ML model training pipeline is run, it will log as a new run for an experiment. For each experiment run, MLFlow tracks metadata associated with the training data, hyper-parameters, hash of the code repo, performance metrics, and whatever artifacts are calculated during validation. Fig. 7 shows a confusion matrix calculated as a result of validation and along the left side of the Figure there are links to additional artifacts of interest. Within the model directory is the pickled model which can then be registered and the URI (Unique Resource Identifier) of the registered model can be referenced directly for deployment to the real-time system.

Apache Airflow provides the ability to schedule nightly execution of DAGs consisting of a collection of automated reports and calculation of performance metrics. The processed performance metrics are used as the source data for long-term performance monitoring which is visualized via Plotly/DASH. Plotly/DASH is a Python framework that enables web-based visualizations of the performance metrics.

V. Machine Learning Performance vs. Legacy Approach

For evaluation of the ML Airport Surface Model (pure-ml) performance, we compare against performance of the legacy adaptation based Airport Surface Model (legacy) as baseline. We use data collected during the Stormy 2022 Field Evaluation between April 29 2022 through September 16 2022. During this time period at KDFW there were 127,898 departures and 126,721 arrivals and at KDAL there were 40,973 departures and 41,177 arrivals used for validation.

A. Runway Prediction Model

Fig. 8 shows the performance of the departure and arrival runway prediction models in the top and bottom of the Figure, respectively. During the Stormy 2022 Field Evaluation, KDAL had runway construction the majority of the time period forcing the airport into a single runway operation. The prediction problem on the single runway operation becomes trivial so we focus the evaluation on the performance at KDFW.

The top subplot of Fig. 8 shows the performance of the KDFW departure runway models where the horizontal axis is the date and the vertical axis is the percentage of predictions for departure runway which matched the actual runway used. The predictions for departure runway are sampled once per departure flight at the OUT event (i.e. pushback). Prediction accuracy for the legacy system and pure-ml system are shown in blue and orange, respectively. For each day in the date range, we measure the percentage of all departures for the given day and plot that with a small circle. The 14 day rolling average of the prediction percentage is plotted with a solid line and provides and indication of performance averaged across multiple days and situations.

The overall accuracy of the pure-ml departure runway prediction model (93.2%) is very close to the accuracy of the legacy departure runway model (94.2%). Not only is the overall accuracy quite close, but the rolling average shows similar trends between the pure-ml model performance compared to legacy model performance which indicates both systems are impacted in similar ways.

The performance of the departure runway model is quite encouraging because the legacy system at KDFW is not truly predicting the departure runway. Instead, the legacy system allows for ATC to set a taxi-plan which defines a departure fix to runway mapping which $assigns$ departures to runways according to the way in which ATC wants to load balance the demand across runways. Fig. 8 shows us that the pure-ml departure runway model is only one percentage point below the accuracy of the legacy ATC assignment via the taxi-plan. This is encouraging and gives us confidence that the pure-ml system could be deployed with or without receiving ATC input regarding load balancing strategies for departures and maintain comparable performance.

The bottom subplot of Fig. 8 shows the performance of the KDFW arrival runway model where the horizontal axis is the date and the vertical axis is the percentage of predictions for arrival runway which matched the actual runway used. The predictions for arrival runway are sampled once per arrival flight at the arrival fix crossing event.

The overall accuracy of the pure-ml arrival runway prediction model (63.0%) is outperforming accuracy of the legacy arrival runway model (56.4%). The legacy arrival runway model uses Time Based Flow Management (TBFM) predictions to generate the arrival runway. Not only is the overall percentage performance of the pure-ml system above the legacy system, but the cloud orange dots shows separation and sits above the cloud of blue dots indicating that the pure-ml system is outperforming the legacy system for arrival runway on a consistent day by day basis.

B. Arrival ON Prediction Model

Fig. 9 shows the performance of the KDFW arrival ON prediction model. The legacy arrival ON model uses a decision tree which selects TBFM Scheduled Time of Arrival (STA) when available, else uses Traffic Flow Management System (TFMS) Earliest Time of Arrival (ETA). The arrival ON model

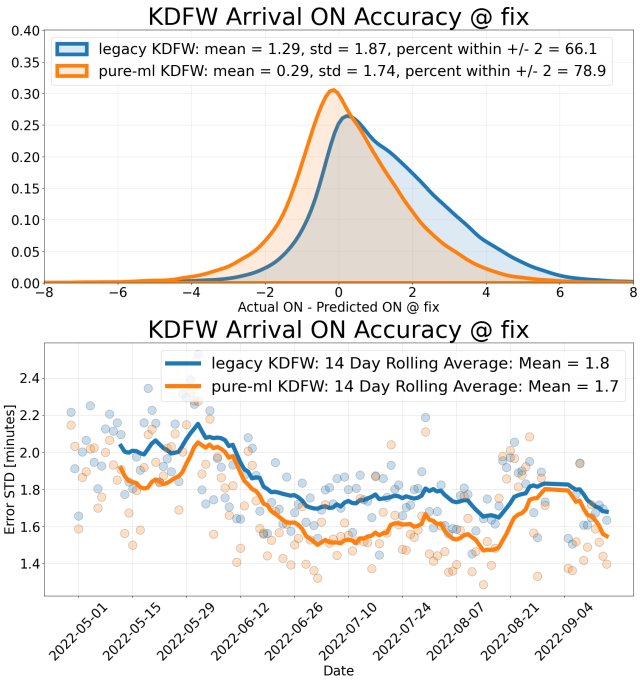

Fig. 9. KDFW arrival ON time.

is evaluated using the error in prediction measured as the difference between the actual ON time minus the predicted ON time sampled at the arrival fix crossing. Each arrival flight contributes to a single data point in each of the three subplots shown in Fig. 9.

The top subplot of Fig. 9 shows the distribution of arrival ON time prediction error for the legacy system and pure-ml system in blue and orange, respectively. The horizontal axis is the error measured in minutes where a positive value indicates the actual ON time was later than the predicted ON time. As can be seen comparing the legacy distribution to the pure-ml distribution, the pure-ml arrival ON time prediction is outperforming the legacy arrival ON time with a standard deviation of 1.74 minutes for the pure-ml system compared to 1.87 minutes for the legacy system.

The main improvement for the arrival ON time prediction model using the pure-ml system is the mean error. The legacy system has mean error of 1.29 minutes compared to pure-ml system with mean error 0.29 minutes. Due to the reduction in mean error the percentage of arrival flights with prediction within a $+/-$ 2 minute window increased from 66.1% for the legacy system up to 78.9% for the pure-ml system.

The bottom subplot of Fig. 9 shows the performance of the KDFW arrival ON prediction model where the horizontal axis is the date and the vertical axis is the standard deviation of the error measured in minutes. Similar to the error distributions shown in the above plot, we see throughout the time period the pure-ml arrival ON model had a lower standard deviation compared to the legacy arrival ON model. The 14 day rolling average for the pure-ml system colored in orange is consistently below the legacy system colored in blue. Both rolling

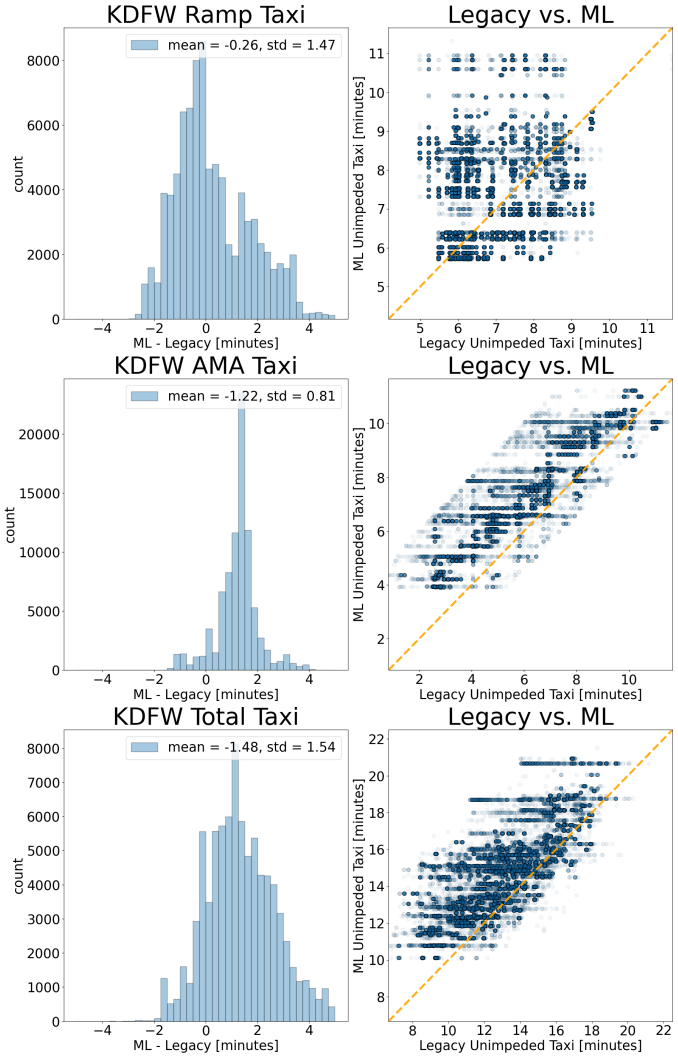

Fig. 10. KDFW unimpeded taxi-out.

averages show similar trends indicating performance of the two models are impacted in similar ways.

C. Unimpeded Taxi-Out Prediction Model

Fig. 10 shows the performance of the KDFW unimpeded taxi-out time prediction. Unlike other models which have a well defined truth value, most aircraft do not experience an unimpeded taxi from gate to runway, so it is difficult to build a data set for validation. Instead of comparing the predicted value to an actual value, for the unimpeded taxi-out model we compare the pure-ml prediction directly to the legacy adaptation based prediction. The goal is not to evaluate the accuracy, but instead evaluate the difference from legacy methods.

The top row of subplot of Fig. 10 shows the results of comparison for the KDFW unimpeded ramp taxi-out prediction. The left side of the top row shows a histogram where the horizontal axis is the difference in unimpeded ramp taxi-out prediction measured as the pure-ml prediction minus the legacy prediction in minutes. The right side of the top row of subplot shows a scatter plot where the horizontal axis is the legacy prediction in minutes and the vertical axis is the pure-ml prediction in minutes. The middle and bottom row of Fig. 10 shows the same type of plots for the AMA taxi-out and the total taxi-out, respectively.

As can be seen in the bottom row of subplots in Fig. 10, the total difference between unimpeded taxi-out prediction has a mean value of 1.48 minutes with standard deviation of 1.54 minutes. The difference is roughly bounded by -2 and 4 minutes. Looking at the second row of Fig. 10 we see the AMA predictions are quite similar between pure-ml and the legacy with mean value 1.22 minutes and a standard deviation 0.81 minutes.

The majority of difference in the total taxi-out predictions seems to be associated with the ramp taxi-out prediction. The top row of Fig. 10 shows the difference in ramp taxi-out has a mean value of 0.26 minutes and standard deviation 1.47 minutes. The difference in ramp taxi-out shows similar bounds to the total taxi with the difference in ramp taxi-out falling between -2 and 4.

Judging the performance of unimpeded taxi-out model is difficult because when we observe a difference between the pure-ml prediction and the legacy prediction we don't know which value is closer to the truth, we just observe the difference. Since the unimpeded taxi-out time prediction is used as input to the scheduler which produces the ETOT prediction, we take the approach that evaluation of the unimpeded taxi time predictions can partially be achieved by evaluating the ETOT prediction accuracy shown in the following Section V-D.

D. Estimated Take Off Time Prediction

For analysis of ETOT prediction accuracy we restrict our attention to AAL major flights at KDFW and SWA major flights at KDAL, as these flights were participants in the field evaluation and providing EOBT predictions. Fig. 11 and 12 show the performance of the ETOT prediction at KDFW and KDAL, respectively. The top subplot shows the distribution of the error in the ETOT prediction for the legacy system and the pure-ml system colored in blue and orange, respectively. The horizontal axis represents the difference in minutes between the actual OFF time and ETOT sampled at the OUT event. The bottom subplot shows the performance of the ETOT prediction sampled at OUT where the horizontal axis is the date and the vertical axis is the standard deviation of the error measured in minutes. The standard deviation of the error is shown for each day with a small circle and the 14 day rolling average is shown with a line.

Fig. 11 shows the KDFW ETOT prediction accuracy is quite similar between the pure-ml system and the legacy system. The pure-ml system has a slightly lower standard deviation of 6.3 minutes compared to the legacy system of 6.61 minutes. The pure-ml system has mean error -1.69 minutes compared to -1.34 minutes. Even with the slightly larger bias, the tighter standard deviation results in the pure-ml system having higher

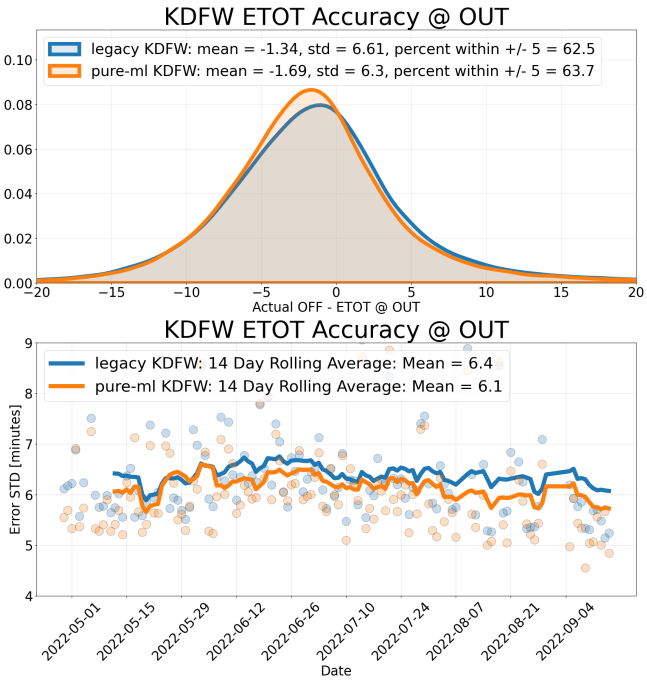

Fig. 11. KDFW ETOT accuracy.

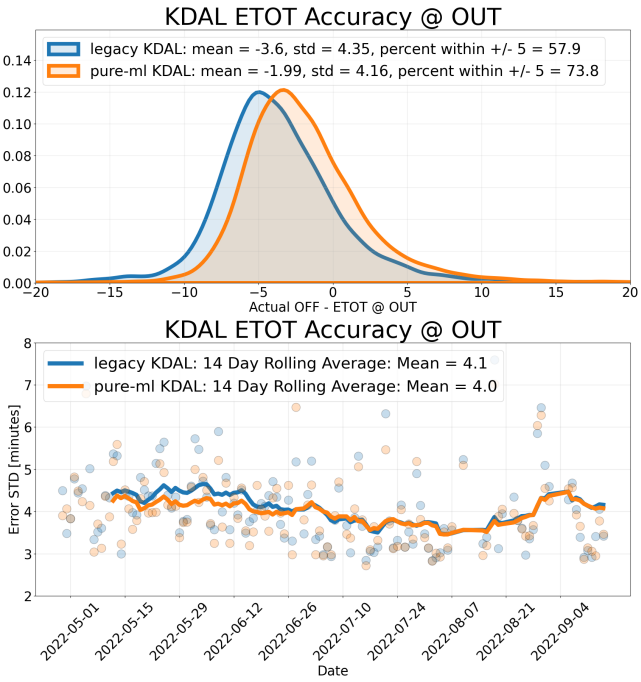

Fig. 12. KDAL ETOT accuracy.

percentage of flights fall within $+/-5$ minutes (63.7%) compared to the legacy system (62.5%). The bottom subplot of Fig. 11 shows the improved standard deviation of error for the pure-ml system was observed throughout the time range during a variety of operating conditions.

Similarly, Fig. 12 shows the pure-ml KDAL ETOT prediction accuracy is slightly improved compared to the legacy system. The main difference between KDFW results in Fig. 11 and the KDAL results in Fig. 12 is that the KDAL pure-ml system has a much smaller mean error of -1.99 minutes compared to the legacy system of -3.6 minutes. This reduction in mean error for the pure-ml system has a significant impact on the percentage of flights falling within $+/-5$ minutes (73.8%) compared to the legacy system (57.9%). The bottom subplot of Fig. 12 shows the standard deviation of error for the pure-ml system was very similar throughout the time range during a variety of operating conditions.

VI. Conclusion

This paper describes the work required for transformation of NASA's legacy IADS system to a real-time ML Airport Surface Model deployed on NASA's Digital Information Platform. The system was deployed as a service-oriented architecture in alignment with FAA's Info-Centric NAS and leveraged ML to replace legacy adaptation to provide a scalable solution. The individual services making up the ML Airport Surface Model can be leveraged as building blocks to accelerate innovation for higher level capabilities.

Deployment of the ML in a real-time operational system required adoption of MLOps best practice to ensure scalable, maintainable, and reproducible development of ML models and end-to-end lifecycle management within the software system. Key software tools including Kedro, Scikit-learn, and MLFlow were adopted within the MLOps infrastructure for both off-line training pipelines and real-time deployment pipelines. The result was an integrated ML software system that was deployed and evaluated in an operational field evaluation and performance was benchmarked against the legacy solution.

Performance of the ML Airport Surface model compared to legacy IADS system was very encouraging. For arrival predictions, the ML outperformed the legacy approach with the arrival runway prediction increasing from 56.4% to 63.0% accuracy and the arrival on time prediction improving where the legacy approach had 66.1% flights and ML approach had 78.9% flights with prediction within $+/-$ 2 minutes of the actual value. For departure predictions, ML departure runway predictions were within one percent of ATC assignments of departure runways and the ETOT predictions at both KDFW and KDAL showed improvement with the ML approach when measuring the standard deviation of prediction error and the percent of flights with prediction within $+/-$ 5 minutes of the actual value.

Overall, the results of the ML Airport Surface Model have been encouraging as it serves as a proof-of-concept for transformation of legacy capabilities into digital services deployed in a cloud-based environment. The performance of the ML Airport Surface Model compared to the legacy approach confirms there is a viable path to replace legacy adaption with ML and maintain or improve system performance. Replacing legacy adaptation with ML is an important topic as many of the legacy FAA systems such as TBFM, TFMS, and Terminal

Flight Data Manager (TFDM) rely on adaptation and ML could provide a more efficient alternative.

REFERENCES

[1] R. Coppenbarger, Y. Jung, T. Kozon, A. Farrahi, W. Malik, H. Lee, E. Chevalley, and M. Kistler, "Benefit opportunities for integrated surface and airspace departure scheduling: a study of operations at charlotte-douglas international airport," in *Digital Avionics Systems Conference (DASC)*, 2016.

[2] Y. Jung, W. Malik, L. Tobias, G. Gupta, T. Hoang, and M. Hayashi, "Performance evaluation of sarda: an individual aircraft-based advisory concept for surface management," *Air Traffic Control Quarterly*, vol. 22, no. 3, pp. 195–221, 2014.

[3] M. Hayashi, T. Hoang, Y. C. Jung, W. Malik, H. Lee, and V. L. Dulchinos, "Evaluation of pushback decision-support tool concept for charlotte douglas international airport ramp operations," in *11th USA/Europe Air Traffic Management Research and Development Seminar*.

[4] J. Thipphavong, J. Jung, H. N. Swenson, K. E. Witzberger, M. I. Lin, J. Nguyen, L. Martin, M. B. Downs, and T. A. Smith, "Evaluation of the controller-managed spacing tools, flight-deck interval management, and terminal area metering capabilities for the atm technology demonstration 1," in *11th USA/Europe Air Traffic Management Research and Development Seminar*.

[5] S. A. Engelland, R. Capps, K. B. Day, M. S. Kistler, F. Gaither, and G. Juro, "Precision departure release capability (pdrc) final report," 2013.

[6] Y. Jung, W. Coupe, A. Capps, S. Engelland, and S. Sharma, "Field evaluation of the baseline integrated arrival, departure, surface capabilities at charlotte douglas interntional airport," in *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*, 2019.

[7] W. J. Coupe, D. Bhadoria, Y. Jung, E. Chevalley, and G. Juro, "Atd-2 field evaluation of pre-departure trajectory option set reroutes in the north texas metroplex," in *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. IEEE, 2022, pp. 1–10.

[8] FAA Air Traffic Organization Surface Operations Office, "U.s. airport surface collaborative decision making (cdm) concept of operations (conops) in the near-term: application of the surface concept at united states airports," 2014.

[9] W. J. Coupe, Y. Jung, H. Lee, L. Chen, and I. J. Robeson, "Scheduling improvements following the phase 1 field evaluation of the atd-2 integrated arrival, departure, and surface concept," in *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*, 2019.

[10] "Charting aviation's future: Operations in an info-centric national airspace system," https://www.faa.gov/sites/faa.gov/files/Charting-Aviations-Future-Operations-in-ICN_0.pdf, accessed: 2023-01-22.

[11] "Nasa's digital information platform," https://nari.arc.nasa.gov/atmx-dip, accessed: 2023-01-22.

[12] B. Sridhar, "Applications of machine learning techniques to aviation operations: Promises and challenges," in *2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*. IEEE, 2020, pp. 1–12.

[13] "Mlops: Continuous delivery and automation pipelines in machine learning," https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning, accessed: 2023-01-22.

[14] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine learning: The high interest credit card of technical debt," 2014.

[15] Y. Zhou, Y. Yu, and B. Ding, "Towards mlops: A case study of ml pipeline platform," in *2020 International conference on artificial intelligence and computer engineering (ICAICE)*. IEEE, 2020, pp. 494–500.

[16] M. M. John, H. H. Olsson, and J. Bosch, "Towards mlops: A framework and maturity model," in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2021, pp. 1–8.

[17] W. J. Coupe, Y. Jung, L. Chen, and I. Robeson, "Atd-2 phase 3 scheduling in a metroplex environment incorporating trajectory option sets," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–10.

[18] E. Chevalley, G. L. Juro, D. Bakowski, I. Robeson, L. X. Chen, W. J. Coupe, Y. C. Jung, and R. A. Capps, "Nasa atd-2 trajectory option set prototype capability for rerouting departures in metroplex airspace," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–10.

[19] J. Rebollo, S. Khater, and W. J. Coupe, "A recursive multi-step machine learning approach for airport configuration prediction," in *AIAA AVIATION 2021 FORUM*, 2021, p. 2406.

[20] A. Churchill, W. J. Coupe, and Y. C. Jung, "Predicting arrival and departure runway assignments with machine learning," in *AIAA AVIATION 2021 FORUM*, 2021, p. 2400.

[21] A. Amblard, S. Youlton, and W. J. Coupe, "Real-time unimpeded taxi out machine learning service," in *AIAA AVIATION 2021 FORUM*, 2021, p. 2401.

[22] D. Wesely, A. Churchill, J. Slough, and W. J. Coupe, "A machine learning approach to predict aircraft landing times using mediated predictions from existing systems," in *AIAA AVIATION 2021 FORUM*, 2021, p. 2402.

[23] W. J. Coupe, D. Bhadoria, Y. Jung, E. Chevalley, and G. Juro, "Shadow evaluation of the atd-2 phase 3 trajectory option set reroute capability in the north texas metroplex," in *Fourteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM 2021)*.

[24] "Kedro: open-source python framework for creating reproducible, maintainable and modular data-science code," https://kedro.org/, accessed: 2023-01-22.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 2011.

[26] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym *et al.*, "Developments in mlflow: A system to accelerate the machine learning lifecycle," in *Proceedings of the fourth international workshop on data management for end-to-end machine learning*, 2020, pp. 1–4.

[27] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, 2016.

[28] "Apache airflow," https://airflow.apache.org/docs/apache-airflow/stable/python-api-ref.html, accessed: 2023-01-22.

[29] "Plotly/dash: The premier data app platform for python," https://plotly.com/dash/, accessed: 2023-01-22.

AUTHOR BIOGRAPHIES

Dr. William Jeremy Coupe is an Aerospace Engineer at NASA Ames Research Center. He received his BS degree in Mathematics from the University of San Francisco and both MS degree in Applied Mathematics and Statistics and Ph.D. degree in Computer Engineering from the University of California, Santa Cruz.

Dr. Alexandre Amblard is a Data Scientist for USRA. He received his BS degree in Physics from the University of Caen-Normandy and both his MS and Ph.D. degree in Physics from the University of Paris-Saclay.

Sarah Youlton is a Software Engineer at USRA. She received both her BS degree in Physics and Mathematics and her MS degree in Applied Mathematics from Santa Clara University.

Matthew Kistler is a Principal Analyst at Mosaic ATM supporting NASA Ames Research Center. He received his BS degree in Aeronautical Engineering from California Polytechnic State University in San Luis Obispo, California.