Predicting Gate Conflicts at Charlotte Douglas International Airport Using NASA ATD-2 Fused Data Sources

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

Andrew Churchill, Isaac Robeson
Mosaic ATM
Leesburg, VA, USA
achurchill@mosaicatm.com
irobeson@mosaicatm.com

***Abstract*—NASA is conducting the Airspace Technology Demonstration-2 to evaluate an Integrated Arrival, Departure, and Surface (IADS) traffic management system. The IADS system is powered by real-time System Wide Information Management feeds which provide an accurate and high fidelity view of the lifecycle of a flight. This data can be leveraged to drive efficiencies in the National Airspace System. For non safety critical applications there is opportunity for third party service providers to offer this type of data-driven prediction service in near real-time. This paper investigates the gate conflict prediction problem as a concrete use case which could help drive efficiencies. We model gate conflicts as a regression problem and describe the iterative process of model building, model validation, and evaluation used to assess the efficacy of our approach. We quantify our predictive accuracy and identify paths for improvement. Through this iterative process we hope to evolve our models and methods to a near real-time prediction service.**

***Index Terms*—Airspace Technology Demonstration-2; Data Driven Services; Predictive Analytics; Airport Gate Conflicts**

I. Introduction

Concepts and technologies to manage arrival, departure, and surface operations 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. To demonstrate these technologies NASA is conducting the Airspace Technology Demonstration-2 (ATD-2) to evaluate an Integrated Arrival, Departure, and Surface (IADS) traffic management system [1], [2].

The IADS concept extends traffic sequencing for the entire life-cycle of a flight from departure gate to arrival gate within multi-airport, metroplex environments. The IADS concept builds on and integrates previous NASA research such as the Terminal Sequencing and Spacing (TSAS) [3], the Precision Departure Release Capability (PDRC) [4], and the Spot and Runway Departure Advisor (SARDA) [5], [6] which each focused on individual airspace domains. The IADS concept was initially developed based on the Surface Collaborative Decision Making (S-CDM) ConOps [7] and refined over time.

The IADS system was deployed to the Charlotte Douglas International Airport (CLT) for a three year field evaluation beginning in September 2017. During the field evaluation the IADS system has succesfully demonstrated three key capabilities: 1) data exchange and integration, 2) departure surface metering, and 3) departure scheduling and electronic negotiation of release time of controlled flights for overhead stream insertion [8]. The IADS system in CLT is a precursor of the FAA Terminal Flight Data Manager (TFDM) which will begin to be rolled out across the United States' busiest airports, starting in 2020, after the conclusion of NASA's field evaluation [9].

A core idea of the IADS concept is to generate a coordinated schedule which enables a smooth flow of traffic from gate to runway for departures and from runway to gate for arrivals [10]. To ensure a smooth flow an arrival flight needs an available gate or else the arrival flight will be considered to be in a gate conflict. To resolve the gate conflict ramp controllers typically hold the arrival in either the Airport Movement Area (AMA) or the ramp area. As the number of gate conflicts increases the complexity and workload for ramp controllers increase [11].

In the future TFDM system the ATC Traffic Management Coordinator (TMC) will be required to set parameters for surface metering which could impact the number of gate conflicts. To help inform the TMC about the impact of the selected parameters TFDM requirements include a prediction of the number of gate conflicts in upcoming Surface Metering Programs (SMP) [12]. The TMC can use this prediction to help calibrate the parameters that govern the amount of excess taxi time that is passed from the AMA back to the gate.

Too often, the development and implementation of new capabilities within the traditional FAA systems has involved cost and or schedule overruns that detract from the anticipated operational value, or involve times to acquire or develop that are not sufficiently responsive to the needs of a changing environment [13]. For non safety critical applications, there is opportunity for third party service providers to offer this type of data-driven prediction in near real-time. Both the IADS system and the future TFDM system are powered by real-time FAA System Wide Information Management (SWIM) [14]

data feeds. The real-time data published across the SWIM cloud enables a software layer on the other side of SWIM to ingest these feeds and provide near real-time decision support outside of the traditional FAA systems.

In this paper we explore the gate conflict prediction problem as a concrete example for a data-driven service that could help drive efficiencies. The purpose of this paper is to explain the iterative process of model building, model validation, and evaluation used to assess the efficacy of our approach. We aim to quantify our predictive accuracy and identify paths for improvement. Through the iterative approach we expect for our models and methods to evolve as the data informs us.

The remainder of the paper is organized as follows. In Section II we provide background information on gate conflict research. Section III describes the different data feeds the IADS system pulls from SWIM and the NASA-developed Fuser technology that is foundational to building a cohesive view of a single flight. In Section IV we implement various regression models and estimate the accuracy for out of sample data. In Section V we describe directions for future research. Lastly, Section VI provides concluding remarks.

II. Background on Gate Conflicts

Previous research has found the number of gate conflicts is influenced by the airline schedules. There is a relationship in the scheduling practice involving bank operations and the time interval between banks that influences the requirement for gates [15]. Moreover, the fluctuation of the actual demand from the scheduled demand creates additional potential for gate conflicts.

To mitigate the impact of gate conflicts, researchers have investigated different forms of the gate assignment problem [16]. Various objective functions have been considered, including a robust gate assignment which aims to minimize the number of gate-reassigned aircraft [17], [18] and a gate assignment that aims to minimize the variance of idle times at the gates [19]. Others considered the gate assignment problem to minimize physical conflicts in the ramp area and reduce interaction between the arrivals and departures [20].

Whereas previous research focused on strategies to mitigate gate conflicts, they did not work towards predicting time periods or specific banks when gate conflicts would be expected to be elevated. In this paper we present a model to predict both the count of gate conflicts and the rate of gate conflicts per arrival flight within a bank of operations. This work is enabled by newly available data that is generated by the IADS system. High fidelity data elements such as the Earliest Off Block Time (EOBT) in combination with FAA SWIM feeds are leveraged for gate conflict predictions.

Identifying and labeling gate conflicts are non trivial tasks. Since two aircraft can't occupy the gate at the same time, the Actual Off Block Time (AOBT) and Actual In Block Time (AIBT) for the departure and arrival, respectively, will not provide insights to whether or not a gate conflict occurred. Labeling of gate conflicts by ramp controllers would be the most accurate approach. However, this approach is not realistic as capturing such a data element would increase ramp controllers' workload to unacceptable levels.

To address this challenge the IADS system leverages a combination of actual and predicted events to label an arrival as a gate conflict. At the time point that the arrival touches down at CLT, the IADS system generates an Unimpeded In Block Time (UIBT) which represents the time the system expects the arrival to get to the assigned gate. At the time point that the arrival touches down, the system also has a prediction of when the departure will push back from the same gate in the form of the Unimpeded Off Block Time (UOBT) outside of surface metering or a Target Off Block Time (TOBT) during surface metering. If the UIBT $\leq$ UOBT + buff_{push} (TOBT + buff_{push} during surface metering) then the arrival flight is labeled as a gate conflict. The parameter buff_{push} defines the expected time the departure will occupy the gate during the pushback process.

In addition to departures, aircraft not currently associated with a flight are persisted at the gate. The location of these persisted aircraft are manually updated in the system by the ramp managers and controllers when they are towed to and from the gates. If an arrival is destined for a gate with a persisted aircraft, the arrival will be labelled as a gate conflict.

Other definitions of a gate conflict could be used which account for the intensity of the gate conflict by requiring UIBT $\leq$ UOBT + $\text{buff}_{push} - \Delta$ to label an arrival a gate conflict. In this definition Δ is a parameter defining the minimum duration of the gate conflict and the value could be determined by Subject Matter Experts (SMEs). For example, if SMEs agree that a gate conflict less than five minutes should not be labeled as a gate conflict, then we can define $\Delta = 5$.

Even with leveraging the actual and predicted events the identified gate conflicts might not materialize. Similarly, arrivals that are identified to not have a gate conflict could still experience a gate conflict if the departure occupying the gate pushes back later than expected. Although these challenges are present in the data our methodology to identify gate conflicts is reasonable given the current system.

III. Data Sources and Preprocessing

A. Data Sources

The ATD-2 IADS system is powered by real-time SWIM data feeds including Traffic Flow Management System (TFMS), SWIM Terminal Data Distribution System (STDDS), SWIM Flight Data Publication Service (SFDPS), Time Based Flow Management (TBFM), Terminal Flight Data Manager (TFDM), and Terminal Automation Information Service (TAIS) [21]. These SWIM data feeds are complemented by other data sources such as ramp surveillance and gate information provided by the airline operators.

The SWIM feeds contain valuable data but can provide inconsistent information on the same flight that is difficult for consumers to understand. Without deep knowledge of the underlying 3T (TFMS, TBFM, and TFDM) systems, plus FAA air traffic systems En Route Automation Modernization (ERAM) and Standard Terminal Automation Replacement

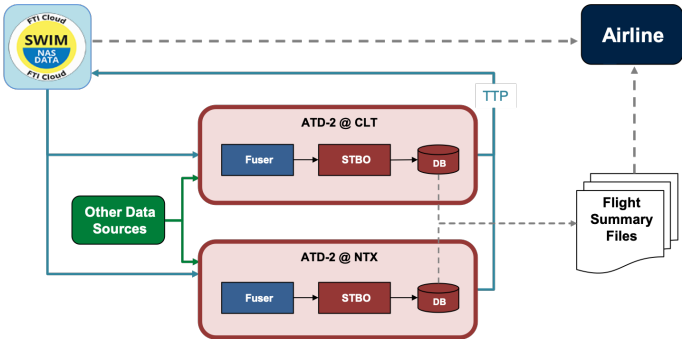

Fig. 1. Data architecture and data flow between SWIM, ATD-2, and airline.

System (STARS), the consumption logic may not lead toward the benefit the community desires [22]. To address this potential mismatch, ATD-2 invested in developing the logic that could address SWIM flight data processing and mediation complexities. Much of this work is embodied in the Fuser service which is illustrated in Fig. 1. 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 common well defined data model.

The fused data sources are used as input by the ATD IADS system and the output is written to a database. In addition to consuming SWIM data the IADS system publishes the TFDM Terminal Publication (TTP) feed back to SWIM and can be consumed within the SWIM Research and Development (R&D) environment. The ATD-2 TTP feed on the SWIM R&D matches the specifications of the future TTP feed to foster industry innovation in preparation for TFDM.

The data written to the database is valuable but often too verbose to be used effectively for analysis. To address this problem, ATD-2 developed the Flight Summary report to serve analysts and user needs [23]. The Flight Summary helps standardize the ATD-2 approach to handling such conditions as human inputs, business logic, measurement convention, complexities of data mediation, order of processing messages, and changes from earlier versions of ATD-2 software. The Flight Summary is generated every morning for post-operations analysis about the previous day operations and contains each unique flight as a row containing over 500 columns of unique data elements and predictions captured at some discrete points in time. A subset of the flights included in the Flight Summary files are distributed to the airline operators for their internal analysis.

B. Data Preprocessing for Machine Learning

In this paper we use data from the Flight Summary files between 2018-01-01 and 2019-09-30 and focus on departures and arrivals operating within an identified scheduled or actual bank. A bank of traffic is a concentrated period of demand and can be identified for the departures, arrivals, or the total operations. Airlines typically define banks by the original scheduled demand and static boundaries in time that define the stop and start of the bank. In contrast, the IADS system is capable of defining banks by both the scheduled demand and actual demand and the boundaries that define the start and end time of the bank are dynamic and adapt with the profile of scheduled or actual demand.

To identify the banks and the associated start and stop times we use the DBSCAN [24] clustering algorithm to label banks based on the density of gate or runway operations. Using this approach we can cluster banks on the density of scheduled or actual demand and can also define and identify banks using a gate-centric or runway-centric view of operations. We find the clustering approach to be more informative than using a bank definition with static start and stop times. As the actual demand fluctuates from the scheduled demand, the bank has the potential to shift earlier or later than the static bank definition and the clustering approach is able to capture this shift and adapt our bank start and stop times to better represent what happened.

The first step in the workflow is data preprocessing. We begin by selecting a subset of the available data elements and metrics available in Flight Summary to use as features in our gate conflict prediction model. The selected features are shown in the data dictionary provided in the appendix. We filter out any bank containing a feature that deviates more than 3.5 times the InterQuartile Range (IQR) away from the median. Before filtering we had 3197 distinct banks and after the filtering process we are left with 1956 unique banks representing 61% of the original data. After filtering we normalize the data such that each metric is transformed to have mean equal to zero and unit variance.

The filter we implemented is relatively conservative given that it only excludes banks that contain a metric deviating more than 3.5 times the IQR. It was surprising to see the filter eliminated 39% of the overall data. This is indicative of the overall uncertainty and noise in the data. The prediction accuracy and results shared in Section IV-E should be taken in context with the understanding that the underlying data is noisy and that the quality of the signal extracted could be impacted by the uncertainty.

IV. Gate Conflict Regression

In this section we investigate two different flavors of the gate conflict regression problems. In the first problem we define the target as the count of gate conflicts within a distinct bank. In the second problem we define the target as the fraction of gate conflicts per arrival to normalize our predictions against the overall arrival demand. For each target we consider the 37 features described in the data dictionary. For each target we consider a Support Vector Machine (SVM) regression and a Gradient Boost (GB) regression.

To validate the entire workflow we implement an iterative cycle of feature selection, hyperparameter tuning, and model validation shown in Fig. 2. After iterating through this cycle ten times the validation metrics are evaluated by a Subject

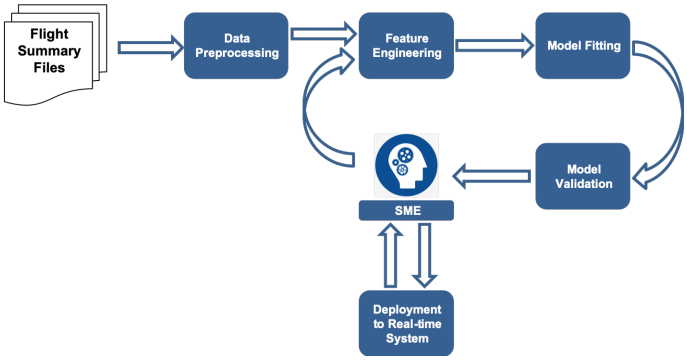

Fig. 2. ATD-2 predictive analytics workflow.

Matter Expert (SME) and potential improvements are identified and investigated. Through this repeated process we aim to develop a model capable of near real-time application.

To implement the validation we use the scikit-learn [25] library for Python. For feature selection, hyperparameter tuning, and measuring prediction accuracy we use Recursive Feature Elimination Cross Validation (RFECV) [26], Grid Search Cross Validation (GridSearchCV) [27], and negative mean squared error (mean_squared_error) [28], respectively.

A. Model Validation

The validation process is described in Python psuedo-code in Algorithm 1. The algorithm is illustrated using the SVM regressor but the same process was implemented for the GB regressor. To better understand the confidence we have in our estimated validation metrics, we implement multiple loops through the feature selection, hyperparameter tuning, and validation. For each loop we save the data to analyze the distribution and estimate the mean with associated 95% confidence interval for the mean.

We iterate through ten loops where we first randomly split the data $70/30$ into a TRAIN and TEST set. For the TRAIN set we call RFECV to select a subset of the features to use. After the features are identified, we iterate through five loops where we randomly shuffle the data and then call GridSearchCV and record the results. The combination of hyperparameters showing the best results are then used to fit the model to the TEST set and the performance is saved for evaluation in Section IV-E.

B. Feature Selection

For the SVM regression it is important to identify the subset of features that provide good performance for the given target. To identify the optimal set of features is computationally expensive so we implement a heuristic approach based on RFECV. Whereas the features identified by RFECV might not be optimal, in practice they seem to be acceptable from a prediction accuracy perspective.

To identify features using RFECV, first the estimator is trained on the initial set of features and k-fold cross validation is implemented to calculate the negative Mean Squared Error (nMSE) for the given set of features. The importance of each

Algorithm 1 Cross Validation Including Model Selection and Hyperparameter Tuning

```
from sklearn.svm import SVR
from sklearn.feature_selection import RFECV
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
results = []
for i=1:30 do
  cv_score = []
  TRAIN = df.sample(frac=0.7,replace=False)
  TEST = df.drop(TRAIN.index)
  rfecv = RFECV(TRAIN)
  rfecv.fit(TRAIN[input],TRAIN[target])
  features = rfecv.support_
  regr_model = SVR()
  for j=1:5 do
    TRAIN_temp = TRAIN.sample(frac=1,replace=False)
    clf = GridSearchCV(regr_model, param_grid, cv=3)
    clf.fit(TRAIN_temp[features],TRAIN_temp[target])
    cv_score.append(clf.cv_results_)
  end for
  selected_hyperparameters = cv_score.sort()
  regr_model_final = SVR(selected_hyperparameters)
  regr_model_final.fit(TRAIN[features],TRAIN[target])
  pred_values = regr_model_final.predict(TEST[features])
  final_scores = - mean_squared_error(TEST[target],pred_values)
  results.append(final_scores)
end for
```

feature is obtained from the fitted coefficients and the least important feature is pruned from the current set of features. That procedure is recursively repeated on the pruned set. The set of features that generated the largest nMSE is selected and used in the remainder of outer loop i, described in Algorithm 1. The results of this process are shown in Fig. 3 and the identified number of features is illustrated with an orange star.

Throughout the iteration process through the outer loop i described in Algorithm 1, we noticed that the features returned by RFECV seemed sensitive to the random selection of the original TRAIN data set. Whereas the general shape and value of the curves describing the nMSE as a function of the number of features was stable, the exact maximum of that curve was vulnerable to fluctuations due to the saturation in prediction accuracy as features increase. In future work, our implementation of RFECV might be improved upon in terms of stability by selecting features within some threshold of the maximum.

C. Hyperparameter fitting

The performance of the SVM and GB regressors can be improved by properly selecting the hyperparameters of the model. For the SVM, the hyperparameters we consider are the kernel type and the parameters C and ϵ and we search over the grid defined by kernel $\in \{\text{linear}, \text{rbf}, \text{poly}\} \times$

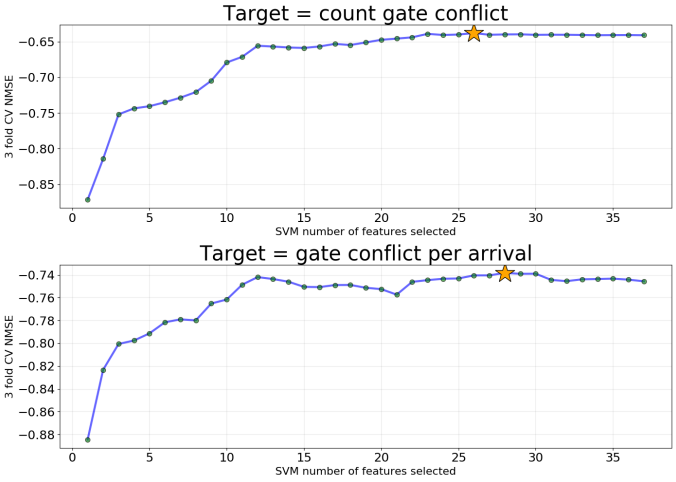

Fig. 3. Support vector regression feature selection.

$C \in \{1, 2, .., 10\} \times \epsilon \in \{0.1, 0.2, .., 2\}$. For the GB, the hyperparameters we consider are the learning rate, number of estimators, max depth, and min samples leaf. For the GB model, we search over the grid defined by learning rate $\in \{0.01, .021, ..2.01\}\times$ number of estimators $\in \{100, 200, 300\}\times$ max depth $\in \{1, 2, 3\}\times$ min samples leaf $\in \{1, 3, 5, 8\}\times$ sub_sample $\in \{0.5, 1\}$.

To perform the grid search over hyperparameters we use GridSearchCV function. GridSearchCV iterates over each combination of hyperparameters, performs 3-fold cross validation and returns the nMSE averaged over the 3 folds. When we implement the 3-fold cross validation on our TRAIN data set containing 1369 rows, we split the data up into 3 discrete subsets of 456 observations. We select one of the three subsets as the test set and use the other 2 subsets as the training set. With only 456 observations in the test set, we observe that the prediction performance can vary.

To better understand the prediction performance of the hyperparameters in the context of the relatively small sample size, we randomly shuffle the data before we pass it to GridSearchCV and repeat this process 5 times. Each of the 5 random shuffles of the data will implement the 3-fold cross validation process on a unique discretization of the data and will generate an average nMSE across the 3 folds. The distribution of the average nMSE across the 5 random shuffles is shown in Fig. 4. The performances of SVM with linear, rbf, and poly kernel are illustrated in yellow, green, and purple respectively and the performance of GB is illustrated in blue.

As can be seen in Fig. 4, the performance between the different methods varies. These results are for visualization purposes only as the final validation will be done using the TEST dataset. For the target count gate conflict the SVM with linear kernel showed the best performance and for the target gate conflict per arrival the GB showed the best performance. For each of these methods we plot the distribution of the average nMSE across the five random samples for the top three hyperparameter combinations with the best combination

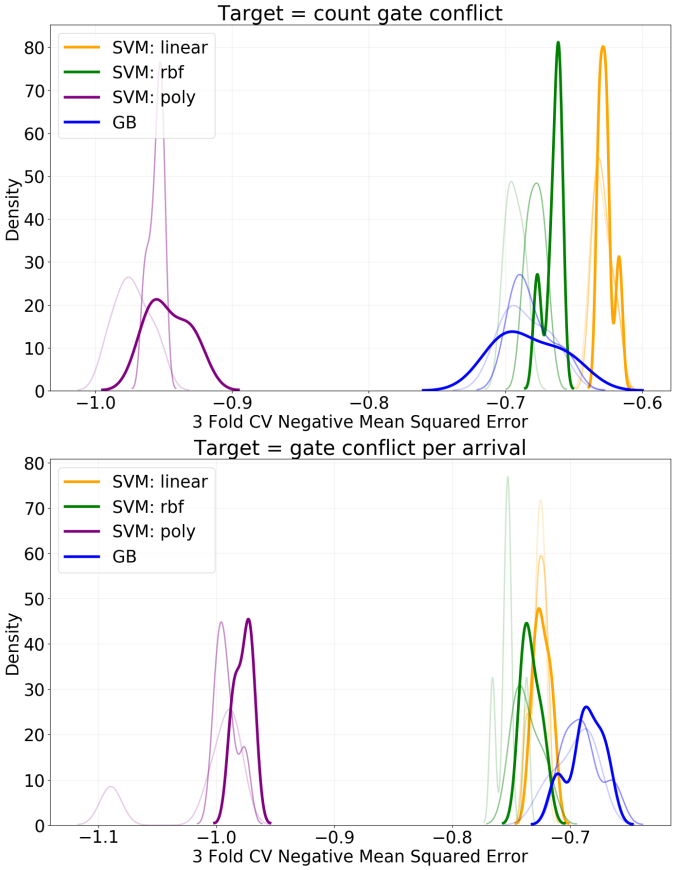

Fig. 4. Hyperparameter results.

plotted with a darker color. For Subsections IV-D and IV-E the SVM and GB methods use the best hyperparameter combination illustrated with darker color in Fig. 4.

D. Feature Importance

For the SVM with linear kernel and the GB regressor we can use the coef_ and feature_importances_ attributes of the model to evaluate the relative importance of the features. For the SVM the relative importance of the features is estimated with the square of the coefficients [29]. The mean value and standard deviation for the SVM squared coefficient and the GB feature importance aggregated from the ten iterations through the outer loop i in Algorithm 1 are shown in Figs. 5 and 6 for the target count gate conflicts and the target gate conflicts per arrival, respectively.

For the target count gate conflicts the top four features for SVM ranked from most to least important were 6, 20, 25, and 2, whereas the top four features for GB ranked from most to least important were 35, 6, 20, and 22. Features 6 and 20 show up for both SVM and GB and are the count of total arrivals and the mean arrival schedule delay predicted at landing. The arrival schedule delay predicted at landing metric is captured when the arrival lands, and measures how early or late the arrival is with respect to its Scheduled In Block Time. The most important feature for GB was the count of arrivals with

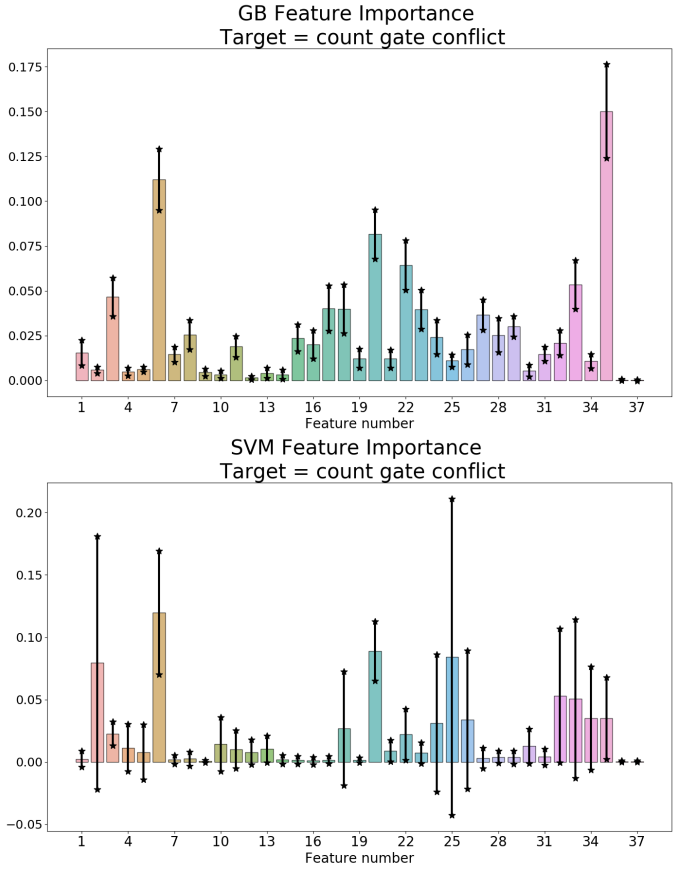

Fig. 5. Feature importance for Target = count gate conflict.

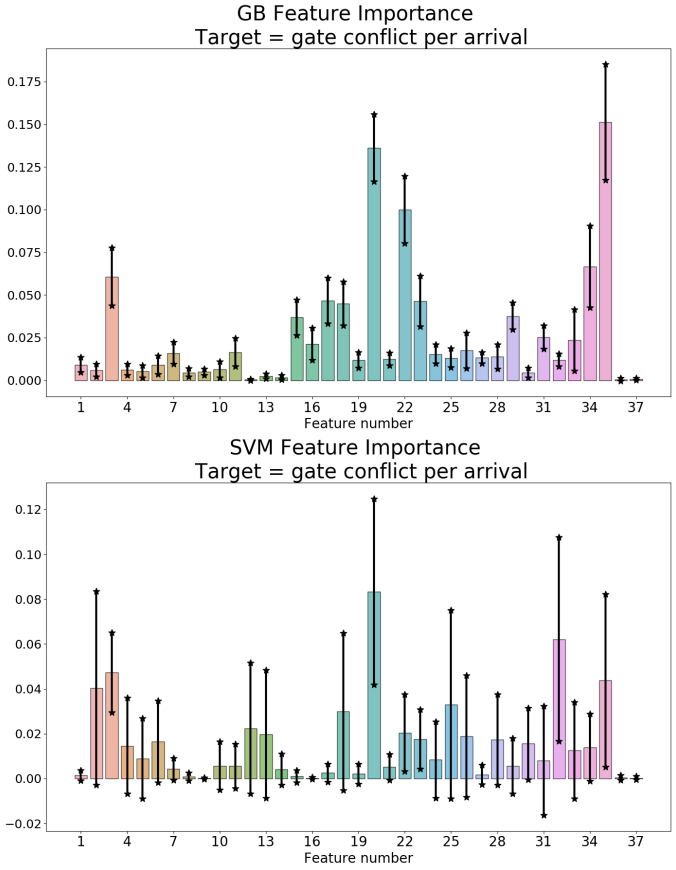

Fig. 6. Feature importance for Target = gate conflict per arrival.

Undelayed In Time at Landing earlier than the end of the actual departure bank, defined by the time the last departure in the bank pushes back.

For the target gate conflicts per arrival, the top four features for SVM were 20, 32, 3, and 35 and the top four features for GB were 35, 20, 22, and 34. Again we find features 20 and 35 important, which are related to how early or late the arrival is at landing with respect to its Scheduled In Block Time and the count of arrivals with Undelayed In Time at Landing earlier than the end of the actual departure bank.

Overall it seems that both the SVM and the GB regressors tend to identify the features related to metrics capable of detecting early arrivals and metrics capable of detecting the interaction between the arrival and departure banks. The early arrivals make sense as a mechanism to increase gate conflicts as the airline schedules only provide small buffers between when a departure occupies the gate and the subsequent arrival is scheduled to arrive.

Other features of interest include features 10-14 describing the controlled flights and surface metered flights. None of these features were identified as important as the ones previously discussed. We notice that for the target gate conflict per arrival, the SVM identified the count of surface metered flights as having some importance, but identified the total gate hold as having no importance. This could be indicative that the underlying congestion and demand which has triggered metering is leading to gate conflicts and not the actual gate holding associated with metering leading to gate conflicts. We note that only a fraction of banks between 2018-01-01 and 2019-09-30 were surface metered, which might obscure the relationship if one were to exist. Given the importance of gate conflicts in the context of surface metering this is an important relationship to keep in mind and to continue to investigate in future research.

E. Validation Results

The results of the validation described in Algorithm 1 in the form of the distribution for the nMSE and the explained variance R^2 are shown in Figs. 7 and 8, respectively. A summary of the results in the form of the estimated 95% confidence interval for the mean of the nMSE and R^2 are shown for the two targets count gate conflicts and gate conflict per arrival in Tables I and II, respectively.

As can be seen in the figures and tables, the nMSE and R^2 metrics are similar between the SVM and GB regressors. Since we normalized the target with mean zero and unit variance we can easily interpret the results of the nMSE in relation to a prediction rule that always uses the average value. A nMSE of less than -1.0 represents a rule with worse performance than always predicting the average, and a nMSE greater than -1.0

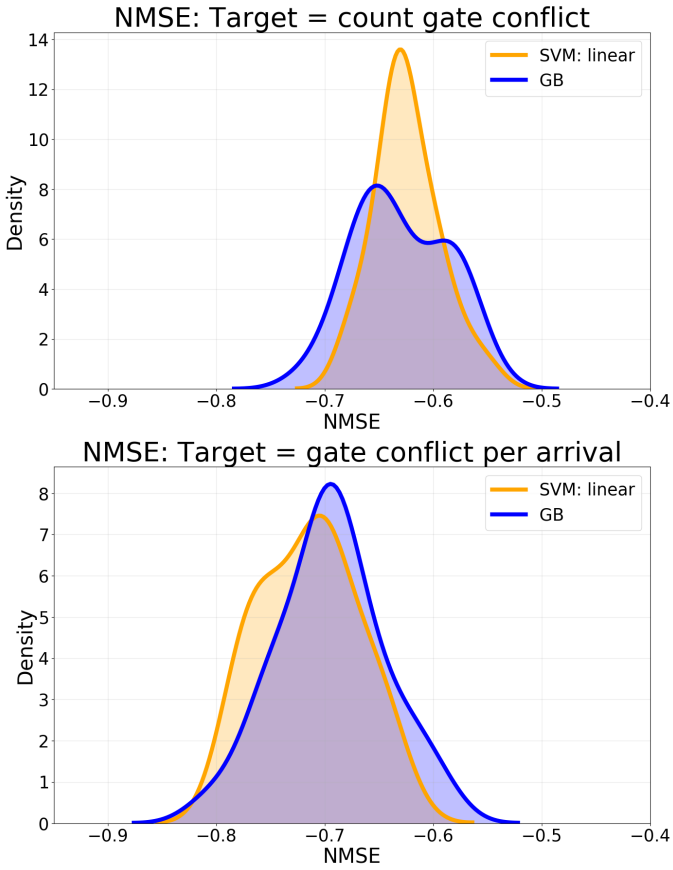

Fig. 7. Validation results for nMSE.

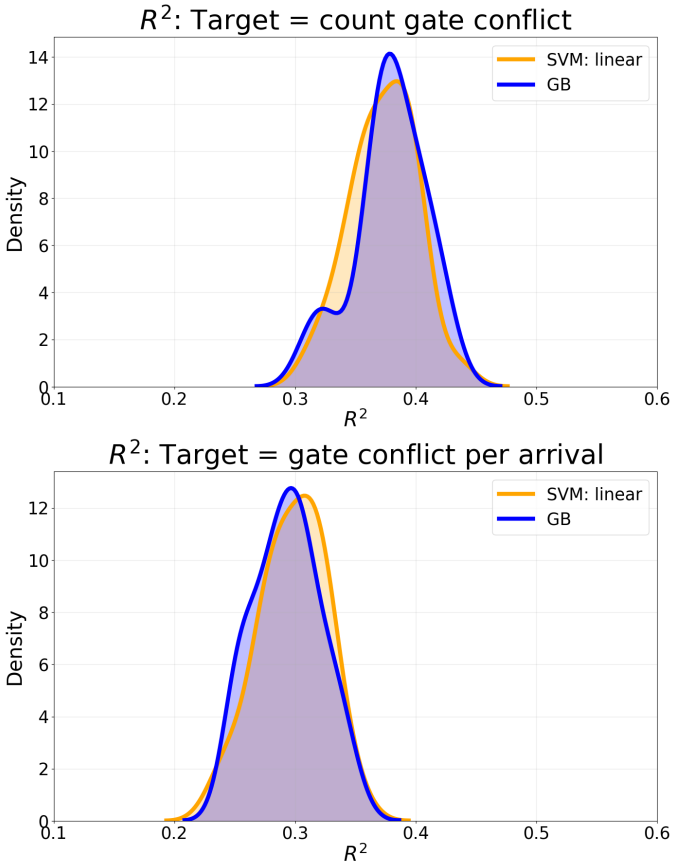

Fig. 8. Validation results for R^2

represents a rule with better performance. For the target count gate conflict we estimate the nMSE to be -0.62 and -0.62 for the SVM and GB, respectively. Similarly, for the target gate conflict per arrival we estimate the nMSE to be -0.71 and -0.69, respectively.

TABLE I
TARGET: COUNT GATE CONFLICT

Model	**nMSE** Target = count gate conflict	**R^2** Target = count gate conflict
SVM	95% CI = $[-0.64, -0.61]$	95% CI = $[0.36, 0.38]$
GB	95% CI = $[-0.65, -0.61]$	95% CI = $[0.37, 0.39]$

TABLE II
TARGET = GATE CONFLICT PER ARRIVAL

Model	**nMSE**	**R^2**
SVM	95% CI = $[-0.73, -0.70]$	95% CI = $[0.29, 0.31]$
GB	95% CI = $[-0.71, -0.68]$	95% CI = $[0.28, 0.30]$

Comparing the sum of the squared error to the variance in the data we can calculate the explained variance R^2. The R^2 results show that the fraction of the variance in the target that is explained by the SVM and GB regressors are 0.37 and 0.38 for the target count gate conflict and 0.30 and 0.29 for the target gate conflict per arrival, respectively.

Overall, we interpret these results as a detectable but weak signal. Without examples of other gate conflict prediction models and their prediction accuracy, it is hard to put the results in context. Given the complexity and uncertainty on the airport surface, different phases of surface operations such as ramp taxi have proven difficult to accurately predict [30]. Moving forward, the validation results of the SVM and GB regressors will be used as a data point to provide context in the evaluation of improved models.

V. FUTURE WORK

The general direction of future work is to continue to pursue concrete examples of data-driven services that could help drive efficiencies in the NAS. At this early stage in the research, the identification of good use cases is as important as the development of the models. We hope through the identification of use cases we can draw the attention of the aviation and data science communities towards the opportunity to develop this software layer of services on the other side of SWIM.

The immediate direction for the gate conflict prediction research is to explore techniques to improve the prediction performance. The results shown in this paper were focused on features derived from descriptive metrics of the bank. A

different approach might focus on predicting individual gate conflicts at the flight level and aggregating the results across the bank. An approach based on an individual flight level gate conflict prediction could leverage the high fidelity schedule and surveillance systems to drive more accurate predictions.

Further analysis is needed to understand the relationship between the different features and their impact on gate conflicts. In this analysis, we identified the relationship between the arrival bank and departure bank as important when predicting gate conflicts. This relationship will be investigated further including exploring new metrics which could capture and quantify this relationship in new or better ways. We also note that the SVM showed a potential relationship between gate conflicts and flights that are controlled or surface metered. If we find that there are specific situations where surface metering is having a negative impact for gate conflicts we can explore the idea to alert the Air Traffic Control (ATC) Traffic Management Coordinator (TMC) that some particular banks should not be surface metered.

Once the model has been improved to the desired level of accuracy, there is a need to transition the model from post operations analysis (post-ops) to a real-time environment. We expect this transition from post-ops to real-time to be non-trivial. The data elements and metrics that we find valuable in post-ops might not be readily available in real-time. For example, the relationship between the departure and arrival banks which we identified as important features require the identification of the start and end of each bank. In post-ops, the start and end of the bank can be consistently determined by our clustering approach but real-time identification of the start and end of a bank would be based on predicted times, as opposed to actual times, which could impact the performance of the model.

VI. Conclusion

In this paper we introduced the gate conflict prediction problem as a concrete example of a data-driven service that could be implemented in near real-time within a software layer on the other side of SWIM. The gate conflict prediction problem is relevant to the future TFDM concept as the system is required to provide a prediction of the number of gate conflicts in the upcoming SMP to help the ATC TMC calibrate parameters that govern the amount of excess taxi time that is passed back from the departure queue to the gate. For non-safety critical applications, these types of third-party decision support services could help drive efficiencies throughout the National Airspace System.

As a first step in this direction, we implemented and analyzed the performance of regression models applied to the gate conflict prediction problem. The features we used were descriptive metrics aggregated at a bank level. For the target metrics, we considered count gate conflicts and gate conflicts per arrival. The models were developed in post-ops analysis to assess our ability to detect a signal and to explore the relationship between the target variables and the features.

The analysis showed a weak but detectable signal between gate conflicts and the features based on the descriptive metrics of the bank. The most important features were determined to be features related to metrics capable of detecting early arrivals and metrics capable of detecting the interaction between the arrival and departure banks. An important avenue of future research will be to explore different features aggregated at the bank level, or the individual flight level, and assess their predictive accuracy.

Overall, the current level of prediction accuracy is not high enough for real-time decision support, but it is hard to fully evaluate the efficacy of our approach in the absence of other gate conflict prediction models. Moving forward, as we continue to develop and evolve our gate conflict prediction model these results will provide a valuable data point to anchor our expectations.

References

[1] Jung, Y., Engelland, S., Capps, A., Coppenbarger, R., Hooey, B., Sharma, S., Stevens, L., and Verma, S., "Airspace Technology Demonstration 2 (ATD-2) phase 1 Concept of Use (ConUse)," 2018.

[2] Ging, A., Engelland, S., Capps, A., Eshow, M., Jung, Y., Sharma, S., Talebi, E., Downs, M., Freedman, C., Ngo, T., Sielski, H., Wang, E., Burke, J., Gorman, S., Phipps, B., and Morgan Ruszkowski, L., "Airspace Technology Demonstration 2 (ATD-2) Technology Description Document (TDD)," 2018.

[3] Thipphavong, J., Jung, J., Swenson, H. N., Witzberger, K. E., Lin, M. I., Nguyen, J., Martin, L., Downs, M. B., and Smith, T. A., "Evaluation of the controller-managed spacing tools, flight-deck interval management, and terminal area metering capabilities for the ATM Technology Demonstration 1," *11th USA/Europe Air Traffic Management Research and Development Seminar, 2015*.

[4] Engelland, S. A., Capps, R., Day, K. B., Kistler, M. S., Gaither, F., and Juro, G., "Precision Departure Release Capability (PDRC) Final Report," 2013.

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

[6] Hayashi, M., Hoang, T., Jung, Y. C., Malik, W., Lee, H., and Dulchinos, V. L., "Evaluation of pushback decision-support tool concept for Charlotte Douglas International Airport ramp operations," *11th USA/Europe Air Traffic Management Research and Development Seminar, 2015*.

[7] 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.

[8] Jung, Y., Coupe, W., Capps, A., Engelland, S., and Sharma, S., "Field evaluation of the baseline integrated arrival, departure, surface capabilities at Charlotte Douglas Interntional Airport," *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*, 2019.

[9] "FAA Terminal Flight Data Manager (TFDM)," https://www.faa.gov/air_traffic/technology/tfdm, Accessed: 2019-02-12.

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

[11] Parke, B., Bakowski, D., Jung, Y., Lee, H., Coupe, W., and Stevens, L., "Human Factors Impact of Different Ramp Controller Scheduling Advisories for ATD-2 Surface Metering in a Human-in-the-Loop Simulation," *AIAA Aviation Forum*, 2020 (submitted).

[12] "FAA TFDM ATT J-6 System Specification Document (SSD)," https://faaco.faa.gov/index.cfm/announcement/view/21538, Accessed: 2019-10-01.

[13] "Federal Aviation Administration agile acquisition principles and practices," https://www.mitre.org/sites/default/files/publications/pr-16-1231-faa-agile-acquisition-principles-practices_0.pdf, Accessed: 2019-10-01.
[14] "FAA System Wide Information Management System (SWIM)," https://www.faa.gov/air_traffic/technology/swim/, Accessed: 2019-10-01.
[15] Hassounah, M. I. and Steuart, G. N., "Demand for aircraft gates," *Transportation Research Record*, , No. 1423, 1993.
[16] Mangoubi, R. and Mathaisel, D. F., "Optimizing gate assignments at airport terminals," *Transportation Science*, Vol. 19, No. 2, 1985, pp. 173–188.
[17] Lim, A. and Wang, F., "Robust airport gate assignment," *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, IEEE, 2005, pp. 8–pp.
[18] Kim, S. H. and Feron, E., "Impact of gate assignment on gate-holding departure control strategies," *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, IEEE, 2012, pp. 4E3–1.
[19] Bolat, A., "Procedures for providing robust gate assignments for arriving aircrafts," *European Journal of Operational Research*, Vol. 120, No. 1, 2000, pp. 63–80.
[20] Kim, S. H., Feron, E., and Clarke, J.-P., "Assigning gates by resolving physical conflicts," *AIAA Guidance, Navigation, and Control Conference*, 2009, p. 5648.
[21] "SWIM Data Feeds," https://www.faa.gov/air_traffic/technology/swim/products/, Accessed: 2019-10-01.
[22] "ATD2 SWIFT," https://www.faa.gov/air_traffic/technology/swim/swift/media/SWIFT%20Meeting%20Presentation%206%20Day%202%205_22_19.pdf, Accessed: 2019-10-01.
[23] "Session 1C: SWIM-Fused data products used by ATD-2 analysts for quantifying NAS performance and benefits (part 1)," https://aviationsystems.arc.nasa.gov/atd2-industry-workshop/presentations.html, Accessed: 2019-10-01.
[24] "scikit-learn DBSCAN," https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html, Accessed: 2019-10-01.
[25] "scikit-learn," https://scikit-learn.org/stable/, Accessed: 2019-10-01.
[26] "scikit-learn recursive feature elimination," https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE, Accessed: 2019-10-01.
[27] "scikit-learn grid search CV," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV, Accessed: 2019-10-01.
[28] "scikit-learn mean squared error," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html, Accessed: 2019-10-01.
[29] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V., "Gene selection for cancer classification using support vector machines," *Machine learning*, Vol. 46, No. 1-3, 2002, pp. 389–422.
[30] Lee, H., Coupe, J., and Jung, Y. C., "Prediction of Pushback Times and Ramp Taxi Times for Departures at Charlotte Airport," *AIAA Aviation 2019 Forum*, 2019, p. 2933.

APPENDIX

Feature #1: Count total operations - Count of departure and arrival operations

Feature #2: Count departures: Count of departure operations

Feature #3: Ratio departures ACTUAL vs. SCHEDULED: Count of ACTUAL departure operations divided by count of SCHEDULED departure operations

Feature #4: Count departures East: Count of departure operations using the East runway

Feature #5: Count departures Center: Count of departure operations using the center runway

Feature #6: Count arrivals: Count of arrival operations

Feature #7: Ratio arrivals ACTUAL vs SCHEDULED: Count of ACTUAL arrival operations divided by count of SCHEDULED arrival operations

Feature #8: Count arrivals East: Count of arrival operations using the East runway

Feature #9: Count arrivals West: Count of arrival operations using the West runway

Feature #10: Count controlled: Count of controlled departure operations that are subject to APREQ or EDCT at actual OFF

Feature #11: Percentage controlled: Count of controlled departure operations divided by total departure operations

Feature #12: Count surface metered: Count of surface metered departure operations

Feature #13: Percentage surface metered: Count of surface metered departure operations divided by total departure operations

Feature #14: Total surface metered gate hold: Sum of total gate hold for surface metered departures

Feature #15: Departure median EOBT accuracy: Departure median $EOBT$ accuracy measured as (Pilot Ready Time - $EOBT$) where the $EOBT$ is sampled 20 minutes prior to Pilot Ready Time

Feature #16: Departure EOBT standard deviation: Departure standard deviation of (Pilot Ready Time - $EOBT$) where the $EOBT$ is sampled 20 minutes prior to Pilot Ready Time

Feature #17: Actual bank overlap at runway: $\min$[Actual Landing Time] - $\min$[Actual Take Off Time], measures the overlap of ACTUAL arrival bank and ACTUAL departure bank at the runway.

Feature #18: Actual bank overlap at gate: $\min$[Undelayed In Time at Landing] - $\max$[AOBT], measures the overlap of ACTUAL arrival bank and ACTUAL departure bank at the gate.

Feature #19: Scheduled bank overlap at gate: $\min$[SIBT] - $\max$[SOBT], measures the overlap of SCHEDULED arrival bank and SCHEDULED departure bank at the gate.

Feature #20: Arrival schedule delay predicted at landing median: Arrival median (Unimpeded In Time at Landing - SIBT), measures the arrival delay compared to the schedule

at actual on

Feature #21: Arrival schedule delay predicted at landing mean: Arrival mean (Unimpeded In Time at Landing - SIBT), measures the arrival delay compared to the schedule at actual on

Feature #22: Departure schedule delay median: Departure median (AOBT - SOBT), measures the departure delay compared to the schedule at actual out

Feature #23: Departure schedule delay mean: Departure mean (AOBT - SOBT), measures the departure delay compared to the schedule at actual out

Feature #24: Departure SCHEDULE bank duration at the gate: Departure SCHEDULED bank duration $\max$[SOBT] - $\min$[SOBT] at the gate

Feature #25: Departure ACTUAL bank duration at the gate: Departure ACTUAL bank duration $\max$[AOBT] - $\min$[AOBT] at the gate

Feature #26: Departure bank duration compression at the gate: Departure ACTUAL bank duration divided by SCHEDULED bank duration at the gate

Feature #27: Arrival SCHEDULE bank duration at the gate: Arrival SCHEDULED bank duration $\max$[SIBT] - $\min$[SIBT] at the gate

Feature #28: Arrival ACTUAL bank duration at the gate: Arrival ACTUAL bank duration $\max$[AOBT] - $\min$[AOBT] at the gate

Feature #29: Arrival bank duration compression at the gate: Arrival ACTUAL bank duration divided by SCHEDULED bank duration at the gate

Feature #30: Count ACTUAL departures in arrival SCHEDULED bank: Count of departures with $AOBT \geq \min$[Undelayed In Time at Landing]

Feature #31: Percent ACTUAL departures in arrival SCHEDULED bank: Count of departures with $AOBT \geq \min$[SIBT] divided by total departure operations

Feature #32: Count ACTUAL departures in arrival ACTUAL bank: Count of departures with $AOBT \geq \min$[Undelayed In Time at Landing] divided by total departure operations

Feature #33: Count ACTUAL arrivals in departure SCHEDULED bank: Count of arrivals with Undelayed In Time at Landing $\leq \max[SOBT]$

Feature #34: Percent ACTUAL arrivals in departure SCHEDULED bank: Count of arrivals with Undelayed In Time at Landing $\leq \max[SOBT]$ divided by total arrival operations

Feature #35: Count ACTUAL arrivals in departure Actual bank: Count of arrivals with Undelayed In Time at Landing $\leq \max[AOBT]$

Feature #36: North flow: One hot encoding variable indicating North flow operations

Feature #37: South flow: One hot encoding variable indicating South flow operations